Video Frame Interpolation via Deformable Separable Convolution

Xianhang Cheng,[1] **Zhenzhong Chen**[1]*
[1]School of Remote Sensing and Information Engineering, Wuhan University, China
{xianhang, zzchen}@whu.edu.cn

Abstract

Learning to synthesize non-existing frames from the original consecutive video frames is a challenging task. Recent kernel-based interpolation methods predict pixels with a single convolution process to replace the dependency of optical flow. However, when scene motion is larger than the predefined kernel size, these methods yield poor results even though they take thousands of neighboring pixels into account. To solve this problem in this paper, we propose to use deformable separable convolution (**DSepConv**) to adaptively estimate kernels, offsets and masks to allow the network to obtain information with much fewer but more relevant pixels. In addition, we show that the kernel-based methods and conventional flow-based methods are specific instances of the proposed DSepConv. Experimental results demonstrate that our method significantly outperforms the other kernel-based interpolation methods and shows strong performance on par or even better than the state-of-the-art algorithms both qualitatively and quantitatively.

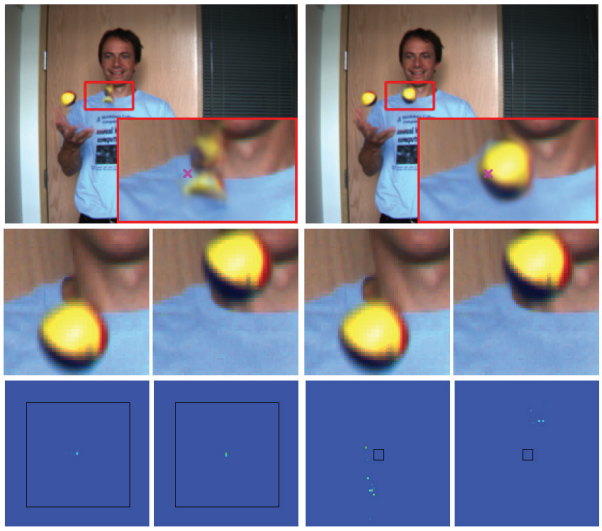
Introduction

Video frame interpolation is the task of synthesizing middle non-existent frames from two consecutive frames, which is considered as one of the important problems in the field of video processing. The ability to generate in-between frames could find applications in various domains, ranging from novel view interpolation (Flynn et al. 2016), slow motion generation (Jiang et al. 2018; Bao et al. 2019) to frame rate conversion (Bao et al. 2018b).

Traditional methods interpolate one frame with a two-step method which first estimate motion information, typically optical flow, and then perform the pixel synthesis operation guided by motion (Baker et al. 2011). However, a well-known problem behind this method is that visual artifacts can be introduced in challenging conditions (*e.g.* occlusion, illumination or nonlinear structural changes), suggesting that the two-step interpolation process struggles to reconstruct plausible results.

To better handle occlusion, recent approaches address the problem aforementioned with more elaborated pipelines by estimating flow information together with occlusion masks or visibility maps with deep convolutional neural networks (CNNs) (Jiang et al. 2018; Bao et al. 2019; 2018a; Liu et al. 2017; van Amersfoort et al. 2017; Liu et al. 2019; Xue et al. 2019; Peleg et al. 2019; Yuan et al. 2019; Hannemose et al. 2019).

Instead of using optical flow, another major trend in this research is to replace the two-step interpolation operation as a convolution process (Niklaus, Mai, and Liu 2017a; 2017b). For each output pixel, a pair of 2D kernels or four 1D kernels (two for horizontal and another two for vertical direction) are learned with a neural network. Notably, large



(a) SepConv-L_1 (b) Ours

Figure 1: An interpolated example. The third row provides magnified views of the effective sampling locations centered at the pink point in the first row. Greener color represents higher weights. The second row shows corresponding convolution patches. Compared with Sepconv (Niklaus, Mai, and Liu 2017b), our method considers more relevant pixels far away from local grid (black rectangle) with a much smaller kernel size and performs better.

*Corresponding Author

kernel size is required for these kernel-based interpolation methods to handle large motion. Though these methods are able to generate reasonable results, there are two main drawbacks: 1) These methods are limited to handle motion larger than kernel size. 2) It is expensive to consider thousands of pixels to synthesize only one output pixel.

In this paper, we address these drawbacks by presenting a more powerful and effective approach, known as *Deformable Separable Convolution* (DSepConv). We argue that the limitation of the previous kernel-based interpolation methods is because they process the pixels only in the *local* neighborhood, which takes no effect on pixels outside the regular grid. As shown in Figure 1, with large motion, the estimated kernels are incorrect despite they have considered 51×51 pixels in the local neighborhood. Drawing inspiration from the success of deformable convolution networks (Dai et al. 2017; Zhu et al. 2019), we propose to learn adaptive kernels, offsets and masks for interpolation, allowing us to use far fewer but more effective pixels to deal with large motion. Moreover, we show that conventional flow-based interpolation methods with CNNs are specific instances of our method. Our experiments show that the proposed method can greatly increase the performance of existing kernel-based methods and perform favorably against representative state-of-the-art interpolation methods.

The contributions of this paper are summarized as follows:

• A novel solution for frame interpolation DSepConv is proposed which learns not only spatially-adaptive separable convolution kernels, but also deformable offsets and masks. This approach is able to use small kernel size to handle strong motion.

• Both recent kernel-based and flow-based interpolation methods are demonstrated as special cases of our proposed DSepConv.

• Despite any complex information (like context, depth, flow and edge information) or post-processing process are not involved in our network, the design of jointly estimating kernels, offsets and masks makes our performance on par or even better than the state-of-the-art methods.

Related Work

Various methods to synthesize intermediate video frames have been introduced. In this section, we provide an overview of recent interpolation methods in the following three parts.

