Computational Visual Media
https://doi.org/10.1007/s41095-021-0208-x Vol. 7, No. 3, September 2021, 393–405
Research Article

Flow-aware synthesis: A generic motion model for video frame interpolation

Jinbo Xing[1,2,*], **Wenbo Hu**[1,2,*], **Yuechen Zhang**[1], **and Tien-Tsin Wong**[1,2] (✉)

Abstract A popular and challenging task in video research, frame interpolation aims to increase the frame rate of video. Most existing methods employ a fixed motion model, e.g., linear, quadratic, or cubic, to estimate the intermediate warping field. However, such fixed motion models cannot well represent the complicated non-linear motions in the real world or rendered animations. Instead, we present an adaptive flow prediction module to better approximate the complex motions in video. Furthermore, interpolating just one intermediate frame between consecutive input frames may be insufficient for complicated non-linear motions. To enable multi-frame interpolation, we introduce the time as a control variable when interpolating frames between original ones in our generic adaptive flow prediction module. Qualitative and quantitative experimental results show that our method can produce high-quality results and outperforms the existing state-of-the-art methods on popular public datasets.

Keywords flow-aware; generic motion model; video frame interpolation

1 Introduction

Video frame interpolation aims to synthesize one or more intermediate frames between original frames. It is a fundamental yet important task, especially in the fields of video research and film production. Due to restrictions of camera sensors and network bandwidth, many videos on the Internet have low frame rate, especially old films and sports videos, and frame interpolation methods can greatly enhance their temporal quality. In addition, interpolation is also widely used in many other applications, including video compression [1, 2], medical imaging [3], and view synthesis [4].

Thanks to deep neural networks, we have witnessed remarkable improvements on the video frame interpolation [5–15]. However, it is still very challenging due to diverse factors, e.g., variations in lighting conditions, occlusion, and non-linear motion. Most state-of-the-art frame interpolation methods explicitly assume motions of objects or the background between consecutive frames to be linear [5, 6, 15–17], i.e., the velocity of each object moving from one frame to another is constant in screen space. Nevertheless, many motions in the real world observed in video frames complex non-linear behaviour, leading to the problem we illustrate with a simple example, the 2D path of a ball, in Fig. 1. Given the positions of the ball at four time $t = 0, \ldots, t = 3$, a linear predictive model would wrongly estimate the location of the ball at time $t = 1.5$, due to the constant velocity assumption in the linear motion model. Recently, some higher order motion models have been employed, such as a quadratic [14] or even a cubic motion model [8], to help overcome this issue. In general, the motions of camera and objects in the scene are irregular, due to variations in forces in the real world. This means that quadratic and cubic models still cannot well represent the complex motion patterns in the real world, which are beyond any fixed model. Similar examples to the one in Fig. 1 can also be found showing the limitations of these fixed motion models.

* Jinbo Xing and Wenbo Hu contributed equally to this work.
1 Department of Computer Science and Engineering, the Chinese University of Hong Kong, Hong Kong SAR, China. E-mail: J. Xing, jbxing@cse.cuhk.edu.hk; W. Hu, wbhu@cse.cuhk.edu.hk; Y. Zhang, zhangyc@link.cuhk.edu.hk; T.-T. Wong, ttwong@cse.cuhk.edu.hk (✉).
2 Shenzhen Key Laboratory of Virtual Reality and Human Interaction Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China.
Manuscript received: 2020-12-31; accepted: 2021-01-27

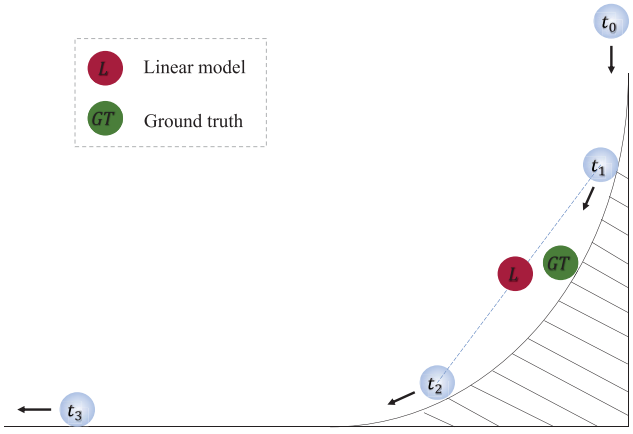

Fig. 1 Simple example of difference between actual and linearly estimated positions. Light gray: a ball's position at four time $t = 0, \ldots, 3$. Purple ball, green ball: linearly estimated position and ground truth position at time $t = 1.5$, respectively.

In this paper, we present a *flow-aware video frame interpolation* (FAI) method to address generic non-linear motions. Without making any physical assumptions about the motion, we propose an *adaptive flow prediction* module which can dynamically estimate the motions for each frame sequence from the input optical flows between successive frames. By doing so, we can bypass the limitations of fixed motion models and allow the network to learn complex motion patterns in a data-driven manner.

As a further issue, interpolating only one intermediate frame between the two adjacent frames is usually insufficient for a good representation of non-linear motion—multiple intermediate frames are needed to better represent details of non-linear motions. It is straightforward for methods with fixed physical motion models to interpolate multiple intermediate frames. The basic idea is that, a time-dependent physical equation of motion time can be used to calculate the required optical flow, so intermediate frames can be synthesized by warping the reference frame or features accordingly. In contrast, it is harder for interpolation methods without a physical motion model to interpolate frames at arbitrary time. To address this problem, we use time as the control variable for our adaptive flow prediction module to allow it to learn the motion parameterized by time. By doing so, the model can predict the required flow at arbitrary time $t \in [0, 1]$ and thus support interpolating multiple intermediate frames.

The proposed method is a generic motion model for video frame interpolation. We use four consecutive frames as the input to estimate complex motions in this paper, but it is scalable to more complex motions by using more input frames. The contributions in this paper can be summarized as

- an adaptive flow prediction module which can overcome the limitation of linear motion models and better represent the complex motions in real world scenarios;
- temporal intervals are used as a control variable to predict motion, enabling our network to synthesize intermediate frames at arbitrary time between the two input frames, and
- our method covers generic motion modeling in video interpolation, and can be further extended to support more complex motions with slight modification.

2 Related work

2.1 Flow-based methods

Flow-based methods are intuitive and have attracted much attention in recent years. The basic idea is to estimate the optical flow between consecutive original frames, and then warp the original frames or features using the required flow, which is computed from the estimated optical flow, to synthesize the intermediate frames. Our method belongs to this type of method.

