Revisiting Adaptive Convolutions for Video Frame Interpolation

Simon Niklaus
Adobe Research

Long Mai
Adobe Research

Oliver Wang
Adobe Research

(a) overlayed input frames   (b) original SepConv [38]   (c) our improved SepConv++   (d) current state of the art [36]
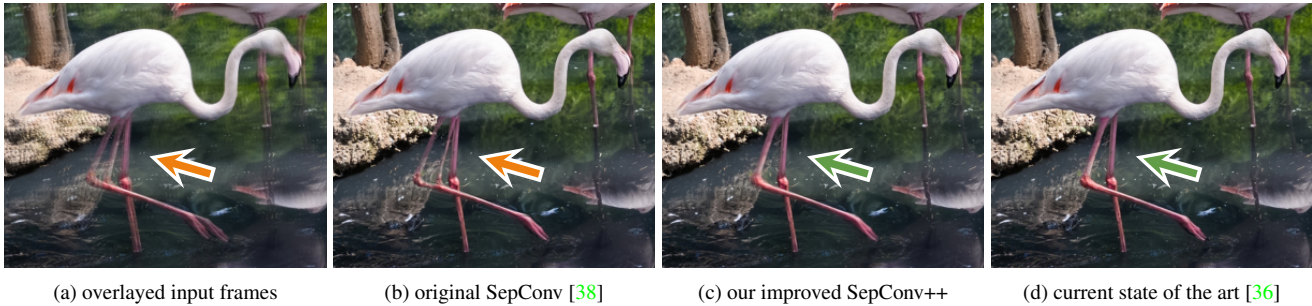
Figure 1: Frame interpolation example where the leg of the flamingo is difficult to handle. Our techniques to improve the original SepConv (b) enable us to synthesize results (c) that are comparable to current state-of-the-art approaches (d).

Abstract

Video frame interpolation, the synthesis of novel views in time, is an increasingly popular research direction with many new papers further advancing the state of the art. But as each new method comes with a host of variables that affect the interpolation quality, it can be hard to tell what is actually important for this task. In this work, we show, somewhat surprisingly, that it is possible to achieve near state-of-the-art results with an older, simpler approach, namely adaptive separable convolutions, by a subtle set of low level improvements. In doing so, we propose a number of intuitive but effective techniques to improve the frame interpolation quality, which also have the potential to other related applications of adaptive convolutions such as burst image denoising, joint image filtering, or video prediction.

1. Introduction

Video frame interpolation, the synthesis of intermediate frames between existing frames of a video, is an important technique with applications in frame-rate conversion [33], video editing [31], novel view interpolation [21], video compression [59], and motion blur synthesis [5]. While the performance of video frame interpolation approaches has seen steady improvements, research efforts have become increasingly complex. For example, DAIN [3] combines optical flow estimation [51], single image depth estimation [26], context-aware image synthesis [35], and adaptive convolu-

tions [37]. However, we show that somewhat surprisingly, it is possible to achieve near state-of-art results with an older, simpler approach by carefully optimizing its individual parts. Specifically, we revisit the idea of using adaptive separable convolutions [38] and augment it with a set of intuitive improvements. This optimized SepConv++ ranks second among all published methods in the Middlebury benchmark [1].

The reason for choosing adaptive separable convolutions to show that an older frame interpolation method can be optimized to produce near state-of-the-art results are threefold. First, kernel-based video frame interpolation jointly performs motion estimation and motion compensation in a single step which makes for an elegant image formation model [37] (see Figure 2 for more details on how this kernel-based interpolation differs from more-traditional flow-based interpolation). Second, adaptive separable convolutions are an efficient way to perform kernel-based interpolation [38]. In practice, filter kernels should be as large as possible to be able to account for large scene motion but this becomes prohibitively expensive with regular two-dimensional instead of two one-dimensional kernels. Third, adaptive convolutions have inspired and are part of many subsequent frame interpolation techniques [3, 4, 8, 9, 25]. As such, our findings on optimizing kernel-based video frame interpolation are directly applicable to the referenced approaches.

The idea of adaptive convolutions bears many names such as kernel prediction, dynamic filtering, basis prediction, or local attention. This technique has been proven effective in burst image denoising to align and merge multiple im-

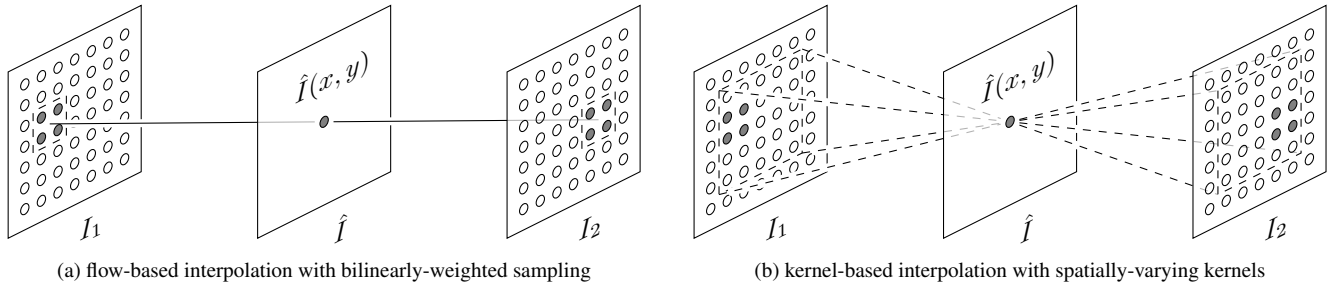| (a) flow-based interpolation with bilinearly-weighted sampling | (b) kernel-based interpolation with spatially-varying kernels |

Figure 2: Illustration of two prevalent video frame interpolation paradigms. Flow-based techniques first estimate the per-pixel motion between two frames and then compensate for it by warping the pixels according the estimated motion (a). This makes it possible to interpolate frames at an arbitrary time but one needs to account for inaccurate motion estimates and handle occlusions where optical flow is undefined. In comparison, frame interpolation via adaptive convolution jointly performs motion estimation and motion compensation in a single step by convolving input frames with spatially-varying kernels (b).

ages [30, 34, 60], in denoising Monte Carlo renderings by taking weighted averages of noisy neighborhoods [2, 14, 55], in the modelling of a broad class of image transformations [45], in optical flow upsampling and joint image filtering [22, 52], in video prediction where adaptive kernels can also model uncertainty [13, 17, 43, 64], in deblurring to model spatially-varying blur [48, 67], or super-resolution where they can be used to merge multiple observations with sub-pixel accuracy [6, 19]. While our paper focuses on improving adaptive separable convolutions for the purpose of frame interpolation, some of the improvements we introduce may be applicable in these related applications as well.