Flow Based Methods

Using optical flow is one of the most conventional strategies for video frame interpolation. Considering the reference frames are not equally informative due to occlusion, mask maps are often estimated together with optical flow for adaptively blending the warped frames. Specifically, Liu et al. (Liu et al. 2017) proposed a fully-convolutional encoder-decoder architecture named Deep Voxel Flow (DVF) to estimate 3D flow across space and time. The in-between frame was then warped by trilinear sampling. Similar method has been introduced by Jiang et al. (Jiang et al. 2018), who used two U-Net architectures to compute bi-directional optical flow and visibility maps. Moreover, CyclicGen (Liu et al. 2019) additionally used edge information (Xie and Tu 2015) and cycle consistency loss, which greatly improved the performance of DVF. In order to get more accurate optical flow, some methods leveraged advanced flow or depth estimation architectures as sub-modules in their networks. For instance, ToFlow (Xue et al. 2019) utilized SPyNet (Ranjan and Black 2017) to get flow information for interpolation. MEMC-Net* (Bao et al. 2018a) chose FlowNetS (Dosovitskiy et al. 2015) for motion estimation. DAIN (Bao et al. 2019) used PWC-Net (Sun et al. 2018) and a depth network (Chen et al. 2016) to explicitly detect the occlusion. Other interpolation methods such as CtxSyn (Niklaus and Liu 2018) and Zoom-In-to-Check (Yuan et al. 2019) warped not only input frames, but also their deep corresponding features. Despite the so far mentioned approaches have been shown effective, the performance can be limited provided that the optical flow or occlusion masks were less accurate.

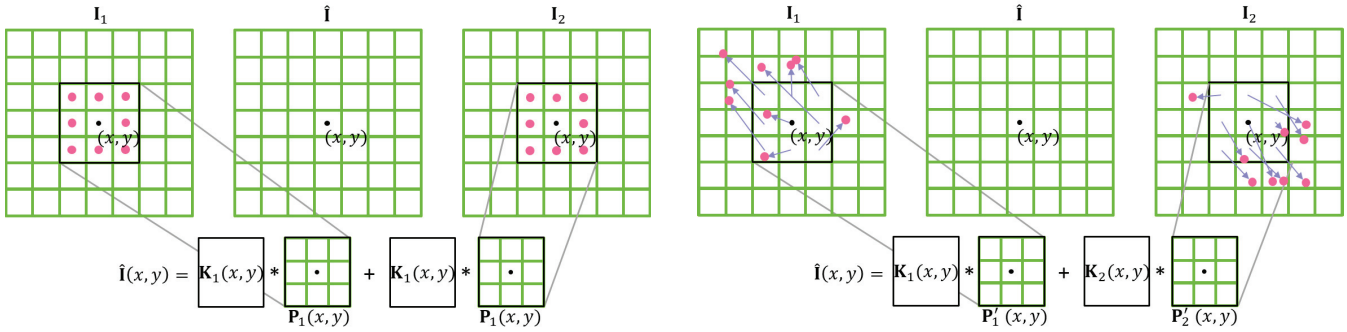
Phase Based Methods

Several methods utilize phase information to learn the motion relationship for video frame interpolation. Meyer et al. (Meyer et al. 2015) proposed the phase-based method which utilized phase information across the levels of a multi-scale pyramid. Such an approach performs well when the motion is small while fails in difficult interpolation settings. To further increase the performance, combined with CNNs, PhaseNet (Meyer et al. 2018) was proposed to better handle challenging scenarios like brightness changes and motion blur. However, their performance can be less effective in coping with the level of detail.

Kernel Based Methods

Kernel based methods regard flow estimation as an intermediate step, which can be circumvented with a single convolution process. These methods have also been exploited in some other tasks like frame prediction (Reda et al. 2018) and motion blur synthesis (Brooks and Barron 2019). As a pioneer of kernel based methods, AdaConv (Niklaus, Mai, and Liu 2017a) was proposed to estimate a pair of spatially-adaptive convolutional kernels for each output pixel with a neural network. To reduce the large memory demand, Niklaus et al. (Niklaus, Mai, and Liu 2017b) proposed SepConv that separated each 2D convolution kernel into a vertical and a horizontal kernels. SepConv increased the performance to some extent, but it failed to handle motion larger than 51 (pixels) and yielded poor results.

Some recent efforts (Choi and Bajic 2019; Deng et al. 2019; Ahn, Jeong, and Kim 2019; Peleg et al. 2019) have been made to mitigate the limitations of SepConv (Niklaus, Mai, and Liu 2017b). However, they all process the information in a *local* neighborhood, making them have to use large kernels (51×51) or operate on down-sampled frames (half, one quarter or one eighth of the original frame size) to handle potential large motion. On the contrary, we propose to use far smaller separable kernels (5×1) with offsets and masks to adaptively convolve with input images in a *non-local* neighborhood. Concurrently, similar work like



(a) Baseline kernel-based methods.

(b) Our proposed.

Figure 2: Comparisons of the sampling locations in 3×3 convolutions between the baseline methods (Niklaus, Mai, and Liu 2017a; 2017b) and our method. Our method can obtain pixels (pink points) outside the local neighborhood with additional learnable offsets (purple arrows), allowing us to better handle large motion. Modulation scalars are omitted for clarity.

ADC (Lee et al. 2019), has been propose to find the spatial transform between the frames. However, different from their method, our proposed DSepConv estimates separable 1D kernels rather than learning 2D kernels. In addition, we learn scalar masks as a modulation mechanism for each shift pixel of the two frames, which guarantees the diversity of the synthesized pixels and leads to a better performance. Moreover, we prove that when the kernel size is set to be 1, our method can be seen as a flow based method.

The Proposed Approach

In this section, we introduce our proposed deformable separable convolution method for video frame interpolation, including the details of our network architecture and our training details.