As a pioneer of flow-based methods, Liu et al. [17] proposed use of a fully-convolutional network, called deep voxel flow (DVF), to predict 3D optical flow vectors across space and time for each pixel, and then synthesize the target frame by trilinear interpolation. To address inaccurate optical flows and occlusion issues, Super Slomo [16] employed two U-Nets to refine the required flow and learn soft visibility maps for blending occluded regions. CyclicGen [18] further made use of edge information and designed a novel cycle consistency loss to produce better details with less training data.

Recently, rather than implicitly training a optical flow network within the framework, more and more methods have directly employed an estimator that is well-trained with ground truth optical flow from other large-scale datasets, to get more accurate optical flow. For example, Bao et al. [5] integrated FlowNet [19] within the proposed MEMC-Net, and

Xue et al. [15] adopted SpyNet [20] in their proposed ToFlow and DAIN [6] utilized PWC-Net [21] as its flow estimator. To further enhance details and address challenging scenarios, e.g., occlusion and large motions, Niklaus and Liu [11] and Yuan et al. [22] proposed to not only warp the input frames but also the contextual features extracted by ResNet [23]. To address blending of occluded regions, DAIN [6] leveraged depth information, while SoftSplat [12] learned blending weights for each overlapped pixel from a depth-related importance mask. Park et al. [24] proposed the BMBC method to estimate the bilateral motions using a bilateral cost volume to obtain more accurate optical flow.

All of the above methods explicitly or implicitly assume the motions of objects or the background in the input frames are linear. A linear model can well approximate simple motions, but it is insufficient for more complex motions, as noted. To represent more complex motions, Xu et al. [14] used a quadratic motion model, and Chi et al. [8] presented a cubic motion model. However, the movements of objects and background are irregular in the real world, due to variations in forces. No matter whether linear or quadratic or cubic motion models are used, all are still fixed physical motion models that cannot well represent the extremely complex motion patterns in the real world. Unlike the above methods, our method can dynamically estimate the motion model for each frame sequence from the input optical flows between successive frames.

2.2 Kernel-based methods

Unlike flow-based methods that explicitly estimate motions, kernel-based methods deal with motions in an implicit way. They directly estimate kernels for convolving with input frames to produce intermediate frames. As a pioneer, Long et al. [25] proposed to regress the target frame from the two input frames using a CNN, but the results always tended to be blurry. To produce more visually pleasing frames, Niklaus et al. [13] proposed to estimate a 2D convolution kernel for each pixel to capture the motion. The output frame can be synthesized by locally convolving the two input frames with the estimated kernels. It can produce intermediate frames with sharper edges, but it is extremely memory-consuming, as it estimates an independent large kernel (41×41) for every output pixel. To improve memory efficiency, Niklaus et al. [26] proposed use of 1D separable convolution kernels instead of 2D kernels, but it still fails to synthesize plausible results for motion larger than the kernel size. Recently, AdaCoF [9] proposed to estimate not only the kernel weights but also offsets for each pixel to support larger motions. DSepConv [7] adopted deformable separable convolution [27, 28] to replace conventional convolution to address large motions with a smaller kernel size.

Since all the kernel-based methods use convolution kernels to implicitly model the motion between frames, they cannot directly interpolate multiple intermediate frames between the two input frames. Although we can recursively feed the interpolated frames back into their model to produce multiple intermediate frames, this approach leads to error accumulation. Also, this solution implicitly assumes a linear motion model, so cannot well represent complex motions in the real world, as discussed above.

2.3 Other methods

Besides flow-based and kernel-based methods, several other novel methods have been proposed to interpolate frames. Meyer et al. [29] proposed a phase-based method that combines phase information across the levels of a multi-scale pyramid. It provides an efficient alternative to optical flow, but large motions of high frequency components cannot be well represented by the estimated phase. In order to alleviate this issue, Meyer et al. [10] proposed PhaseNet to combine the phase-based motion representation with a neural network decoder, to improve robustness. Recently, FeFlow [30] was devised to synthesize intermediate frames in a structure-to-texture manner. It divides the video frame interpolation task into two steps: structure-guided interpolation and texture-refinement; an attention mechanism is employed in their method to better handle occlusions. CAIN [31] adopted channel attention to spread the information in feature maps into multiple channels and extracted motion clues from them.

Like kernel-based methods, these methods also have to adopt recursive processing to interpolate multiple intermediate frames, again leading to error accumulation and implicitly assuming a linear motion model, while our method can dynamically estimate non-linear motion models for each frame sequence and directly interpolate multiple intermediate frames.

3 Approach

3.1 Overview

The target of frame interpolation is to increase the frame rate of video. Given four consecutive video frames I_{-1}, I_0, I_1, and I_2, our goal is to interpolate a frame I_t that is temporally between I_0 and I_1. The overall pipeline of our method is shown in Fig. 2. Our method can be divided into two stages, adaptive flow prediction and frame synthesis.

To better use the input information, we regard both I_0 and I_1 as reference frames. As shown in Fig. 2, for each reference frame, we estimate a group of basic optical flows from it to the other three input frames, denoted by $\{f_{0\to-1}, f_{0\to1}, f_{0\to2}\}$ for I_0, and $\{f_{1\to2}, f_{1\to0}, f_{1\to-1}\}$ for I_1. To model the complicated non-linear motion, each group of basic flows and the time t are then fed into the proposed adaptive flow prediction (AFP) module to predict the optical flow from the reference frame to the target frame, that is represented as $f_{0\to t}$ and $f_{1\to t}$ for the two reference frame, respectively. Here, the input time t is used to control the time at which the required flow is predicted, so our method can interpolate multiple intermediate frames as needed.

If we directly warp the reference frames (I_0 and I_1) with the required flows to produce the warped frames ($\hat{I}_0$ and $\hat{I}_1$) and fuse them to give the final results, the results tend to be blurred. Therefore, we further use a pyramid context extractor to extract multi-scale contextual features for the reference frames (I_0 and I_1), denoted $\{F_0^1, F_0^2, F_0^3\}$ and $\{F_1^1, F_1^2, F_1^3\}$ for I_0 and I_1, respectively. We then employ a forward warping layer [12] to warp not only the reference frames (I_0 and I_1) but also their multi-scale contextual features ($\{F_0^1, F_0^2, F_0^3\}$ and $\{F_1^1, F_1^2, F_1^3\}$). Finally, the warped reference frames ($\hat{I}_0$, $\hat{I}_1$) and warped pyramid contextual features are fed into the frame synthesis network to produce a residual map between the ground truth and the average blending of frame $\hat{I}_0$ and $\hat{I}_1$. Our final prediction of the interpolated frame is obtained by summing the residual map and the average blending of frame $\hat{I}_0$ and $\hat{I}_1$. We provide the details of each component of our approach in the following sections.