In summary, we revisit adaptive convolutions for frame interpolation and propose the following set of techniques that improve the method of SepConv [38] by a significant 1.76 dB on the Middlebury benchmark examples [1] with publicly known ground truth (the relative improvement of each individual technique is shown in parenthesis).

- delayed padding (+0.37 dB)
- input normalization (+0.30 dB)
- network improvements (+0.42 dB)
- kernel normalization (+0.52 dB)
- contextual training (+0.18 dB)
- self-ensembling (+0.18 dB)

These improvements allow our proposed SepConv++ to quantitatively outperform all other frame interpolation approaches with the exception of SoftSplat [36] even though many of these methods are much more sophisticated.

2. Related Work

With their work on adaptive convolutions for frame interpolation, Niklaus *et al.* [37] proposed to perform joint motion estimation and motion compensation based on predicting spatially-varying kernels. This idea led to several interesting new developments such as the usage of adaptive separable convolutions [38], adaptive warping layers

that combine optical flow estimates and adaptive convolutions [3, 4], additional per-coefficient offset vectors [8, 9], spatially-varying deformable convolutions [25, 47], or loss functions that leverage adaptive convolutions [42]. Many of these efforts introduce novel ideas that enable smaller kernel sizes while simultaneously being able to compensate for arbitrarily-large motion. In this paper, we go back to the roots of kernel-based frame interpolation and revisit the idea of adaptive separable convolutions [38], which strike a balance between simplicity and efficacy. By careful experimentation, we demonstrate several intuitive techniques that allow our proposed SepConv++ to achieve near state-of-the-art results despite being based on an older approach.

Aside from kernel-based interpolation, there are exciting research efforts that leverage explicit motion estimation in the form of optical flow for video frame interpolation. These efforts include estimating optical flow from the perspective of the frame that is ought to be synthesized [29], not only warping the original frames but also their feature representations such that a synthesis network can predict better results from the additional context [35], reconstructing optical flow representations from the perspective of the frame that is ought to be synthesized from a given inter-frame optical flow [18], fine-tuning the optical flow estimate for the given task at hand [63], softmax splatting for differentiable forward warping in combination with feature pyramid synthesis [36], leveraging multiple optical flow fields from the perspective of the frame that is ought to be synthesized [41], or utilizing a coarse-to-fine interpolation scheme [54, 65]. In comparison to kernel-based approaches, flow-based interpolation techniques have the advantage that their motion estimation component can be supervised on additional training data with ground truth optical flow. Despite this additional supervision, our proposed SepConv++ still outperforms all flow-based methods with the exception of SoftSplat [36].

Other approaches for frame interpolation that neither use adaptive convolutions nor optical flow include techniques based on phase [32, 33] or approaches that directly synthesize the intermediate frame [12]. There are also in-
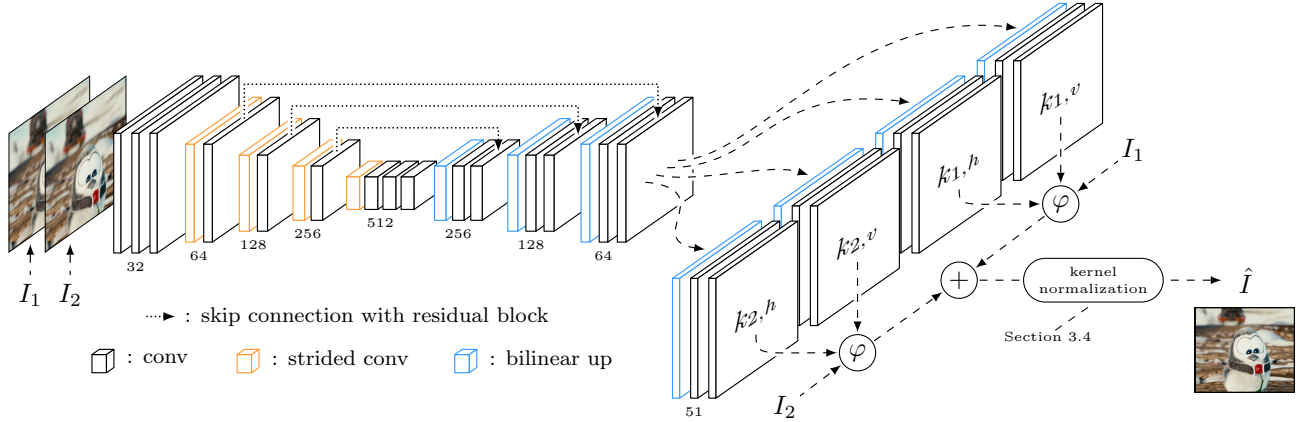
Figure 3: Overview of our frame interpolation framework where φ denotes the adaptive separable convolution operator. We adopted the illustration style of the SepConv paper [38] to make it easier to compare our architecture with the original one.

teresting research efforts that perform frame interpolation in unison with a second video processing task like super-resolution [23, 61], deblurring [46], dequantization [56], or with non-traditional acquisition setups [27, 40, 58]. Given that neural networks for frame interpolation are usually trained on off-the-shelf videos, it also seems natural to conduct research for test-time adaptation [11, 44]. While most frame interpolation techniques only operate on two frames as input and hence assume linear motion, there are also interesting approaches that assume quadratic or cubic motion models [10, 28, 62]. Our work is orthogonal to these ideas.

3. Method

Given two consecutive frames I_1 and I_2 from a video, the frame interpolation task that we are targeting is the synthesis of the intermediate frame $\hat{I}$ that is temporally centered between the given input frames. To achieve this, we use the approach from Niklaus *et al.* [38] that leverages adaptive separable convolutions by having a neural network ϕ predict a set of pixel-wise spatially-varying one-dimensional filter kernels $\langle K_{1,h}, K_{1,v}, K_{2,h}, K_{2,v} \rangle$ as follows.

$$\langle K_{1,h}, K_{1,v}, K_{2,h}, K_{2,v} \rangle = \phi(I_1, I_2) \qquad (1)$$

These spatially-varying kernels can then be used to process the input frames to yield $\hat{I}$ through an adaptive separable convolution operation φ. Specifically, I_1 is filtered with the separable filters $\langle K_{1,h}, K_{1,v} \rangle$ while I_2 is filtered with the separable filters $\langle K_{2,h}, K_{2,v} \rangle$ as follows.

$$\hat{I} = \varphi(I_1, K_{1,h}, K_{1,v}) + \varphi(I_2, K_{2,h}, K_{2,v}) \qquad (2)$$

These spatially-varying kernels capture motion and re-sampling information, which makes for an effective image formation model for frame interpolation. To be able to account for large motion, the kernels should be as large as possible. However, with larger kernels it is more difficult to estimate all coefficients. We adopt the relatively moderate

kernel size of 51 pixels from the original SepConv [38]. We subsequently describe our proposed techniques to improve adaptive separable convolutions for frame interpolation.