Adaptive Deformable Separable Convolution

Let us assume $\mathbf{I}_1$ and $\mathbf{I}_2$ to represent the two input frames, $\hat{\mathbf{I}}$ denotes the frame to be interpolated temporally in the midpoint of the two frames. For each pixel $\hat{\mathbf{I}}(x, y)$ to be synthesized, a pair of convolution kernels are estimated to adaptively convolve the local patches $\mathbf{P}_1(x, y)$ and $\mathbf{P}_2(x, y)$ centered at (x, y) from $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively. This process can be formulated as

$$\hat{\mathbf{I}}(x, y) = \mathbf{K}_1(x, y) * \mathbf{P}_1(x, y) + \mathbf{K}_2(x, y) * \mathbf{P}_2(x, y), \quad (1)$$

where $\mathbf{K}_1$ and $\mathbf{K}_2 \in \mathrm{R}^{n \times n}$ represent $n \times n$ 2D convolution kernels. Figure 2(a) illustrates the approach. For standard local convolution, the kernel size has to be large enough to capture large motion. In AdaConv (Niklaus, Mai, and Liu 2017a), the kernel size n is set to be 41. However, estimating such an amazing number of kernels (41×41) simultaneously entails heavy computational load. Instead of estimating the entire 2D kernels, SepConv (Niklaus, Mai, and Liu 2017b) approximates each 2D kernel with two 1D kernels $\langle \mathbf{k}_{1,v}, \mathbf{k}_{1,h} \rangle$ or $\langle \mathbf{k}_{2,v}, \mathbf{k}_{2,h} \rangle$ with formulation:

$$\begin{cases} \mathbf{K}_1(x, y) = \mathbf{k}_{1,v}(x, y) \cdot \mathbf{k}_{1,h}^{\top}(x, y), \\ \mathbf{K}_2(x, y) = \mathbf{k}_{2,v}(x, y) \cdot \mathbf{k}_{2,h}^{\top}(x, y), \end{cases} \quad (2)$$

reducing the number of kernel parameters from n^2 to $2n$ for each kernel. The modification enables the kernel size to be larger (51×51) than AdaConv as a result of the reduction of computational burden. Nonetheless, despite thousands of pixels have been considered, these methods are limited to motions up to n pixels between two input frames.

Inspired from recent deformable convolution networks (Dai et al. 2017; Zhu et al. 2019), we propose to use much smaller convolution kernels with additional offsets and masks, which allow us to focus on fewer but more relevant pixels rather than all the pixels in a large neighborhood (shown in Figure 2(b)). Given a convolution kernel and its corresponding local patch with specific kernel size n ($n = 5$ in our method), let $\mathbf{p}_{i,j}$ denote the pre-specified offset for the j-th ($j \in [0, n^2)$) location in a specific patch and i represents either of the two input frames. Thus each pixel in patch $\mathbf{P}_i(x, y)$ centered at (x, y) in frame $\mathbf{I}_i$ can be represented as $\mathbf{P}_i(x, y; \mathbf{p}_j)$. In our method, learnable offset $\Delta\mathbf{p}_{i,j}$ and modulation scalar $\Delta\mathbf{m}_{i,j}$ are estimated for each pixel located at $\mathbf{p}_{i,j}$ in each patch. As a result, the pixels in the modulated patches can be expressed as

$$\begin{cases} \mathbf{P}_1^{'}(x, y; \mathbf{p}_{1,j}) = \mathbf{P}_1(x, y; \mathbf{p}_{1,j} + \Delta\mathbf{p}_{1,j}) \cdot \Delta\mathbf{m}_{1,j}, \\ \mathbf{P}_2^{'}(x, y; \mathbf{p}_{2,j}) = \mathbf{P}_2(x, y; \mathbf{p}_{2,j} + \Delta\mathbf{p}_{2,j}) \cdot \Delta\mathbf{m}_{2,j}. \end{cases} \quad (3)$$

As the offsets are typically fractional, pixels located at non-integral coordinates are bilinearly sampled. In addition to the modulated patches, we estimate 1D separable kernels to approximate 2D kernels in Eq. (2). Therefore, our final interpolation process is expressed as

$$\begin{aligned} \hat{\mathbf{I}}(x, y) = \mathbf{k}_{1,v}(x, y) \cdot \mathbf{k}_{1,h}^{\top}(x, y) * \mathbf{P}_1^{'}(x, y) \\ + \mathbf{k}_{2,v}(x, y) \cdot \mathbf{k}_{2,h}^{\top}(x, y) * \mathbf{P}_2^{'}(x, y). \end{aligned} \quad (4)$$

In our method, both previous kernel-based methods (Niklaus, Mai, and Liu 2017b) and conventional flow-based methods are specific instances. In Eq. (3), it is easy to make out that when $\Delta\mathbf{p} = \mathbf{0}$ and $\Delta\mathbf{m} = \mathbf{1}$, the interpolation process is the same as the one in SepConv (Niklaus, Mai, and

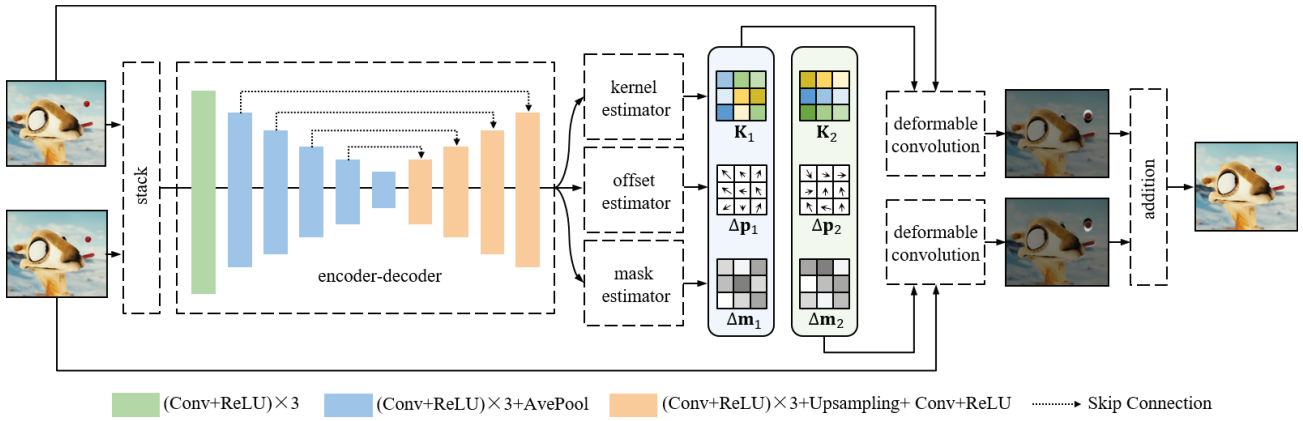


Figure 3: Illustration of the architecture of our proposed DSepConv network. The encoder-decoder architecture extracts features, which are given to three sub-modules to estimate separable kernels, offsets and masks for each output pixel. The input frames are then convolved adaptively with deformable convolution described in Eq. (3) and (4).