3.2 Adaptive flow prediction

The goal of the adaptive flow prediction (AFP) module is to dynamically estimate the motions for each frame sequence from the basic optical flows between successive input frames, to better represent complex motions than traditional linear, quadratic, or cubic motion models. We employ the off-the-shelf method PWC-Net [21] to produce the basic flows, as it is a state-of-the-art optical flow estimation method widely used in recent research. For each reference frame (I_0 and I_1), we estimate three different basic optical flows. Note that to support interpolating multiple intermediate frames, we need to control the target time of the required flow. Therefore, we also

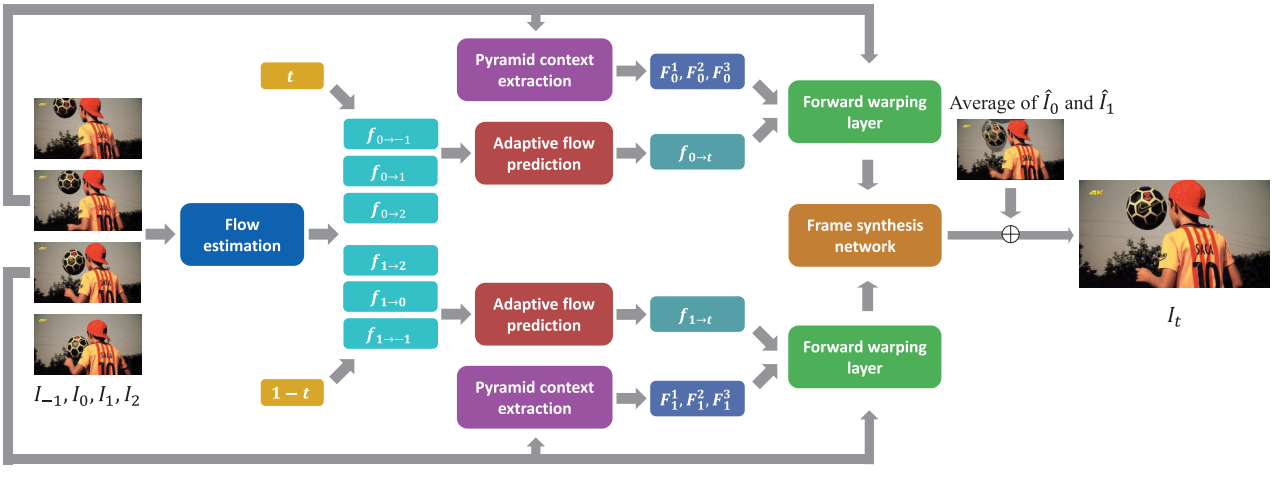

Fig. 2 Overview of our FAI approach. Given four consecutive input frames (I_{-1}, I_0, I_1, and I_2), an off-the-shelf flow estimator first estimates two basic optical flow groups, $\{f_{0\to-1}, f_{0\to1}, f_{0\to2}\}$, and $\{f_{1\to2}, f_{1\to0}, f_{1\to-1}\}$. These groups are concatenated with corresponding time and fed into the adaptive flow prediction (AFP) module to produce flows $f_{0\to t}$ and $f_{1\to t}$. These are used to warp the reference frames (I_0 and I_1) and their contextual features extracted by a pyramid context extraction module, which are then fed to the frame synthesis network to produce the final target frame.

expand the target time t into a tensor of shape of $H \times W \times 1$, where H and W are the height and width of input frames, and concatenate it with the three optical flows as the input to the AFP module to generate the flows $f_{0\to t}$ and $f_{1\to t}$. Mathematically, this procedure can be represented as

$$f_{0\to t} = \text{AFP}(f_{0\to -1}, f_{0\to 1}, f_{0\to 2}, t) \quad (1)$$

$$f_{1\to t} = \text{AFP}(f_{1\to 2}, f_{1\to 0}, f_{1\to -1}, 1-t) \quad (2)$$

By analyzing the motion patterns within the input basic flows, the AFP module estimates the respective required bi-directional optical flows from reference frames I_0 and I_1 to the intermediate target frame I_t. Since multilayer perceptrons (MLP) are known to powerfully approximate functions, we employ them to approximate the complex non-linear motions of the input consecutive frames. As shown in Eq. (1), the input basic flow group $\{f_{0\to -1}, f_{0\to 1}, f_{0\to 2}\}$ and the output required flow $f_{0\to t}$ are spatially aligned with the same reference frame I_0; a similar observation holds for Eq. (2). Therefore, we can use a 1×1 convolution to realize the temporal MLP. Without any spatially resampling, the AFP module only needs to learn the motion patterns from the input basic flow group and to estimate the required flow values.

The architecture of our AFP module is shown in Fig. 3; it consists of six convolution layers. The first five layers have Leaky Rectified Linear Units (LeakyReLU) [32] as activation function; the last layer acts as the output layer, so no activation function is applied. The kernel size in each convolution layer is set to one, as discussed above. The shape of the feature map in each layer is $H \times W \times C$, and the shape of feature maps within the AFP module remains unchanged, while the number of channels C increases at the beginning and decreases to two at the end, as the predicted flow for each pixel should be a 2D vector. The AFP module can learn and predict $f_{0\to t}$ from the stacked flows and time t, so we refer to the proposed method (FAI) as a flow-aware synthesis method.

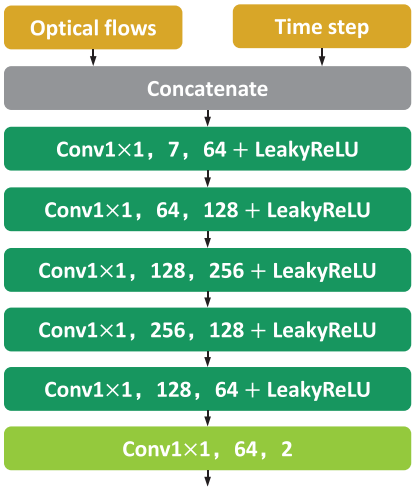

Fig. 3 Network architecture of the adaptive flow prediction (AFP) module. Its input is the basic flow group and the target time, while the output is the required optical flow at the target time. Numbers inside each convolution layer indicate the number of input and output channels, respectively.