3.1. Delayed Padding

As with all convolutions of non-singular size, the input needs to be padded if the output has to have the same resolution as the input. Specifically, the original SepConv [38] pads the input frames by 25 pixels before estimating the adaptive kernel coefficients via a neural network ϕ.

$$\langle K_{1,h}, K_{1,v}, K_{2,h}, K_{2,v} \rangle = \phi(\text{pad}(I_1), \text{pad}(I_2)) \qquad (3)$$

In contrast, we propose not to pad the input images when they are given to ϕ but instead to pad them when the predicted kernels are applied to the input images as follows.

$$\hat{I} = \varphi(\text{pad}(I_1), K_{1,h}, K_{1,v}) + \varphi(\text{pad}(I_2), K_{2,h}, K_{2,v}) \qquad (4)$$

This delayed padding has two positive effects. First, it improves the computational efficiency. Using an Nvidia V100, the original SepConv implementation takes 0.027 seconds to interpolate a frame at a resolution of 512×512 pixels. In comparison, it takes 0.018 seconds with the delayed padding. At a resolution of 1024×1024 pixels, it takes 0.083 seconds with the original padding and 0.065 seconds with our proposed delayed padding. Second, it improves the quality of the interpolated results since the neural network ϕ does not have to deal with large padded boundaries that are outside of the manifold of natural images (we use replication padding as in [38]). Specifically, delayed padding improves the interpolation results on the Middlebury benchmark examples [1] with publicly known ground truth by 0.37 dB. Please see our ablation experiments in Section 4.1 for more details.

3.2. Input Normalization

The contrast and brightness of the input frames should not affect the quality of the synthesized results. In other words,
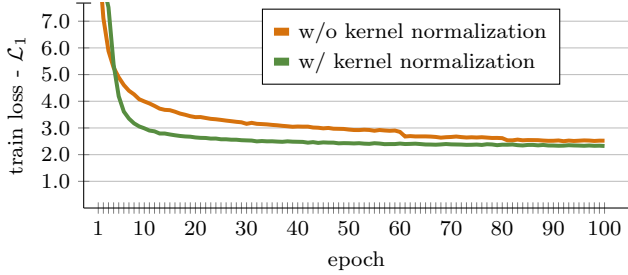
Figure 4: Comparing the training loss with and without our proposed kernel normalization, demonstrating that kernel normalization greatly improves the model convergence. Note that we halve the learning rate after 60 and 80 epochs.

the network should be invariant to contrast and brightness. While it would be difficult to enforce such an invariance during training, it can easily be achieved by normalizing the input frames before feeding them to the network and denormalizing the synthesized result [35]. For image synthesis via adaptive convolutions, one can skip the denormalization step by applying the adaptive convolutions on the original input frames and only normalizing them when feeding them to the neural network that predicts the spatially-varying kernels.

To normalize the input frames, we shift and rescale their intensity values to have zero mean and unit standard deviation. There are multiple possibilities to do so, we have found normalizing the two images jointly while treating each color channel separately to work well. That is, for each color channel we compute the mean and standard deviation of I_1 and I_2 as if they were one image. This input normalization improves the interpolation quality on the Middlebury benchmark examples [1] with publicly known ground truth by 0.31 dB. We also tried separately normalizing the input frames and computing singular instead of per-channel statistics but have found these approaches to be less effective.

One could make a similar argument about shift invariance since the quality of the synthesized interpolation results should not be affected by jointly translating the input frames. However, we have not been able to improve the interpolation quality by incorporating current techniques for improving the shift invariance of a neural network [66, 69].

3.3. Network Improvements

Since the publciation of SepConv [38], there has been great progress in deep learning architectures, and we experimented with incorporating these into an updated neural network architecture as shown in Figure 3. Specifically, we added residual blocks [16] to the skip connections that join the two halves of the U-Net, we changed the activation function to parametric rectified linear units [15], we replaced the average pooling with strided convolutions, and we use a Kahan sum within the adaptive separable convolutions. Together, these changes lead to a 0.42 dB improvement in

(a) Ours - $\mathcal{L}_{Ctx}$ (b) Ours - $\mathcal{L}_{Ctx}$ - 8×
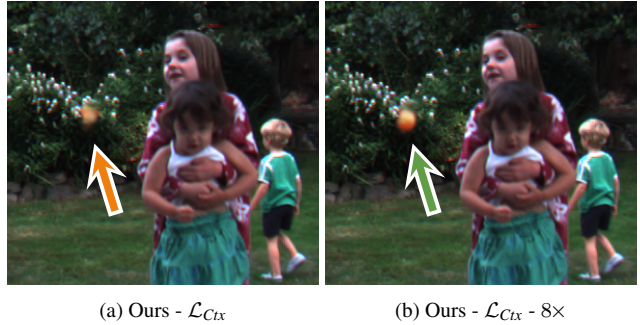
Figure 5: Comparing interpolation results without (a) and with (b) ensembling where eight independent predictions are combined (ensembling smooths uncertain estimates).

terms of interpolation quality on the Middlebury benchmark examples [1] with publicly known ground truth.

These architectural changes do unfortunately not come for free. While it took 0.065 seconds to synthesize a result at a resolution of 1024×1024 pixels using an Nvidia V100 with the original SepConv architecture, the new architecture takes 0.185 seconds. This is surprising at first since our new architecture has fewer convolutions than the original one (31 versus 47 convolutions) and our architecture is more parameter-efficient overall (13.6 million versus 21.7 million parameters). However, our sub-networks that predict the adaptive kernel coefficients perform bilinear upsampling first whereas the original network performed bilinear upsampling last which leads to a significant increase in compute.