Liu 2017b). On the other hand, specifically, when $n = 1$, the patches become single pixels via bilinear interpolation. Therefore, the vectors in Eq. (4) become scalars. In this case, the interpolation process can be reformulated as

$$\begin{aligned}&\hat{\mathbf{I}}(x,y) = \\ &k_{1,v}(x,y) \cdot k_{1,h}(x,y) \cdot \mathbf{I}_1(x+\Delta x_1, y+\Delta y_1) \cdot \Delta m_1 + \\ &k_{2,v}(x,y) \cdot k_{2,h}(x,y) \cdot \mathbf{I}_2(x+\Delta x_2, y+\Delta y_2) \cdot \Delta m_2,\end{aligned} \tag{5}$$

where $\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2$ represent learnable offsets and $\Delta m_1, \Delta m_2$ denote masks. Notably, Eq. (5) can be seen as a bi-directional warping function, where each offset can be regarded as a component of optical flow and $k_{1,v}(x,y) \cdot k_{1,h}(x,y) \cdot \Delta m_1$ as well as $k_{2,v}(x,y) \cdot k_{2,h}(x,y) \cdot \Delta m_2$ can be considered as occlusion masks.

Network Architecture

We use a fully convolutional neural network which is similar to SepConv (Niklaus, Mai, and Liu 2017b). The whole network can be divided into the following submodules: the encoder-decoder architecture, kernel estimator, offset estimator and mask estimator as illustrated in Figure 3. The detailed configurations are provided in Section 2 of the Supplementary Material.

Encoder-decoder Architecture Given two input frames, the encoder-decoder architecture aims to extract deep features for estimating kernels, masks and offsets for each output pixel. We use a U-Net architecture which is the same as the one in (Niklaus, Mai, and Liu 2017b), where skip-connections are employed to facilitate the feature mixture across encoder and decoder.

Kernel Estimator The kernel estimator consists of four parallel sub-networks with analogous structure to estimate vertical and horizontal 1D kernels for each pixel of the two frames. For each sub-network, three 3×3 convolution layers with Rectified Linear Units (ReLU) (Nair and Hinton 2010), a bilinear upsampling layer and another 3×3 convolution layer are stacked, yielding a tensor with n channels. Subsequently, the estimated four 1D kernels are used to approximate two 2D kernels described in Eq. (2).

Offset Estimator The offset estimator, sharing the same structure as the kernel one described above, contains four parallel sub-networks to learn two directional (vertical and horizontal) offsets for each location of the two frame patches. With a specific kernel size n, there are n^2 pixels in each regular grid patch. Therefore, the number of the output channel in each sub-network is set to be n^2.

Mask Estimator Inspired by (Zhu et al. 2019), learnable masks $\Delta\mathbf{m}$ are introduced as a modulation mechanism that expands the scope of modeling and gives a significant improvement in performance. The design of mask estimator is similar, whose only difference is that the output channels are fed to a sigmoid layer. There are two parallel sub-networks, each of which produces tensors with n^2 channels.

Deformable Convolution The deformable convolution utilizes the estimated kernels, offsets and mask to adaptively convolve one input frame, yielding an intermediate interpolation result. Note that this operation is adaptive and represents Eq. (3) and (4) in this paper, which does not totally resemble the process described in (Zhu et al. 2019). Finally, the target frame is generated by adding the two intermediate results. In the right part of Figure 3, the intermediate results generated from deformable convolution look dimmer than the final result in brightness except area with occlusion (*e.g.* area around the red ball), suggesting the effectiveness of deformable convolution to handle motion and occlusion.

Training

Loss Functions We combine two kinds of loss functions to penalize the interpolated frame $\hat{\mathbf{I}}$ that is not similar to the ground truth $\mathbf{I}^{GT}$. In addition, to encourage the network to be invariant to the temporal order of the input frames, a temporal symmetry term is added in each loss function (Peleg et al. 2019). Here we assume $\hat{\mathbf{I}}'$ to represent the result generated by the temporal flipping of the input frames.

The first loss measures the difference between the interpolated pixel color and the ground-truth color with the function:

$$\mathcal{L}_c = \rho(\hat{\mathbf{I}} - \mathbf{I}^{GT}) + \rho(\hat{\mathbf{I}}' - \mathbf{I}^{GT}), \tag{6}$$

$$\rho(x) = \sqrt{x^2 + \epsilon^2}, \tag{7}$$

where $\rho(\cdot)$ represents the Charbonnier penalty function (Charbonnier et al. 1994) and the constant ϵ is set to be $1e$-6.