Note that our method is scalable to addressing more complex motions. With slight modification, further basic flows can be fed into our AFP module to extend its capability to approximate more complex motions. It can be described as

$$f_{0\to t} = \text{AFP}(f_{0\to t_1}, \ldots, f_{0\to t_n}, t) \quad (3)$$

where t_1 to t_n denote a sequence of time, assuming frames at those time steps can be used as inputs. $f_{1\to t}$ can be determined in a similar way.

3.3 Frame synthesis

Having predicted the required optical flow using the AFP module, we need to warp the reference frames (I_0 and I_1) to the target time step. To better use the information inside the reference frames, we not only warp them directly but also their multi-scale contextual features to the target time step. We employ the pyramid feature extraction network proposed by Niklaus and Liu [12] to extract multi-scale contextual features of the reference frames at three scales F^1, F^2, and F^3. The multi-scale contextual features for I_0 and I_1 are denoted by $\{F_0^1, F_0^2, F_0^3\}$ and $\{F_1^1, F_1^2, F_1^3\}$, respectively.

Our AFP module generates flows $f_{0\to t}$ and $f_{1\to t}$ aligned with the reference frames I_0 and I_1, respectively. Therefore, forward warping is a more suitable way to get the warped frames and contextual features, rather than backward warping. We adopt the differentiable forward warping layer proposed by softmax splatting [12], so the whole framework can be trained jointly. Note that we resize the predicted flows ($f_{0\to t}$ and $f_{1\to t}$) to each scale in the pyramids of multi-scale contextual features, and rescale the flow vector values accordingly, to allow us to warp the pyramidal contextual features.

Forward warping can leave holes due to occlusion. To fill in the missing information and enhance the details in the final synthesized frame, we employ

GridNet [33] as our frame synthesis network. GridNet contains three rows and six columns. Inspired by Niklaus and Liu [11], we adopt bilinear upsampling in GridNet to avoid checkerboard artifacts, and incorporate parametric rectified linear units to stabilize training. Specifically, we concatenate the warped input frames ($\hat{I}_0$, $\hat{I}_1$) and the first level of contextual features ($\hat{F}^1$) as input to the first row of GridNet, and feed the second ($\hat{F}^2$) and third ($\hat{F}^3$) level contextual features into the second and third rows of GridNet, respectively. To encourage convergence, we let the frame synthesis network learn the residual map between the ground truth target frame and the average blending of warped reference frames ($\hat{I}_0$ and $\hat{I}_1$).

3.4 Loss functions

Inspired by Refs. [9, 26], we consider two different types of loss functions in our method, color loss $\mathcal{L}_1$ and combination loss $\mathcal{L}_{\text{com}}$. The color loss and combination loss make our network focus on quantitative quality and visual quality, respectively.

3.4.1 Color loss

The color loss $\mathcal{L}_1$ is defined as the L_1 norm of the pixel-wise color difference. Alternatively, following recent works [6, 24], we optimize the L_1 norm using the Charbonnier penalty function [34]. Mathematically, it can be written:

$$\mathcal{L}_1 = \| \rho(I_{\text{out}} - I_{\text{gt}}) \|_1 \tag{4}$$

$$\rho(x) = \sqrt{x^2 + \epsilon^2} \tag{5}$$

where ϵ is set to 10^{-6} in our experiments.

3.4.2 Combination loss

It has been shown that introducing perceptual loss into image generation tasks can produce more visually pleasing results and sharper edges [35, 36]. The basic idea is to supervise the synthesized results in the feature domain. Various feature extractors ϕ can be utilized to map the synthesized frame into feature space to compute the perceptual loss. We empirically adopt relu4_4 layer of the VGG-19 network here. However, using perceptual loss by itself may lead to color distortion. Thus, we combine the $\mathcal{L}_1$ loss and perceptual loss together to form a combination loss:

$$\mathcal{L}_{\text{com}} = \| \rho(I_{\text{out}} - I_{\text{gt}}) \|_1 + \lambda \| \phi(I_{\text{out}}) - \phi(I_{\text{gt}}) \|_2 \tag{6}$$

where λ is set to 2×10^{-5} in our experiments.

4 Experiments

In this section, we provide implementation details and a comparison with other state-of-the-art methods on widely used datasets. We also conduct several ablation study experiments to evaluate the effectiveness of the modules in our method.

4.1 Implementation details

Our training dataset consisted of two parts. The first contained 25 video clips with a frame rate of 240 fps and resolution of 720×1280, collected from YouTube. These video clips were diverse in terms of action and scene type. However, the cameras in these videos were mostly still. Thus we randomly selected some consecutive frames from GOPRO [37] and Adobe240 [38] datasets as our another part of the training dataset. These were recorded with hand-held cameras and therefore contain more complex motions. Consequently, the final training dataset consisted of 14,819 training samples, each sample with 25 consecutive frames following QVI [14]. Our model took the 1st, 9th, 17th, and 25th frames as inputs (I_{-1}, I_0, I_1, and I_2) to synthesize 7 frames I_t from 10th to 16th as ground-truth I_{gt} at time steps $t = 0.125, 0.25, 0.5, \ldots, 0.875$. During the training phase, we resized the frames to 360×640 and randomly cropped them to 256×256. We also performed data augmentation by randomly flipping frames vertically and horizontally. In order to increase the diversity of our dataset, we randomly changed the temporal order with probability 0.5. Our experiments were performed on a single NVIDIA TITAN RTX GPU.

Since we adopt forward warping, there are some gaps in $\hat{I}_0$ and $\hat{I}_1$. We experienced a degradation in performance and hard convergence of the model when jointly training the whole network. In order to preserve the functionality of our AFP module, we chose to train AFP first supervised by weak ground truth optical flow $f_{0\to t}$ and $f_{1\to t}$. The training loss for AFP was $\mathcal{L}_1$. Next, we trained the whole network with the AFP module fixed. To train our network, we used AdaMax [39] with $\beta = (0.9, 0.999)$ and mini-batch size of 8 samples. The initial learning rate was set to 2×10^{-4} and reduced by a factor of 0.5 for every 30 epochs. We trained our network with flow estimation and AFP module fixed for 70 epochs and then fine tuned the flow estimation for another

10 epochs. We will release our source code upon publication.