3.4. Kernel Normalization

The initial paper on adaptive convolutions for video frame interpolation includes a softmax layer to normalize the kernels [37], which is similar to but different from normalized convolutions [24] for addressing missing samples in the input signal. Such a kernel normalization is missing in the separable formulation since a softmax layer can no longer be used with this setup [38]. As a result, the neural network that predicts the kernel coefficients needs to take great care not to alter the apparent brightness of a synthesized pixel. We propose a simple normalization step that can be applied to any kernel-based image formation model. Specifically, we not only apply the adaptive separable convolution on the input but also on a singular mask. Afterwards, the filtered input can be divided by the filtered mask as follows such that denormalized kernel weights are compensated for.

$$\hat{I} = \frac{\varphi\left(I_1, K_{1,h}, K_{1,v}\right) + \varphi\left(I_2, K_{2,h}, K_{2,v}\right)}{\varphi\left(\mathbf{1}, K_{1,h}, K_{1,v}\right) + \varphi\left(\mathbf{1}, K_{2,h}, K_{2,v}\right)} \quad (5)$$

This simple kernel normalization step improves the quality of the synthesis results and greatly helps with the convergence of the model during training as shown in Figure 4. With an improvement by 0.52 dB on the Middlebury benchmark examples [1] with publicly known ground truth, the

	training dataset	Middlebury Baker et al. [1]		Vimeo-90k Xue et al. [63]		UCF101 - DVF Liu et al. [29]		Xiph - 1K (4K resized to 1K)		Xiph - 2K (4K resized to 2K)	
		PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement
original SepConv	proprietary	35.73	–	33.80	–	34.79	–	36.22	–	34.77	–
reimplementation	Vimeo-90k	35.49	–	33.81	–	34.63	–	35.89	–	34.18	–
+ delayed padding	— „ —	35.86	+ 0.37 dB	34.31	+ 0.50 dB	35.09	+ 0.46 dB	36.00	+ 0.11 dB	34.16	- 0.02 dB
+ input normalization	— „ —	36.16	+ 0.30 dB	34.50	+ 0.19 dB	35.18	+ 0.09 dB	36.06	+ 0.06 dB	34.14	- 0.02 dB
+ improved network	— „ —	36.58	+ 0.42 dB	34.55	+ 0.05 dB	35.19	+ 0.01 dB	36.58	+ 0.52 dB	34.76	+ 0.62 dB
+ normalized kernels	— „ —	37.10	+ 0.52 dB	34.79	+ 0.24 dB	35.22	+ 0.03 dB	36.78	+ 0.20 dB	34.77	+ 0.01 dB
+ contextual training	— „ —	37.28	+ 0.18 dB	34.83	+ 0.04 dB	35.24	+ 0.02 dB	36.83	+ 0.05 dB	34.84	+ 0.07 dB
+ self-ensembling	— „ —	37.46	+ 0.18 dB	34.97	+ 0.14 dB	35.29	+ 0.05 dB	37.00	+ 0.17 dB	35.10	+ 0.26 dB

Table 1: Ablation experiments to quantitatively analyze the effects of our proposed techniques. In short, they each positively affect the interpolation quality across different dataset as long as the inter-frame motion does not exceed the kernel size.

kernel normalization has the most significant impact on the quality of the synthesized results. Please see our ablation experiments in Section 4.1 for more details.

3.5. Contextual Training

With adaptive convolutions for video frame interpolation, there is no constraint that forces the kernel prediction network to estimate coefficients that account for the true motion. Instead, the kernel prediction network may simply index pixels that have the desired color, which is similar to view synthesis by appearance flow [68]. This may hurt the generalizability of the trained neural network though, which is why we force it to predict coefficients that agree with the true motion through a contextual loss. Specifically, we not only filter the input frames but also their context which have been obtained from an off-the-shelf network ψ (we have found relu1_2 of a pre-trained VGG [49] network and a trade-off weight $\alpha = 0.1$ to be effective). We then minimize not only the difference between the prediction and the ground truth in color space but also in the contextual space as follows (note that we omitted the previously introduced kernel normalization step in this definition for brevity).

$$\mathcal{L}_{Ctx} = \left\| \langle \hat{I}, \alpha \cdot \hat{I}_\psi \rangle - \langle I_{gt}, \alpha \cdot \psi(I_{gt}) \rangle \right\|_1 \qquad (6)$$

where

$$\hat{I} = \varphi(I_1, K_{1,h}, K_{1,v}) + \varphi(I_2, K_{2,h}, K_{2,v}) \qquad (7)$$

$$\hat{I}_\psi = \varphi(\psi(I_1), K_{1,h}, K_{1,v}) + \varphi(\psi(I_2), K_{2,h}, K_{2,v}) \quad (8)$$

Since each pixel in the contextual space not only describes the color of a single pixel but also encodes its local neighborhood, this loss effectively prevents the kernel prediction network from simply indexing pixels based on their color. Supervising the kernel prediction using this contextual loss yields an improvement of 0.18 dB on the Middlebury benchmark examples [1] with publicly know ground truth.

Note that while this loss shares resemblance to a content loss [20], it is fundamentally different from it. A content loss applies a VGG feature loss directly on the synthesized result which would not prevent a kernel prediction network from estimating coefficients that mimic appearance flow. In contrast, we extract VGG features from the input frames before applying the adaptive separable convolution. In doing so, the kernel prediction network cannot just index a pixel with the same color as the ground truth as this may lead to significant differences in the VGG feature space.

3.6. Self-ensembling

In image classification [7] and super-resolution [53], a singular prediction is often enhanced by combining the predictions of multiple transformed versions of the same input. Such transforms include rotations, mirroring, or cropping. Not all image transforms are reversible though, but they have to be when wanting to combine predictions of pixel-wise tasks. Surprisingly, there is no study of the effect of such a self-ensembling approach in the context of frame interpolation. We hence propose to adopt this enhancement scheme for frame interpolation and conduct a large-scale study of its effect in Section 4.2, demonstrating improvements across the board. In doing so, we consider taking the mean and taking the median of up to sixteen predictions with transforms based on reversing the input frames, flipping them, mirroring them, and applying rotations by ninety degrees.

Effectively, self-ensembling for video frame interpolation smooths predictions in areas where the kernel estimation is uncertain. As shown in Figure 5, this can visually lead to a smooth result instead of one with visible artifacts. However, self-ensembling can computationally be prohibitively expensive as, for example, using eight predictions instead of just a single one will require eight times more compute.