The second loss function aims to sharpen the generated frame by penalizing the differences of frame gradient predictions (Mathieu, Couprie, and LeCun 2015), which is defined as:

$$\begin{aligned}\mathcal{L}_{gdl} = \sum_{i,j} & \big\| |\hat{\mathbf{I}}_{i,j} - \hat{\mathbf{I}}_{i-1,j}| - |\mathbf{I}^{GT}_{i,j} - \mathbf{I}^{GT}_{i-1,j}| \big\|_1 \\ & + \big\| |\hat{\mathbf{I}}_{i,j} - \hat{\mathbf{I}}_{i,j-1}| - |\mathbf{I}^{GT}_{i,j} - \mathbf{I}^{GT}_{i,j-1}| \big\|_1 \\ & + \big\| |\hat{\mathbf{I}}'_{i,j} - \hat{\mathbf{I}}'_{i-1,j}| - |\mathbf{I}^{GT}_{i,j} - \mathbf{I}^{GT}_{i-1,j}| \big\|_1 \\ & + \big\| |\hat{\mathbf{I}}'_{i,j} - \hat{\mathbf{I}}'_{i,j-1}| - |\mathbf{I}^{GT}_{i,j} - \mathbf{I}^{GT}_{i,j-1}| \big\|_1 .\end{aligned} \tag{8}$$

Finally, the total loss function is given as:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_{gdl}. \tag{9}$$

Training Strategy Our training dataset is Vimeo90K (Xue et al. 2019). The triplets in this dataset were randomly flipped horizontally or vertically for data augmentation.

The network was trained using Adam optimizer (Kingma and Ba 2014). We first trained our network for 120 epochs using a learning rate schedule of $1e$-4, dropping by half every 40 epochs. The training patch size was randomly cropped into 128×128 pixels and the batch size was 16. Inspired by some previous work training their networks with larger patches (Niklaus and Liu 2018; Bao et al. 2018a; 2019), we fine-tuned our network using patches of size 256×256 and the entire frames with learning rates of $6.25e$-6 and $3.125e$-6, respectively.

Experiments

Evaluation Datasets and Metrics

We test our network on several datasets with different resolutions, including UCF101 (Soomro, Zamir, and Shah 2012), Vimeo90K (Xue et al. 2019) and Middleburry dataset (Baker et al. 2011). For UCF101 dataset, we use 379 triplets chosen by Liu et al. (Liu et al. 2017) with a resolution of 256×256 pixels. The Vimeo90K dataset contains 3,782 triplets in the test set with a resolution of 256×448 pixels. The Middleburry benchmark has been widely used for assessing frame interpolation methods, which consists an *Evaluation* set (hidden ground truth) and an *Other* set (with ground truth).

For quantitative evaluation, we use Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) on UCF101 and Vimeo90K datasets. In addition, we report the average Inerpolation Error (IE) on the Middleburry dataset.

Ablation Study

In this section, we perform comprehensive ablations to analysis our network structure, including different size of the estimated kernels and the usage of the mask estimator.

Methods	UCF101		Vimeo90K		M.B.
	SSIM	PSNR	SSIM	PSNR	IE
1×1 + M	0.9666	34.45	0.9659	33.63	2.57
3×3 + M	<u>0.9680</u>	34.79	<u>0.9731</u>	34.52	2.21
5×5	<u>0.9680</u>	<u>34.98</u>	0.9728	<u>34.55</u>	<u>2.11</u>
5×5 + M	**0.9686**	**35.08**	**0.9738**	**34.73**	**2.06**

Table 1: Quantitative evaluation on different network architecture: kernel size of $N \times N$ with ($N \times N$ + M) or without masks ($N \times N$). M.B. is short for the *Other* set of Middleburry dataset. The bold numbers and underlined numbers depict the best and the second best performances.

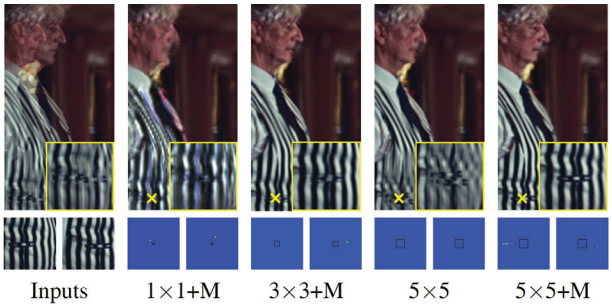


Figure 4: The effect of different network architectures.

Kernel Size For each pixel to be synthesized, the kernel size n indicates how many pixels in the non-regular grid augmented with offsets could be referenced. Larger kernel size enables the network take more pixels into consideration. However, it inevitably introduces an increase computation.

To understand the effect of different numbers of reference pixels in each pixel synthesis, we trained several models that generate kernels with different size ($n = 1, 3, 5$, respectively) and show the quantitative results in Table 1. Larger kernel sizes like $n = 7, 9, 11$ are not considered as they increase the FLOPs of the network when $n = 5$ by 12.8%, 69.0% and 173.8%, respectively. We can observe that referencing more pixels can lead to a better performance. When increasing the kernel size from 1×1 to 5×5, the network has a PSNR gain of 0.65 dB and 1.04 dB on UCF101 and Vimeo90K, respectively. A visualization of the referenced pixels with different kernel size is shown in the first row of Figure 4. And we show their representative kernels in the second row. By referencing more relevant pixels in color, the model that uses larger kernel size can generate shaper and clearer interpolation result.

Mask Estimator To examine the effectiveness of the mask estimator in our network, we trained a network without estimating masks. As shown in Table 1 and Figure 4, network with mask estimator gives a significant improvement in performance. This can be attributed to the capability of the modulation mechanism which adjusts offsets in perceiving input patches (Zhu et al. 2019).

Methods	Sub-networks						UCF101		Vimeo90K		M.B.	Parameters (million)
	flow	kernel	mask	context	other	post-proc.	SSIM	PSNR	SSIM	PSNR	IE	
DVF	Enc-Dec	bilinear(2)	√	×	×	×	0.9631	34.12	0.9462	31.54	—	1.60
ToFlow	SPyNet	bilinear(2)	√	×	×	√	0.9667	34.58	0.9682	33.73	2.51	1.07
AdaConv	×	learned(41)	×	×	×	×	—	—	0.9568	32.33	—	—
SepConv-L_f	×	learned(51)	×	×	×	×	0.9655	34.69	0.9674	33.45	2.44	21.6
SepConv-L_1	×	learned(51)	×	×	×	×	0.9669	34.78	0.9702	33.79	2.27	21.6
CtxSyn	PWC-Net	bilinear(2)	×	ResNet	×	×	—	34.62	—	—	—	—
CyclicGen	Enc-Dec	bilinear(2)	√	×	HED	×	<u>0.9684</u>	**35.11**	0.9490	32.09	2.86	3.04
CyclicGen_large	Enc-Dec	bilinear(2)	√	×	HED	×	0.9658	34.69	0.9395	31.46	3.04	19.8
MEMC-Net*	FlowNetS	learned(4)	√	ResNet	×	√	0.9683	35.01	<u>0.9742</u>	34.40	2.10	70.3
IM-Net	×	learned(25)	√	Enc-Dec	×	√	—	—	—	33.50	—	—
DAIN	PWC-Net	learned(4)	×	Enc-Dec	Megadepth	√	0.9683	34.99	**0.9756**	<u>34.71</u>	**2.04**	24.0
DSepConv	×	learned(5)	√	×	×	×	**0.9686**	<u>35.08</u>	0.9738	**34.73**	<u>2.06</u>	21.8