4.2 Evaluation datasets and metrics.

4.2.1 Overview

We evaluated our approach on four widely used datasets, including two multi-frame interpolation datasets: GOPRO and Adobe240, and two single-frame interpolation datasets: DAVIS [40] and Vimeo90K septuplet [15].

For quantitative evaluation, we used peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [41], and interpolation error (IE) [42] between I_t and I_{gt} on the evaluation datasets, adopting root-mean-squared (RMS) difference for IE in our experiments.

4.2.2 Multi-frame interpolation datasets

The GOPRO dataset consists of 33 high-quality videos while Adobe240 consists of 133 videos, both recorded by high-speed hand-held cameras and designed for benchmarking deblurring tasks. The frame rate is 240 fps and resolution is 720×1280. We extracted 4275 samples of 25 consecutive frames from GOPRO and 8702 from Adobe240 as set following QVI and randomly separated the samples into training and testing parts: for GOPRO, 2775:1500; for Adobe240, an equal split. Following the test settings in QVI [14], we kept the resolution to 720×1280 for GOPRO and resized frames for Adobe to 360×640 during testing. For each sample, the 1st, 9th, 17th, 25th frames(I_{-1}, I_0, I_1, and I_2) are used to synthesize the frames I_t from 10th to 16th.

4.2.3 Single-frame interpolation datasets

DAVIS is a video dataset originally designed for segmentation tasks, with a frame rate of 30 fps. Xu et al. [14] previously extracted 2849 quintuples (I_{-1}, I_0, I_1 I_2 as inputs and $I_{0.5}$ as the target) from DAVIS. We used this data and resized the frames to 480×856 for our evaluation. Vimeo90K septuplet data was originally designed for video denoising, deblocking, and super-resolution, and contains 7824 samples of 7 consecutive frames with a resolution of 256×448. We took the 1st, 3rd, 5th, and 7th frames as inputs (I_{-1}, I_0, I_1, and I_2) to synthesize the 4th frame, corresponding to $I_{0.5}$ in our experiments.

4.3 Comparisons with state-of-the-arts

4.3.1 Overview

We compared our method with five state-of-the-art interpolation methods, including SepConv-$\mathcal{L}_1$[26], Super SloMo [16], QVI [14], DAIN [6], and AdaCoF [9]. We used the authors' released codes for Super SloMo, QVI, DAIN, and AdaCoF and corresponding retrained versions on our training dataset. Since the training code of SepConv is not publicly available, we could not retrain it and directly used the released model in our experiments. Note that our model interpolates frames at arbitrary input time. We compared the above methods on both the multi-frame and single-frame interpolation datasets.

4.3.2 Quantitative evaluation

A quantitative comparison for the multi-frame interpolation datasets is shown in Table 1. Following QVI [14], we evaluated different methods in two settings. Evaluation metrics for the 4th frame are denoted "center", while the average over all 7 interpolated frames is denoted "whole". We can see that on GOPRO and Adobe240 datasets, the proposed method consistently and significantly outperforms all the other methods which use a linear physical motion model assumption. Moreover, our method achieves 0.3 dB and 0.8 dB PSNR gains respectively compared with QVI which assumes

Table 1 Quantitative comparison to state-of-the-art methods on GOPRO and Adobe240 multi-frame interpolation datasets. Numbers in **bold** and underlined represent the first and the second best performances, respectively

Method	GOPRO						Adobe240					
	Whole			Center			Whole			Center		
	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE
SepConv-$\mathcal{L}_1$	29.36	0.9193	10.08	27.40	0.8898	12.38	32.25	0.9542	7.42	30.88	0.9398	8.77
Super SloMo	28.67	0.9180	10.41	27.61	0.8953	11.94	30.63	0.9461	8.32	30.72	0.9412	8.57
AdaCoF	28.49	0.9058	10.93	26.96	0.8824	12.78	31.05	0.9380	8.47	30.11	0.9274	9.47
DAIN	29.35	0.9221	9.93	27.66	0.8956	11.98	32.01	0.9537	7.50	30.86	0.9412	8.64
QVI	30.29	**0.9415**	8.59	28.70	0.9190	10.32	32.43	0.9649	6.74	31.73	0.9579	7.40
Ours-$\mathcal{L}_{\text{com}}$	**30.40**	0.9394	8.60	28.86	0.9192	10.26	**33.24**	0.9655	**6.42**	32.57	0.9597	7.01
Ours-$\mathcal{L}_1$	30.38	0.9401	**8.58**	**29.00**	**0.9215**	**10.11**	33.17	**0.9658**	6.45	**32.70**	**0.9608**	**6.90**

a quadratic model for the motions. Similarly, our method also performs favorably against the other methods except for QVI on the single-frame interpolation datasets, as shown in Table 2.

4.3.3 Properties of methods

Following the analysis method in DSepConv [7], we analyse the properties of different methods in the following ways:

- number of parameters (Param.), in millions;
- number of input frames (Input);
- sub-networks used, including flow, kernel, context, mask.

The results are shown in Table 3. Enc-Dec denotes the self-trained flow estimation module with Encoder–Decoder network architecture. LH denotes the learned hierarchical feature extraction module defined in DAIN. In addition, the kernel in flow-based methods with bilinear interpolation operations is denoted bilinear(k), where k is the kernel size. In particular, we can see that our method uses the fewest parameters, only about half of the number used in other methods.

4.3.4 Qualitative evaluation

A qualitative evaluation can better show the visual differences in results of different methods. A comparison of our method with other state-of-the-art methods on some challenging scenarios is shown in Fig. 4. These scenarios contain complex motions with both translational and rotational motion; our adaptive flow prediction module can effectively address such complicated non-linear motions.

We can see that the results of our methods are more visually pleasing in Fig. 4, e.g., in the top row, our method synthesizes the flamingo legs more clearly than other methods. In the 3rd row, the frame generated by our method contains the whole wheel and less distortion in the crosswalk compared to results generated by other methods. Also, our method can better address occlusions, as shown in the bottom row, where our synthesized frame contains a clear background and fewer artifacts near edges compared to results of other methods.