4. Experiments

We subsequently evaluate our contributions by answering the following questions. What is the impact of each of our proposed techniques? What is the effect of self-ensembling

		Middlebury Baker et al. [1]		Vimeo-90k Xue et al. [63]		UCF101 - DVF Liu et al. [29]		Xiph - 1K (4K resized to 1K)		Xiph - 2K (4K resized to 2K)	
	reduction	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement
Ours - $\mathcal{L}_{Ctx}$	none	37.28	–	34.83	–	35.24	–	36.83	–	34.84	–
Ours - $\mathcal{L}_{Ctx}$ - 2×	mean	37.41	+ 0.13 dB	34.91	+ 0.08 dB	35.26	+ 0.02 dB	36.92	+ 0.09 dB	35.01	+ 0.17 dB
Ours - $\mathcal{L}_{Ctx}$ - 4×	mean	37.45	+ 0.04 dB	34.95	+ 0.04 dB	35.28	+ 0.02 dB	36.98	+ 0.06 dB	35.09	+ 0.08 dB
Ours - $\mathcal{L}_{Ctx}$ - 8×	mean	<u>37.46</u>	+ 0.01 dB	34.97	+ 0.02 dB	35.29	+ 0.01 dB	37.00	+ 0.02 dB	35.10	+ 0.01 dB
Ours - $\mathcal{L}_{Ctx}$ - 16×	mean	<u>37.46</u>	+ 0.00 dB	<u>34.99</u>	+ 0.02 dB	<u>35.30</u>	+ 0.01 dB	<u>37.02</u>	+ 0.02 dB	<u>35.13</u>	+ 0.03 dB
Ours - $\mathcal{L}_{Ctx}$	none	37.28	–	34.83	–	35.24	–	36.83	–	34.84	–
Ours - $\mathcal{L}_{Ctx}$ - 2×	median	37.41	+ 0.13 dB	34.91	+ 0.08 dB	35.26	+ 0.02 dB	36.92	+ 0.09 dB	35.01	+ 0.17 dB
Ours - $\mathcal{L}_{Ctx}$ - 4×	median	37.44	+ 0.03 dB	34.94	+ 0.03 dB	35.27	+ 0.01 dB	36.97	+ 0.05 dB	35.07	+ 0.06 dB
Ours - $\mathcal{L}_{Ctx}$ - 8×	median	<u>37.47</u>	+ 0.03 dB	34.96	+ 0.02 dB	35.28	+ 0.01 dB	36.99	+ 0.02 dB	35.09	+ 0.02 dB
Ours - $\mathcal{L}_{Ctx}$ - 16×	median	<u>37.47</u>	+ 0.00 dB	<u>34.98</u>	+ 0.02 dB	<u>35.29</u>	+ 0.01 dB	<u>37.01</u>	+ 0.02 dB	<u>35.13</u>	+ 0.04 dB

Table 2: Effect of combining multiple independent predictions when taking their mean (top) and their median (bottom).

		Middlebury Baker et al. [1]		Vimeo-90k Xue et al. [63]		UCF101 - DVF Liu et al. [29]		Xiph - 1K (4K resized to 1K)		Xiph - 2K (4K resized to 2K)	
	venue	without ensemble	with ensemble	without ensemble	with ensemble	without ensemble	with ensemble	without ensemble	with ensemble	without ensemble	with ensemble
SepConv - $\mathcal{L}_1$	ICCV 2017	35.73 +0.41,	36.14	33.80 +0.24,	34.04	34.79 +0.14,	34.93	36.22 +0.24,	36.46	34.77 +0.39,	35.16
CtxSyn - $\mathcal{L}_{Lap}$	CVPR 2018	36.93 +0.39,	37.32	34.39 +0.31,	34.70	34.62 +0.27,	34.89	36.87 +0.32,	37.19	35.72 +0.32,	36.04
DAIN	CVPR 2019	36.69 +0.35,	37.04	34.70 +0.25,	34.95	35.00 +0.13,	35.13	36.78 +0.26,	37.04	35.93 +0.26,	36.19
CAIN	AAAI 2020	35.11 +0.23,	35.34	34.65 +0.16,	34.81	34.98 +0.10,	35.08	36.21 +0.19,	36.40	35.18 +0.21,	35.39
EDSC - $\mathcal{L}_C$	arXiv 2020	36.82 +0.44,	37.26	34.83 +0.27,	35.10	35.13 +0.09,	35.22	36.73 +0.31,	37.04	OOM +0.00,	OOM
AdaCoF	CVPR 2020	35.72 +0.47,	36.19	34.35 +0.45,	34.80	35.16 +0.12,	35.28	36.26 +0.47,	36.73	34.82 +0.40,	35.22
SoftSplat - $\mathcal{L}_{Lap}$	CVPR 2020	38.42 +0.26,	38.68	36.10 +0.18,	36.28	35.39 +0.09,	35.48	37.96 +0.23,	38.19	36.63 +0.16,	36.79
BMBC	ECCV 2020	36.79 +0.12,	36.91	35.06 +0.15,	35.21	35.16 +0.08,	35.24	36.59 +0.37,	36.96	OOM +0.00,	OOM
Ours - $\mathcal{L}_{Ctx}$	N/A	37.28 +0.18,	37.46	34.83 +0.14,	34.97	35.24 +0.05,	35.29	36.83 +0.17,	37.00	34.84 +0.26,	35.10

Table 3: Effect of combining the mean of eight independent predictions for several video frame interpolation methods.

for video frame interpolation? How does our SepConv++ compare to the original SepConv? How does our SepConv++ compare to other frame interpolation techniques?

Implementation. We generally follow the implementation details of the original SepConv [38]. However, they were using a proprietary training dataset whereas our reimplementation is incorporating Vimeo-90k [63] instead. Furthermore, we simplified the augmentation pipeline and refrained from shifting the cropping windows in opposing directions.

Datasets. We adopt the test set selection from [36] and conduct our evaluation on Vimeo-90k [63], the Middlebury benchmark samples with publicly known ground truth [1], the Liu et al. [29] samples from UCF101 [50], and footage from Xiph[1]. We do not adopt the "4K" cropped version of Xiph that [36] proposed though and instead focus on downscaled versions at a resolution of 1K as well as 2K.

Metrics. We limit the evaluation herein to the PSNR metric since SSIM [57] is subject to unexpected and unintuitive results [39]. However, we provide equivalent tables with

SSIM instead of PSNR in the supplementary material. These supplementary results support our claims and are generally aligned with PSNR in terms of relative improvements.