Table 2: Quantitative comparisons and analysis on different frame interpolation algorithms using CNNs. The bold numbers and underlined numbers depict the best and the second best performances. Our method is comparable even without using any complex information.

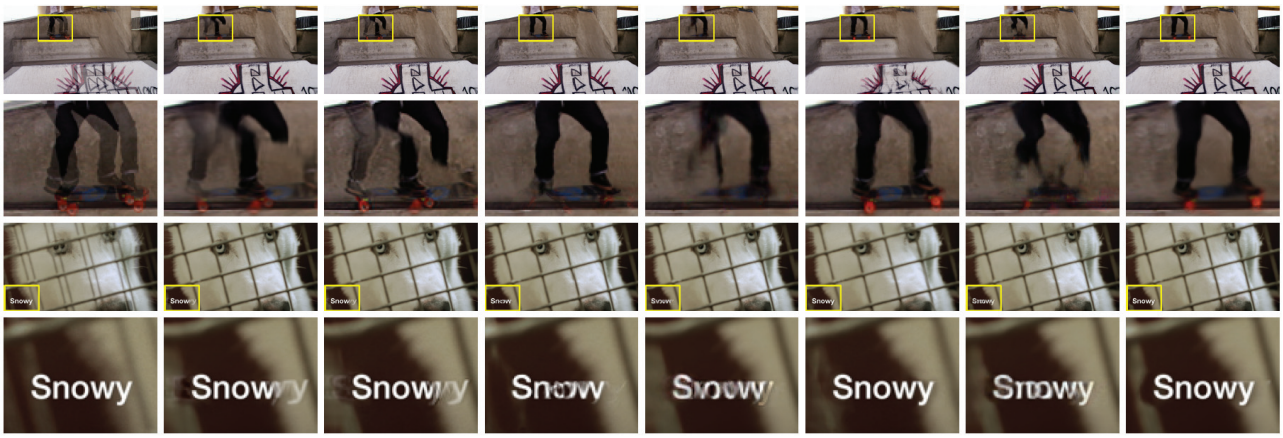


Figure 5: Visual comparisons on the Vimeo90K dataset. Our method reconstructs the legs (top) and the letter (bottom) well.

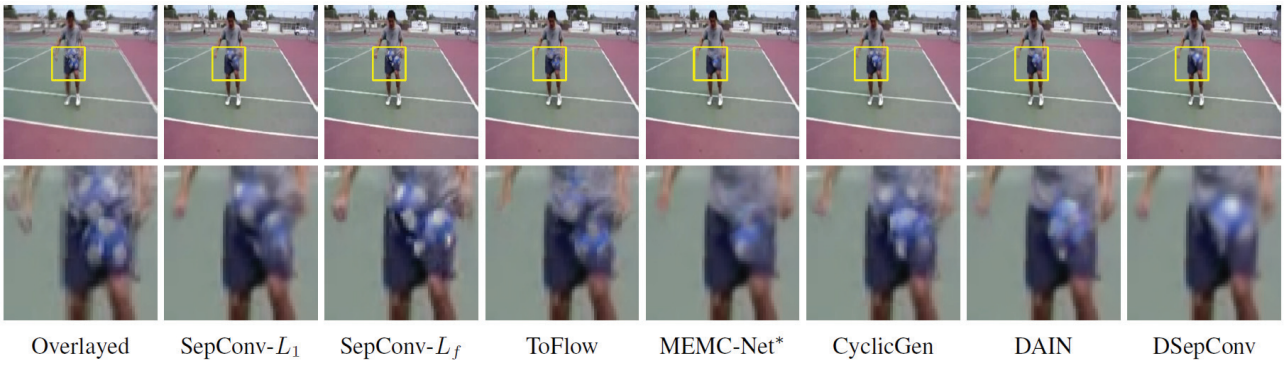


Figure 6: Visual comparisons on the UCF101 dataset. Our method reconstructs the football with a clearer shape.

Comparisons with State-of-the-arts

We compare our method with state-of-the-art interpolation methods, including DVF (Liu et al. 2017), ToFlow (Xue et al. 2019), AdaConv (Niklaus, Mai, and Liu 2017a), SepConv (Niklaus, Mai, and Liu 2017b), CtxSyn (Niklaus and Liu 2018), CyclicGen (Liu et al. 2019), MEMC-Net* (Bao et al. 2018a), IM-Net (Peleg et al. 2019), DAIN (Bao et al. 2019) and ADC (Lee et al. 2019). Notably, considering that some methods provide more than one version of the same model, we evaluate all their performances and treat them differently (*e.g.* SepConv-L_1 and SepConv-L_f are trained with different loss functions, CyclicGen and CyclicGen_large are de-