Moreover, as intended, combination loss $\mathcal{L}_{\text{com}}$ produces better visual results than color loss $\mathcal{L}_1$, as shown in Fig. 4, while $\mathcal{L}_1$ outperforms the $\mathcal{L}_{\text{com}}$ quantitatively as shown in Table 1.

4.3.5 Quality consistency evaluation

Besides average frame quality, quality consistency of frames along the time axis is also important to video quality. If the frame quality varies too much over time, people will experience the video to be flickering, leading to discomfort. To evaluate consistency of quality, we computed PSNR values at each time step of the results from different methods on the Adobe240 dataset: see Fig. 5. For all methods, the PSNR values at the center time ($t = 4$) tend to be lower than at the marginal time ($t = 1$, $t = 7$). This is because the time difference between the input frames and the target frames at the marginal time is lower than that at the center time. The PSNR curve for our method is consistently above the curves of other methods, and it is also smoother, indicating that our method can produce both better frame quality and more consistent results.

4.4 Ablation study

To evaluate the effectiveness of the individual components of our model, we conducted ablation studies using the multi-frame interpolation datasets. We considered the following variants of our method:

(1) Linear w/o t: without the adaptive flow prediction module;

(2) Linear w/ t: without the adaptive flow prediction module but with time t as input;

Table 2 Quantitative comparison to state-of-the-art methods on the DAVIS and Vimeo90K single-frame interpolation datasets. Numbers marked in **bold** and underlined represent the first and the second best performances, respectively

Method	DAVIS			Vimeo90K		
	PSNR	SSIM	IE	PSNR	SSIM	IE
SepConv-$\mathcal{L}_1$	26.20	0.8567	15.66	33.72	0.9678	6.16
Super SloMo	26.05	0.8534	15.77	33.04	0.964	6.53
AdaCoF	25.95	0.8493	15.86	33.01	0.9619	6.62
DAIN	27.30	0.8834	13.55	33.63	0.9681	6.16
QVI	**27.60**	**0.8927**	**12.85**	**34.39**	**0.9725**	**5.60**
Ours-$\mathcal{L}_{\text{com}}$	27.25	0.8828	13.66	33.43	0.9692	6.21
Ours-$\mathcal{L}_1$	27.44	0.8870	13.39	33.69	0.9703	6.03

Table 3 Properties of different video frame interpolation methods

Method	Param. (M)	Input	Sub-networks			
			Flow	Kernel	Context	Mask
SepConv-$\mathcal{L}_1$	21.60	2	×	Learned(51)	×	×
Super SloMo	39.61	2	Enc-Dec	Bilinear(2)	×	✓
AdaCoF	21.84	2	×	Learned(5)	×	✓
DAIN	24.03	2	PWC-Net	Learned(4)	LH	×
QVI	29.23	4	PWC-Net	Bilinear(2)	×	×
Ours	**12.83**	4	PWC-Net	Bilinear(2)	Pyramid	×

Fig. 4 Visual comparison of results of our method and other state-of-the-art frame interpolation methods on the DAVIS dataset.

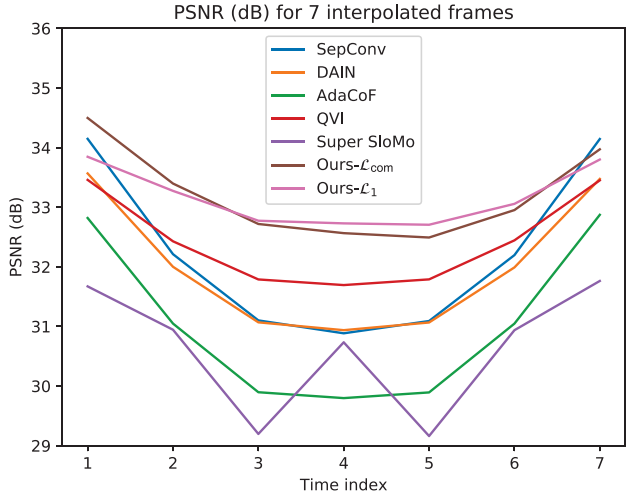

Fig. 5 Quality consistency evaluation. PSNR values for synthesized frames over time, for different methods, on the Adobe240 dataset.

(3) Ours w/o t: without t as input;
(4) Ours: the full proposed model.

In (1) and (2), since there is no adaptive flow prediction module to predict the required flow $f_{0\to t}$ and $f_{1\to t}$ from reference frames to target frames, we just apply the linear flow combination method, to produce intermediate flows as $tf_{0\to 1}$ and $(1-t)f_{1\to 0}$. In (1) and (3), the network cannot synthesize the frame at an arbitrary time, it can only generate a single in-between frame $I_{0.5}$, but we can recursively interpolate 7 frames for (1). Note that all the variants are trained with $\mathcal{L}_1$ loss.

The performance of the above variants was evaluated on the GOPRO, Adobe240, DAVIS, and Vimeo90K datasets, with results as shown in Table 4. We can see that the linear variants of our methods (Linear w/o t and Linear w/ t) cannot compete with our full methods, as they cannot well represent the complex motions in the dataset. Moreover, the performance of Ours w/o t for center frame is slightly better than Ours, because the network can pay more attention to the quality of the center frame if we do not require it to interpolate other intermediate frames.

Table 4 Ablation study results for both multi-frame interpolation datasets (GOPRO and Adobe240) and single-frame interpolation datasets (DAVIS and Vimeo90K). Numbers in **bold** and underlined represent the first and the second best performances, respectively.

Method	GOPRO						Adobe240						DAVIS			Vimeo90K		
	Whole			Center			Whole			Center			Center			Center		
	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE	PSNR	SSIM	IE
Linear w/o t	29.48	0.9222	9.81	27.72	0.8950	11.97	31.62	0.9503	7.67	30.93	0.9414	8.56	27.25	0.8834	13.54	33.09	0.9659	6.45
Linear w/ t	29.35	0.9208	9.98	27.68	0.8941	12.03	31.99	0.9537	7.50	30.97	0.9417	8.55	27.28	0.8836	13.57	33.14	0.9665	6.44
Ours w/o t	—	—	—	**29.52**	**0.9282**	**9.52**	—	—	—	**33.05**	**0.9637**	**6.57**	**27.98**	**0.8977**	**12.56**	**33.99**	**0.9716**	**5.82**
Ours	**30.38**	**0.9401**	**8.58**	29.00	0.9215	10.11	**33.16**	**0.9658**	**6.45**	32.70	0.9608	6.90	27.44	0.8870	13.39	33.69	0.9703	6.03

Therefore, Ours w/o t would be a better choice if one only want to interpolate the center frame.