4.1. Ablation Experiments

We analyze how our proposed techniques affect the interpolation quality in Table 1. In short, each technique improves the synthesis quality in terms of PSNR across a variety of datasets. The results on the UCF101 samples as well as the 2K version of the Xiph videos are relatively inconsistent though. However, this behavior is not surprising as several of the UCF101 samples are invalid where the ground truth is identical to either the first or the second input frame (like for examples 1, 141, or 271). As for the high-resolution Xiph videos, the amount of inter-frame motion that is present in in-the-wild 2K footage is expected to exceed the maximum magnitude of 51 pixels that our adaptive separable kernels can compensate for. We also note that our reimplementation is subject to worse results than the original SepConv [38] on all datasets except the test split of our training dataset. This finding indicates that there are better training datasets than

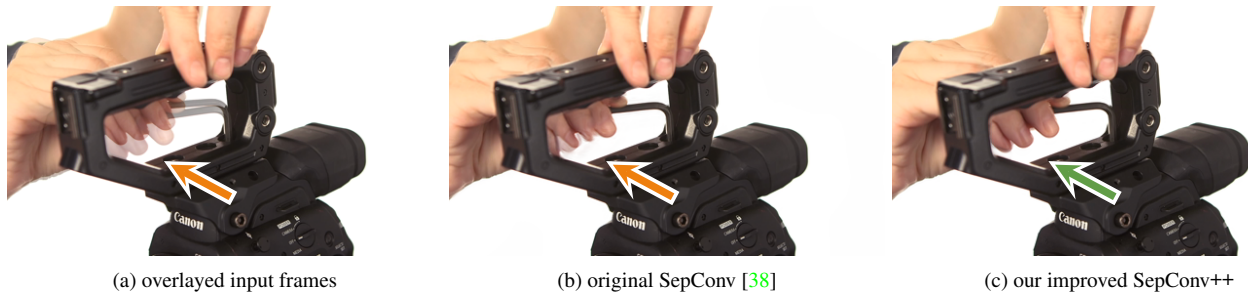| (a) overlayed input frames | (b) original SepConv [38] | (c) our improved SepConv++ |

Figure 6: Qualitative comparison with SepConv. We purposefully only show a single example here for brevity and kindly refer to our supplementary video which shows this example as well as many more examples in a fully interpolated sequence.

	training dataset	Middlebury Baker *et al*. [1]		Vimeo-90k Xue *et al*. [63]		UCF101 - DVF Liu *et al*. [29]		Xiph - 1K (4K resized to 1K)		Xiph - 2K (4K resized to 2K)	
		PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement	PSNR ↑	relative improvement
SepConv - $\mathcal{L}_1$	proprietary	35.73	–	33.80	–	34.79	–	36.22	–	34.77	–
Ours - $\mathcal{L}_{Ctx}$	Vimeo-90k	37.28	+ 1.55 dB	34.83	+ 1.03 dB	35.24	+ 0.45 dB	36.83	+ 0.61 dB	34.84	+ 0.07 dB
Ours - $\mathcal{L}_{Ctx}$ - 8×	— ,, —	37.46	+ 0.18 dB	34.97	+ 0.14 dB	35.29	+ 0.05 dB	37.00	+ 0.17 dB	35.10	+ 0.26 dB

Table 4: Quantitative comparison with SepConv. We list two separate results of our proposed approach, one without and one with self-ensembling. The self-ensembling is denoted by 8× as it represents a combination of eight independent estimates.

Vimeo-90k [63] for supervising video frame interpolation tasks and that our proposed SepConv++ could perform even better if it had been supervised on the dataset from [38].

4.2. Self-ensembling for Frame Interpolation

Our findings on self-ensembling for video frame interpolation summarized in Table 2 where we take the mean as well as the median of up to sixteen independent prediction. These findings indicate that any form of self-ensembling is superior to a singular prediction, while taking the mean or taking the median is similarly effective. However, there are diminishing returns in the number of predictions.

Next, we analyze the effect of self-ensembling on all methods that we compare to in this paper. Specifically, we take the mean of eight independent predictions since using sixteen predictions takes much more compute while providing little benefit. As shown in Table 3, all methods benefit from self-ensembling across all datasets. However, self-ensembling has little benefit for practical applications of these frame interpolation techniques since they can already take minutes to process a single second of high-resolution footage. By combining eight independent predictions, this processing time can now become tens of minutes to process a single second of high-resolution footage which is beyond the threshold of being practical for many applications.

4.3. Comparison with SepConv

The premise of our paper is that an older and simpler frame interpolation approach, namely SepConv [38], can be optimized to produce near state-of-the-art results. In

this section, we compare our SepConv++ with the original SepConv. Specifically, we show a representative qualitative result in Figure 6 which demonstrates the efficacy of our proposed techniques. Please also consider our supplementary video to better examine this as well as additional examples in motion. Our quantitative comparison in Table 4 further shows that our proposed techniques are effective across a variety of datasets as long as the inter-frame motion does not exceed the kernel size (as it occurs for the 2K version of Xiph). Note that this table lists our results with and without self-ensembling for fairness since self-ensembling can easily be applied to all video frame interpolation methods.

4.4. Comparison with Others

Even though we base our approach off of an older and simpler frame interpolation technique, we are able to achieve near state-of-the-art quality. To demonstrate this, we compare our SepConv++ to competitive approaches for frame interpolation based on kernel prediction (SepConv [38], EDSC [8], and AdaCoF [25]), based on optical flow estimation and compensation (CtxSyn [35], DAIN [3], Soft-Splat [36], and BMBC [41]), and based on directly synthesizing the intermediate frame (CAIN [12]). We summarize the findings in Table 5, where we separately list our results with and without self-ensembling for fairness. In summary, our proposed approach is only outperformed by SoftSplat and generally does not fair well on the 2K version of the Xiph footage where the inter-frame motion exceeds what our adaptive separable kernels with 51 pixels can compensate for. However, SoftSplat was additionally supervised on training data with ground truth optical flow, whereas our approach

	venue	Middlebury Baker et al. [1]		Vimeo-90k Xue et al. [63]		UCF101 - DVF Liu et al. [29]		Xiph - 1K (4K resized to 1K)		Xiph - 2K (4K resized to 2K)	
		PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank	PSNR ↑	absolute rank
SepConv - $\mathcal{L}_1$	ICCV 2017	35.73	8th of 10	33.80	10th of 10	34.79	9th of 10	36.22	9th of 10	34.77	8th of 10
CtxSyn - $\mathcal{L}_{Lap}$	CVPR 2018	36.93	4th of 10	34.39	8th of 10	34.62	10th of 10	36.87	3rd of 10	35.72	3rd of 10
DAIN	CVPR 2019	36.69	7th of 10	34.70	6th of 10	35.00	7th of 10	36.78	5th of 10	35.93	2nd of 10
CAIN	AAAI 2020	35.11	10th of 10	34.65	7th of 10	34.98	8th of 10	36.21	10th of 10	35.18	4th of 10
EDSC - $\mathcal{L}_C$	arXiv 2020	36.82	5th of 10	34.83	4th of 10	35.13	6th of 10	36.73	6th of 10	OOM	OOM
AdaCoF	CVPR 2020	35.72	9th of 10	34.35	9th of 10	35.16	4th of 10	36.26	8th of 10	34.82	7th of 10
SoftSplat - $\mathcal{L}_{Lap}$	CVPR 2020	38.42	1st of 10	36.10	1st of 10	35.39	1st of 10	37.96	1st of 10	36.63	1st of 10
BMBC	ECCV 2020	36.79	6th of 10	35.06	2nd of 10	35.16	4th of 10	36.59	7th of 10	OOM	OOM
Ours - $\mathcal{L}_{Ctx}$	N/A	37.28	3rd of 10	34.83	4th of 10	35.24	3rd of 10	36.83	4th of 10	34.84	6th of 10
Ours - $\mathcal{L}_{Ctx}$ - 8×	— ,, —	37.46	2nd of 10	34.97	3rd of 10	35.29	2nd of 10	37.00	2nd of 10	35.10	5th of 10