Methods	Mequon			Schefflera			Urban			Teddy			Backyard			Basketball			Dumptruck			Evergreen			AVERAGE		
	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.	all	disc.	unt.
AdaConv	3.57	6.88	1.41	4.34	5.67	2.52	5.00	5.86	2.98	6.91	8.89	4.89	10.2	12.8	3.21	5.33	10.1	2.27	7.30	16.6	1.92	6.94	10.8	1.67	6.20	9.70	2.61
SepConv-L_1	2.52	4.83	1.11	3.56	5.04	1.90	4.17	4.15	2.86	5.41	6.81	3.88	10.2	12.8	3.37	5.47	10.4	2.21	6.88	15.6	1.72	6.63	10.3	1.62	5.61	8.74	2.33
ToFlow	2.54	4.35	1.16	3.70	5.19	1.88	3.43	3.89	1.93	5.05	6.43	3.39	9.84	12.3	3.42	5.34	10.0	2.28	6.88	15.2	1.61	7.14	11.0	1.69	5.49	8.55	2.17
CtxSyn	**2.24**	**3.72**	**1.04**	**2.96**	**4.16**	**1.35**	4.32	**3.42**	3.18	**4.21**	**5.46**	**3.00**	9.59	11.9	3.46	5.22	9.76	2.22	7.02	15.4	1.58	6.66	10.2	1.69	5.28	8.00	2.19
MEMC-Net*	2.39	3.92	1.28	3.36	4.52	2.07	3.37	3.86	2.20	4.84	5.93	3.72	8.55	10.6	3.14	4.70	8.81	2.03	6.40	14.2	1.58	6.37	9.87	1.57	5.00	7.71	2.20
DAIN	2.38	4.05	1.26	3.28	4.53	1.79	3.32	3.77	2.05	4.65	5.88	3.41	7.88	9.74	**3.04**	4.73	8.90	2.04	6.36	14.3	1.51	**6.25**	**9.68**	1.54	**4.86**	7.61	2.08
ADC	2.54	4.31	1.29	3.27	4.46	1.62	3.76	3.76	1.70	5.27	6.37	3.19	8.66	10.8	3.11	4.78	9.04	**2.01**	5.72	12.8	**1.41**	6.56	10.2	**1.51**	5.07	7.72	**1.98**
DSepConv	2.47	4.39	1.21	3.32	4.60	1.72	**3.28**	3.66	**1.50**	5.11	6.36	3.23	**7.85**	**9.69**	3.11	**4.68**	**8.78**	2.04	**5.65**	**12.5**	1.44	6.54	10.2	1.58	**4.86**	**7.52**	**1.98**

Table 3: Quantitative comparisons (interpolation error) on the Middlebury *Evaluation* dataset. *disc.*: regions with discontinuous motion. *unt.*: textureless regions. The bold numbers and underlined numbers depict the best and the second best performances.

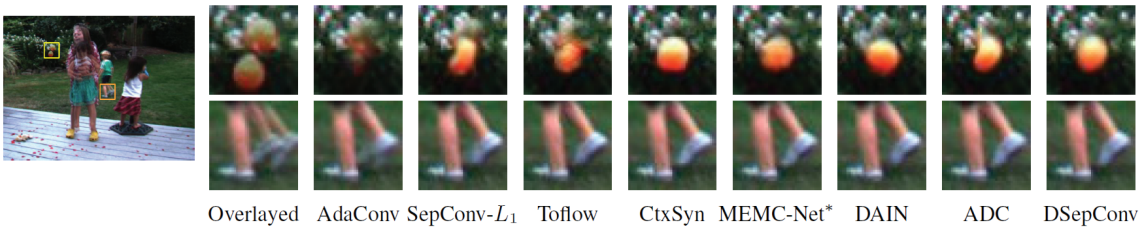


Figure 7: Visual comparisons on the Middleburry dataset. Our method maintains the structures of both the ball and feet well.

signed with different receptive fields.).

Sub-network Comparisons We analyse different sub-networks contributed to the final interpolation results with following components: flow, kernel, mask, context estimation networks as well as post-processing networks (abbreviated by post-proc.). Specially, networks for learning information that falls outside the mentioned five modules will be categorized as "other" class shown in the left of Table 2. A self-defined encoder-decoder structure is abbreviated by Enc-Dec. Sub-networks with pre-trained models are filled with their names. In addition, the bilinear interpolation process can be seen as using bilinear kernels with size of 2×2, while spatially-adaptive kernels learning from CNNs have their own specific kernel size.

Quantitative and Visual Comparisons We perform quantitative comparisons on UCF101, Vimeo90K and the Middlebury *Other* set (abbreviated by M.B.) in the right of Table 2. Sharing similar network structure and parameters, our method outperforms our baseline SepConv-L_1 by more than 0.3dB and 0.9dB (PSNR) on UCF101 and Vimeo90K datasets, respectively. Moreover, without relying on any extra complex information such as flow, context, edge or depth information, our method shows strong performance on par or even better than the other state-of-the-art methods.

We further show some visual comparisons with state-of-the-art methods that are publicly available. The top row in Figure 5 is an example where there are strong motions between the input frames.The SepConv-L_1 and SepConv-L_f can not well reconstruct the legs of the man due to a wrong estimation of local convolution kernels. The MEMC-Net* and DAIN methods generate obvious artifacts despite they use a couple of sub-modules in their networks. Toflow can produce clear result in the man's leg but some information is lost in the skateboard. In contrast, our method reconstructs them well. The bottom row of Figure 5 shows an example of regions under discontinuous motion. The motion is continuous between the frames except the subtitle "Snowy". Both CyclicGen and our method can generate clear results while the other methods can not handle the discontinuity well.

In Figure 6, we show an example from UCF101 dataset. Our method can restore clear details of the football while the results generated by the other interpolation methods suffer from either artifacts or blur. More comparisons are available in Section 3 of the Supplementary Material.

Online Server In Table 3, we show some comparisons on the *Evaluation* set of the Middleburry benchmark. The proposed method performs favorably on most sequences and achieves the best average performance against the compared approaches. Furthermore, at the time of our submission, our method ranks the 3rd best performance among over 160 algorithms listed on the benchmark website.

Figure 7 shows a visual comparison downloaded from the online server. Our method reconstructs the ball and the feet of the boy with a clear boundary while AdaConv, SepConv, ToFlow, MEMC-Net* and ADC suffer from some blur.