Besides the average frame quality, the quality consistency of frames is also important as mentioned above. Therefore, we also evaluate the quality consistency of our method and its variants. We use them to interpolate the intermediate seven frames, and compute the PSNR value for each frame on the Adobe240 dataset, and plot the PSNR values against the time index in Fig. 6. We can see that no matter with $\mathcal{L}_1$ or $\mathcal{L}_{\text{com}}$ loss, our method can always produce more consistent results for all the consecutive frames compared with the linear variants of our method.

To visualize the difference of estimated flows from the linear model and our proposed AFP module, we show two examples in Fig. 7 from the DAVIS dataset that contain non-linear motions. The top row shows the optical flow $f_{0\to0.5}$, while the second row shows $f_{1\to0.5}$. We estimate flows between the input frames, e.g., $\{I_0, I_{0.5}\}$ and $\{I_1, I_{0.5}\}$, as the approximate ground truth (first column). The optical flows in the second and last columns are produced by the linear model and our proposed AFP module, respectively. We visualize the optical flow in HSV color space, where hue indicates direction. We can see that the hue of the visualization predicted by AFP is closer to the ground truth, indicating that AFP can better address non-linear motions than a conventional linear motion model.

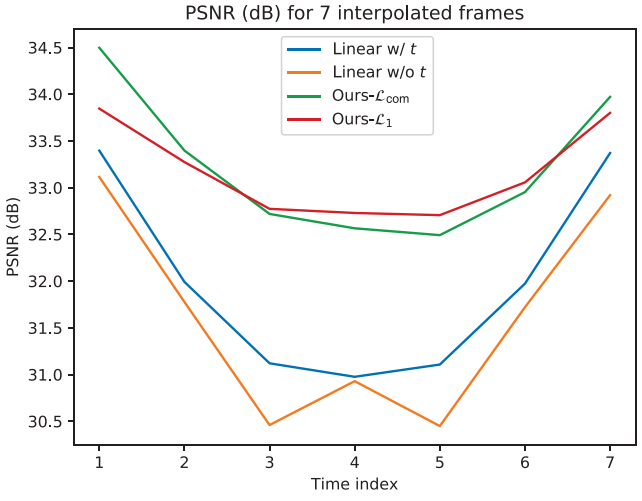

Fig. 6 Quality consistency evaluation. PSNR for synthesized frames versus time for our method and its variants on the Adobe240 dataset.

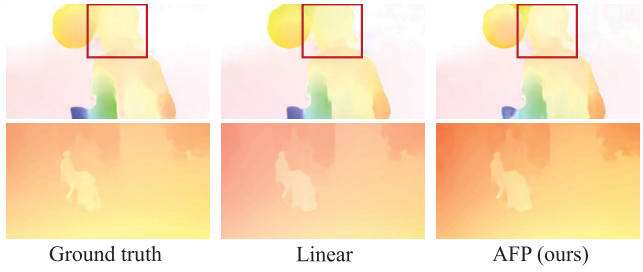

Fig. 7 Visualization of optical flows from different methods on two examples from the DAVIS dataset. Above: optical flow $f_{0\to0.5}$. Below: $f_{1\to0.5}$.

5 Conclusions

We have presented a flow-aware multi-frame interpolation method to address the complicated non-linear motions in the real world by dynamically learning the motions for each frame sequence with our proposed adaptive flow prediction module. By introducing time as a control variable for the adaptive flow prediction module, our method can interpolate multiple intermediate frames from consecutive input frames. Such that the frame interpolated videos can better present the complex non-linear motions. Our generic motion model is scalable and can be extended to support more complicated motions with more input frames. Extensive experiments show the quality of our results. Both qualitative and quantitative experimental results indicate that our methods outperform existing state-of-the-art methods on widely used datasets.

Acknowledgements

This project was supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC General Research Fund (Project No. CUHK 14201017), Shenzhen Science and Technology Program (No. JCYJ20180507182410327), and the Science and Technology Plan Project of Guangzhou (No. 201704020141).

Electronic Supplementary Material Supplementary material is available in the online version of this article at `https://doi.org/10.1007/s41095-021-0208-x`.

References

[1] Lu, G.; Zhang, X. Y.; Chen, L.; Gao, Z. Y. Novel integration of frame rate up conversion and HEVC coding based on rate-distortion optimization. *IEEE Transactions on Image Processing* Vol. 27, No. 2, 678–691, 2018.

[2] Wu, C.-Y.; Singhal, N.; Krähenbühl, P. Video compression through image interpolation. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11212*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer Cham, 425–440, 2018.

[3] Karargyris, A.; Bourbakis, N. Three-dimensional reconstruction of the digestive wall in capsule endoscopy videos using elastic video interpolation. *IEEE Transactions on Medical Imaging* Vol. 30, No. 4, 957–971, 2011.

[4] Flynn, J.; Neulander, I.; Philbin, J.; Snavely, N. Deep stereo: Learning to predict new views from the world's imagery. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5515–5524, 2016.

[5] Bao, W. B.; Lai, W. S.; Zhang, X. Y.; Gao, Z. Y.; Yang, M. H. 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, 933–948, 2021.

[6] Bao, W. B.; Lai, W. S.; Ma, C.; Zhang, X. Y.; Gao, Z. Y.; Yang, M. H. Depth-aware video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3698–3707, 2019.

[7] Cheng, X. H.; Chen, Z. Z. Video frame interpolation via deformable separable convolution. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 7, 10607–10614, 2020.

[8] Chi, Z. X.; Mohammadi Nasiri, R.; Liu, Z.; Lu, J. W.; Tang, J.; Plataniotis, K. N. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12372*. Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 107–123, 2020.

[9] Lee, H.; Kim, T.; Chung, T. Y.; Pak, D.; Ban, Y.; Lee, S. AdaCoF: Adaptive collaboration of flows for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5315–5324, 2020.

[10] Meyer, S.; Djelouah, A.; McWilliams, B.; Sorkine-Hornung, A.; Gross, M.; Schroers, C. PhaseNet for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 498–507, 2018.

[11] Niklaus, S.; Liu, F. Context-aware synthesis for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1701–1710, 2018.