Table 5: Quantitative comparison with recent approaches for video frame interpolation. In addition to highlighting the best result by underlining it, we emphasize the second-best result via a dotted underline. Note that some methods were unable to run on 2K footage due to exceeding the 16 gigabytes of memory available on our graphics card (denoted as "OOM").

	Middlebury (mean error)		Middlebury (median error)	
	IE ↓	absolute rank	IE ↓	absolute rank
SepConv - $\mathcal{L}_1$	5.61	8th of 9	5.44	8th of 9
CtxSyn - $\mathcal{L}_{Lap}$	5.28	7th of 9	4.77	7th of 9
DAIN	4.86	6th of 9	4.69	5th of 9
CAIN	–	–	–	–
EDSC - $\mathcal{L}_C$	4.72	4th of 9	4.69	5th of 9
AdaCoF	4.75	5th of 9	4.48	4th of 9
SoftSplat - $\mathcal{L}_{Lap}$	4.22	1st of 9	3.97	1st of 9
BMBC	4.48	3rd of 9	4.16	3rd of 9
Ours - $\mathcal{L}_{Ctx}$	4.45	2nd of 9	4.13	2nd of 9

Table 6: Quantitative results on the official Middlebury benchmark [1]. This benchmark does not list CAIN [12].

was solely supervised on the Vimeo-90k [63] dataset.

We also submitted our results to the organizers of the Middlebury benchmark [1] where our SepConv++ ranks second in terms of interpolation error among all published methods (currently not publicly visible, please see our supplementary material). We show a summary of the benchmark in Table 6 with all methods that we compare to in this paper.

4.5. Discussion

While we were able to show that it is possible to achieve near state-of-the-art video frame interpolation results using an older technique by carefully optimizing it, we did not fundamentally alter its image formation model. As such, the two main limitations of video frame interpolation via adaptive separable convolutions remain. First, they often do not produce satisfying interpolation results on high-resolution input footage due to the limited kernel size. This is exemplified by the relatively poor performance of SepConv++ on the

2K version of the Xiph footage. Second, they are limited to synthesizing the interpolation result at the temporal position that they have been supervised on. To synthesize interpolation results at $t = 0.75$ instead of $t = 0.5$, one would either have to train a kernel prediction network with the ground truth at $t = 0.75$ or first synthesize the interpolation result at $t = 0.5$ and then recursively use this result as well as the input at $t = 1$ to yield $t = 0.75$. This limits practical applications where, for example, a given video with 50 frames per second needs to be converted to 60 frames per second.

5. Conclusion

In this paper, we show, somewhat surprisingly, that it is possible to achieve near state-of-the-art video frame interpolation results with an older, simpler approach by carefully optimizing it. Our optimizations are conceptually intuitive, effective in improving the interpolation quality, and are directly applicable to a variety of frame interpolation techniques. Furthermore, while our paper focuses on improving adaptive separable convolutions for the purpose of frame interpolation, some of our proposed techniques may be applicable to related applications as well, such as image denoising, joint image filtering, or video prediction.

References

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 1, 2, 3, 4, 5, 6, 7, 8

[2] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics*, 36(4):97:1–97:14, 2017. 2

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-Aware Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 7

[4] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement. *arXiv/1810.08768*, 2018. 1, 2

[5] Tim Brooks and Jonathan T. Barron. Learning to Synthesize Motion Blur. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[6] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward Real-World Single Image Super-Resolution: A New Benchmark and a New Model. In *IEEE International Conference on Computer Vision*, 2019. 2

[7] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep Into Convolutional Nets. In *British Machine Vision Conference*, 2014. 5

[8] Xianhang Cheng and Zhenzhong Chen. Multiple Video Frame Interpolation via Enhanced Deformable Separable Convolution. *arXiv/2006.08070*, 2020. 1, 2, 7

[9] Xianhang Cheng and Zhenzhong Chen. Video Frame Interpolation via Deformable Separable Convolution. In *AAAI Conference on Artificial Intelligence*, 2020. 1, 2

[10] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N. Plataniotis. All at Once: Temporally Adaptive Multi-Frame Interpolation With Advanced Motion Modeling. In *European Conference on Computer Vision*, 2020. 3

[11] Myungsub Choi, Janghoon Choi, Sungyong Baik, Tae Hyun Kim, and Kyoung Mu Lee. Scene-Adaptive Video Frame Interpolation via Meta-Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[12] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel Attention Is All You Need for Video Frame Interpolation. In *AAAI Conference on Artificial Intelligence*, 2020. 2, 7, 8

[13] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised Learning for Physical Interaction Through Video Prediction. In *Advances in Neural Information Processing Systems*, 2016. 2

[14] Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. Sample-Based Monte Carlo Denoising Using a Kernel-Splatting Network. *ACM Transactions on Graphics*, 38(4):125:1–125:12, 2019. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep Into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision*, 2015. 4

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4

[17] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic Filter Networks. In *Advances in Neural Information Processing Systems*, 2016. 2

[18] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[19] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit Motion Compensation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *European Conference on Computer Vision*, 2016. 5

[21] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics*, 35(6):193:1–193:10, 2016. 1

[22] Beomjun Kim, Jean Ponce, and Bumsub Ham. Deformable Kernel Networks for Joint Image Filtering. *arXiv/1910.08373*, 2019. 2

[23] Soo Ye Kim, Jihyong Oh, and Munchurl Kim. FISR: Deep Joint Frame Interpolation and Super-Resolution With a Multi-Scale Temporal Loss. In *AAAI Conference on Artificial Intelligence*, 2020. 3

[24] Hans Knutsson and Carl-Fredrik Westin. Normalized and Differential Convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1993. 4