Discussions and Limitations

Compared with previous methods, the proposed DSepConv is not constrained by neither the kernel size nor the accuracy of optical flow. However, like other kernel-based methods, our approach can only generate a single in-between frame.

Conclusion

In this paper we propose deformable separable convolution for video frame interpolation. The key to make the method practical is that our method can adaptively process the information in a non-local neighborhood by learning offsets and masks besides separable kernels. Effectively this allows us to handle large motion even with a small size of convolution kernel. And as demonstrated, both baseline kernel-based methods and flow-based methods are special cases of

our method and we perform favorably against the state-of-the-arts on diverse datasets qualitatively and quantitatively.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under contract No. 61771348 and National Key R&D Program of China under contract No. 2017YFB1002202.

References

Ahn, H.-E.; Jeong, J.; and Kim, J. W. 2019. A fast 4k video frame interpolation using a hybrid task-based convolutional neural network. *Symmetry*.

Baker, S.; Scharstein, D.; Lewis, J.; Roth, S.; Black, M. J.; and Szeliski, R. 2011. A database and evaluation methodology for optical flow. *IJCV*.

Bao, W.; Lai, W.; Zhang, X.; Gao, Z.; and Yang, M. 2018a. MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *CoRR* abs/1810.08768.

Bao, W.; Zhang, X.; Chen, L.; Ding, L.; and Gao, Z. 2018b. High-order model and dynamic filtering for frame rate up-conversion. *TIP*.

Bao, W.; Lai, W.-S.; Ma, C.; Zhang, X.; Gao, Z.; and Yang, M.-H. 2019. Depth-aware video frame interpolation. In *CVPR*.

Brooks, T., and Barron, J. T. 2019. Learning to synthesize motion blur. In *CVPR*.

Charbonnier, P.; Blanc-Feraud, L.; Aubert, G.; and Barlaud, M. 1994. Two deterministic half-quadratic regularization algorithms for computed imaging. In *ICIP*.

Chen, W.; Fu, Z.; Yang, D.; and Deng, J. 2016. Single-image depth perception in the wild. In *NIPS*.

Choi, H., and Bajic, I. V. 2019. Deep frame prediction for video coding. *CoRR* abs/1901.00062.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *ICCV*.

Deng, J.; Yu, H.; Wang, Z.; Wang, X.; and Huang, T. 2019. Self-reproducing video frame interpolation. In *MIPR*.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; and Brox, T. 2015. Flownet: Learning optical flow with convolutional networks. In *ICCV*.

Flynn, J.; Neulander, I.; Philbin, J.; and Snavely, N. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*.

Hannemose, M.; Jensen, J. N.; Einarsson, G.; Wilm, J.; Dahl, A. B.; and Frisvad, J. R. 2019. Video frame interpolation via cyclic fine-tuning and asymmetric reverse flow. In *SCIA*.

Jiang, H.; Sun, D.; Jampani, V.; Yang, M.-H.; Learned-Miller, E.; and Kautz, J. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Lee, H.; Kim, T.; Chung, T.; Pak, D.; Ban, Y.; and Lee, S. 2019. Learning spatial transform for video frame interpolation. *CoRR* abs/1907.10244.

Liu, Z.; Yeh, R. A.; Tang, X.; Liu, Y.; and Agarwala, A. 2017. Video frame synthesis using deep voxel flow. In *ICCV*.

Liu, Y.-L.; Liao, Y.-T.; Lin, Y.-Y.; and Chuang, Y.-Y. 2019. Deep video frame interpolation using cyclic frame generation. In *AAAI*.

Mathieu, M.; Couprie, C.; and LeCun, Y. 2015. Deep multi-scale video prediction beyond mean square error. In *ICLR*.

Meyer, S.; Wang, O.; Zimmer, H.; Grosse, M.; and Sorkine-Hornung, A. 2015. Phase-based frame interpolation for video. In *CVPR*.

Meyer, S.; Djelouah, A.; McWilliams, B.; Sorkine-Hornung, A.; Gross, M.; and Schroers, C. 2018. Phasenet for video frame interpolation. In *CVPR*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Niklaus, S., and Liu, F. 2018. Context-aware synthesis for video frame interpolation. In *CVPR*.

Niklaus, S.; Mai, L.; and Liu, F. 2017a. Video frame interpolation via adaptive convolution. In *CVPR*.

Niklaus, S.; Mai, L.; and Liu, F. 2017b. Video frame interpolation via adaptive separable convolution. In *ICCV*.

Peleg, T.; Szekely, P.; Sabo, D.; and Sendik, O. 2019. IM-Net for high resolution video frame interpolation. In *CVPR*.

Ranjan, A., and Black, M. J. 2017. Optical flow estimation using a spatial pyramid network. In *CVPR*.

Reda, F. A.; Liu, G.; Shih, K. J.; Kirby, R.; Barker, J.; Tarjan, D.; Tao, A.; and Catanzaro, B. 2018. Sdc-net: Video prediction using spatially-displaced convolution. In *ECCV*.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*.

Sun, D.; Yang, X.; Liu, M.-Y.; and Kautz, J. 2018. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*.

van Amersfoort, J. R.; Shi, W.; Acosta, A.; Massa, F.; Totz, J.; Wang, Z.; and Caballero, J. 2017. Frame interpolation with multi-scale deep loss functions and generative adversarial networks. *CoRR* abs/1711.06045.

Xie, S., and Tu, Z. 2015. Holistically-nested edge detection. In *ICCV*.

Xue, T.; Chen, B.; Wu, J.; Wei, D.; and Freeman, W. T. 2019. Video enhancement with task-oriented flow. *IJCV*.

Yuan, L.; Chen, Y.; Liu, H.; Kong, T.; and Shi, J. 2019. Zoom-In-to-Check: Boosting video interpolation via instance-level discrimination. In *CVPR*.

Zhu, X.; Hu, H.; Lin, S.; and Dai, J. 2019. Deformable convnets v2: More deformable, better results. In *CVPR*.