[12] Niklaus, S.; Liu, F. Softmax splatting for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5436–5445, 2020.

[13] Niklaus, S.; Mai, L.; Liu, F. Video frame interpolation via adaptive convolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2270–2279, 2017.

[14] Xu, X.; Siyao, L.; Sun, W.; Yin, Q.; Yang, M.-H. Quadratic video interpolation. In: Proceedings of the Advances in Neural Information Processing Systems, 2019.

[15] Xue, T. F.; Chen, B. A.; Wu, J. J.; Wei, D. L.; Freeman, W. T. Video enhancement with task-oriented flow. *International Journal of Computer Vision* Vol. 127, No. 8, 1106–1125, 2019.

[16] Jiang, H.; Sun, D.; Jampani, V.; Yang, M.-H.; Learned-Miller, E.; Kautz, J. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9000–9008, 2018.

[17] Liu, Z. W.; Yeh, R. A.; Tang, X. O.; Liu, Y. M.; Agarwala, A. Video frame synthesis using deep voxel flow. In: Proceedings of the IEEE International Conference on Computer Vision, 4473–4481, 2017.

[18] Liu, Y.-L.; Liao, Y.-T.; Lin, yen-yu; Chuang, Y.-Y. Deep video frame interpolation using cyclic frame generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 8794–8802, 2019.

[19] Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazirbas, C.; Golkov, V.; van Der Smagt, P.; Cremers, D.; Brox, T. FlowNet: Learning optical flow with convolutional networks. In: Proceeedings of the IEEE International Conference on Computer Vision, 2758–2766, 2015.

[20] Ranjan, A.; Black, M. J. Optical flow estimation using a spatial pyramid network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2720–2729, 2017.

[21] Sun, D.; Yang, X.; Liu, M.-Y.; Kautz, J. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8934–8943, 2018.

[22] Yuan, L. Z.; Chen, Y. B.; Liu, H. T.; Kong, T.; Shi, J. B. Zoom-in-to-check: Boosting video interpolation via instance-level discrimination. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12175–12183, 2019.

[23] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[24] Park, J.; Ko, K.; Lee, C.; Kim, C. S. BMBC: Bilateral motion estimation with bilateral cost volume for video interpolation. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12359*. Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 109–125, 2020.

[25] Long, G. C.; Kneip, L.; Alvarez, J. M.; Li, H. D.; Zhang, X. H.; Yu, Q. F. Learning image matching by simply watching video. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9910*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 434–450, 2016.

[26] Niklaus, S.; Mai, L.; Liu, F. Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE International Conference on Computer Vision, 261–270, 2017.

[27] Dai, J. F.; Qi, H. Z.; Xiong, Y. W.; Li, Y.; Zhang, G. D.; Hu, H.; Wei, Y. C. Deformable convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, 764–773, 2017.

[28] Zhu, X. Z.; Hu, H.; Lin, S.; Dai, J. F. Deformable ConvNets V2: More deformable, better results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9300–9308, 2019.

[29] Meyer, S.; Wang, O.; Zimmer, H.; Grosse, M.; Sorkine-Hornung, A. Phase-based frame interpolation for video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1410–1418, 2015.

[30] Gui, S. R.; Wang, C. Y.; Chen, Q. H.; Tao, D. C. FeatureFlow: Robust video interpolation via structure-to-texture generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 14001–14010, 2020.

[31] Choi, M.; Kim, H.; Han, B.; Xu, N.; Lee, K. M. Channel attention is all you need for video frame interpolation. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 7, 10663–10671, 2020.

[32] Maas, A. L.; Hannun, A. Y.; Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning, 2013.

[33] Fourure, D.; Emonet, R.; Fromont, E.; Muselet, D.; Tremeau, A.; Wolf, C. Residual conv-deconv grid network for semantic segmentation. In: Proceedings of the British Machine Vision Conference, 181.1–181.13, 2017.

[34] Charbonnier, P.; Blanc-Feraud, L.; Aubert, G.; Barlaud, M. Two deterministic half-quadratic regularization algorithms for computed imaging. In: Proceedings of the International Conference on Image Processing, 168–172, 1994.

[35] Dosovitskiy, A.; Brox, T. Generating images with perceptual similarity metrics based on deep networks. In: Proceedings of the 30th Conference on Neural Information Processing Systems, 2016.

[36] Johnson, J.; Alahi, A.; Li, F. F. Perceptual losses for real-time style transfer and super-resolution. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9906*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 694–711, 2016.

[37] Nah, S.; Kim, T. H.; Lee, K. M. Deep multi-scale convolutional neural network for dynamic scene deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 257–265, 2017.

[38] Su, S. C.; Delbracio, M.; Wang, J.; Sapiro, G.; Heidrich, W.; Wang, O. Deep video deblurring for hand-held cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 237–246, 2017.

[39] Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations, 2015.

[40] Perazzi, F.; Pont-Tuset, J.; McWilliams, B.; van Gool, L.; Gross, M.; Sorkine-Hornung, A. A benchmark dataset and evaluation methodology for video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 724–732, 2016.

[41] Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* Vol. 13, No. 4, 600–612, 2004.

[42] Baker, S.; Scharstein, D.; Lewis, J. P.; Roth, S.; Black, M. J.; Szeliski, R. A database and evaluation methodology for optical flow. *International Journal Computer Vision* Vol. 92, No. 1, 1–31, 2011.

Jinbo Xing received his B.Sc. degree in computer science from the Chinese University of Hong Kong in 2020. He is currently an M.Sc. student in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. His research interests include computer vision and computer graphics.

Wenbo Hu is currently a Ph.D. student in the Department of Computer Science and Engineering, the Chinese University of Hong Kong. He received his B.Sc. degree in computer science and technology from Dalian University of Technology, China, in 2018. His research interests include computer vision, computer graphics, and deep learning.

Yuechen Zhang is currently a final-year undergraduate student majoring in computer science at the Chinese University of Hong Kong. His research interests include semantic segmentation, video frame interpolation, and neural style transfer.

Tien-Tsin Wong received his B.Sc., M.Phil., and Ph.D. degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively, where he is currently a professor in the Department of Computer Science and Engineering. His main research interests include computer graphics, computational manga, precomputed lighting, image based rendering, GPU techniques, medical visualization, multimedia compression, and computer vision. He received an *IEEE Transactions on Multimedia* Prize Paper Award 2005 and a Young Researcher Award 2004.