[25] Hyeongmin Lee, Taeoh Kim, Tae-Young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 7

[26] Zhengqi Li and Noah Snavely. MegaDepth: Learning Single-View Depth Prediction From Internet Photos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[27] Songnan Lin, Jiawei Zhang, Jinshan Pan, Zhe Jiang, Dongqing Zou, Yongtian Wang, Jing Chen, and Jimmy Ren. Learning Event-Driven Video Deblurring and Interpolation. In *European Conference on Computer Vision*, 2020. 3

[28] Yihao Liu, Liangbin Xie, Siyao Li, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced Quadratic Video Interpolation. *arXiv/2009.04642*, 2020. 3

[29] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video Frame Synthesis Using Deep Voxel Flow. In *IEEE International Conference on Computer Vision*, 2017. 2, 5, 6, 7, 8

[30] Talmaj Marinc, Vignesh Srinivasan, Serhan Gül, Cornelius Hellge, and Wojciech Samek. Multi-Kernel Prediction Networks for Denoising of Burst Images. In *IEEE International Conference on Image Processing*, 2019. 2

[31] Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus H. Gross. Deep Video Color Propagation. In *British Machine Vision Conference*, 2018. 1

[32] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus H. Gross, and Christopher Schroers. PhaseNet for Video Frame Interpolation. In

IEEE Conference on Computer Vision and Pattern Recognition, 2018. 2

[33] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-Based Frame Interpolation for Video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 2

[34] Ben Mildenhall, Jonathan T. Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst Denoising With Kernel Prediction Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[35] Simon Niklaus and Feng Liu. Context-Aware Synthesis for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 4, 7

[36] Simon Niklaus and Feng Liu. Softmax Splatting for Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 6, 7

[37] Simon Niklaus, Long Mai, and Feng Liu. Video Frame Interpolation via Adaptive Convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 4

[38] Simon Niklaus, Long Mai, and Feng Liu. Video Frame Interpolation via Adaptive Separable Convolution. In *IEEE International Conference on Computer Vision*, 2017. 1, 2, 3, 4, 6, 7

[39] Jim Nilsson and Tomas Akenine-Möller. Understanding SSIM. *arXiv/2006.13846*, 2020. 6

[40] Avinash Paliwal and Nima Khademi Kalantari. Deep Slow Motion Video Reconstruction With Hybrid Imaging System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1557–1569, 2020. 3

[41] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. BMBC: Bilateral Motion Estimation With Bilateral Cost Volume for Video Interpolation. In *European Conference on Computer Vision*, 2020. 2, 7

[42] Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. IM-Net for High Resolution Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[43] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. SDC-Net: Video Prediction Using Spatially-Displaced Convolution. In *European Conference on Computer Vision*, 2018. 2

[44] Fitsum A. Reda, Deqing Sun, Aysegul Dundar, Mohammad Shoeybi, Guilin Liu, Kevin J. Shih, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Unsupervised Video Interpolation Using Cycle Consistency. In *IEEE International Conference on Computer Vision*, 2019. 3

[45] Steven M. Seitz and Simon Baker. Filter Flow. In *IEEE International Conference on Computer Vision*, 2009. 2

[46] Wang Shen, Wenbo Bao, Guangtao Zhai, Li Chen, Xiongkuo Min, and Zhiyong Gao. Blurry Video Frame Interpolation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[47] Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video Interpolation via Generalized Deformable Convolution. *arXiv/2008.10680*, 2020. 2

[48] Hyeonjun Sim and Munchurl Kim. A Deep Motion Deblurring Network Based on Per-Pixel Adaptive Kernels With Residual Down-Up and Up-Down Modules. In *CVPR Workshops*, 2019. 2

[49] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv/1409.1556*, 2014. 5

[50] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in the Wild. *arXiv/1212.0402*, 2012. 6

[51] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[52] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision*, 2020. 2

[53] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven Ways to Improve Example-Based Single Image Super Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 5

[54] Joost R. van Amersfoort, Wenzhe Shi, Alejandro Acosta, Francisco Massa, Johannes Totz, Zehan Wang, and Jose Caballero. Frame Interpolation With Multi-Scale Deep Loss Functions and Generative Adversarial Networks. *arXiv/1711.06045*, 2017. 2

[55] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising With Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics*, 37(4):124:1–124:15, 2018. 2

[56] Yang Wang, Haibin Huang, Chuan Wang, Tong He, Jue Wang, and Minh Hoai. GIF2Video: Color Dequantization and Temporal Interpolation of GIF Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3

[57] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[58] Zihao W. Wang, Weixin Jiang, Kuan He, Boxin Shi, Aggelos K. Katsaggelos, and Oliver Cossairt. Event-Driven Video Frame Synthesis. In *ICCV Workshops*, 2019. 3

[59] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video Compression Through Image Interpolation. In *European Conference on Computer Vision*, 2018. 1

[60] Zhihao Xia, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. Basis Prediction Networks for Effective Burst Denoising With Large Kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[61] Xiaoyu Xiang, Yapeng Tian, Yulun Zhang, Yun Fu, Jan P. Allebach, and Chenliang Xu. Zooming Slow-Mo: Fast and Accurate One-Stage Space-Time Video Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[62] Xiangyu Xu, Li Si-Yao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic Video Interpolation. In *Advances in Neural Information Processing Systems*, 2019. 3

[63] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video Enhancement With Task-Oriented Flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 2, 5, 6, 7, 8

[64] Tianfan Xue, Jiajun Wu, Katherine L. Bouman, and Bill Freeman. Visual Dynamics: Probabilistic Future Frame Synthesis via Cross Convolutional Networks. In *Advances in Neural Information Processing Systems*, 2016. 2

[65] Haoxian Zhang, Yang Zhao, and Ronggang Wang. A Flexible Recurrent Residual Pyramid Network for Video Frame Interpolation. In *European Conference on Computer Vision*, 2020. 2

[66] Richard Zhang. Making Convolutional Networks Shift-Invariant Again. In *International Conference on Machine Learning*, 2019. 4

[67] Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Wangmeng Zuo, Haozhe Xie, and Jimmy S. J. Ren. Spatio-Temporal Filter Adaptive Network for Video Deblurring. In *IEEE International Conference on Computer Vision*, 2019. 2

[68] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A. Efros. View Synthesis by Appearance Flow. In *European Conference on Computer Vision*, 2016. 5

[69] Xueyan Zou, Fanyi Xiao, Zhiding Yu, and Yong Jae Lee. Delving Deeper Into Anti-Aliasing in ConvNets. *arXiv/2008.09604*, 2020. 4