Revisiting Adaptive Convolutions for Video Frame Interpolation

Simon Niklaus
Adobe Research

Long Mai
Adobe Research

Oliver Wang
Adobe Research

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 convolutions [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 three-fold. 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 (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-
where they can be used to merge multiple observations with
tive separable convolutions [38], adaptive warping layers
interesting new developments such as the usage of adap-
tics based on phase [32, 33] or approaches that directly
these methods are much more sophisticated.

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 fil-
tering [22, 52], in video prediction where adaptive kernels
 can also model uncertainty [13, 17, 43, 64], in deburring
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 ap-
proaches 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 in-
terpolation, Niklaus et al. [37] proposed to perform joint
motion estimation and motion compensation based on pre-
dicting spatially-varying kernels. This idea led to several
interesting new developments such as the usage of adap-
tive separable convolutions [38], adaptive warping layers
that combine optical flow estimates and adaptive convolu-
tions [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 experi-
mentation, 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 representa-
tions 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 tech-
niques have the advantage that their motion estimation
component can be supervised on additional training data
with ground truth optical flow. Despite this additional su-
ervision, 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 tech-
niques based on phase [32, 33] or approaches that directly
synthesize the intermediate frame [12]. There are also in-
These spatially-varying kernels capture motion and re-
process all coefficients. We adopt the relatively moderate
sampling information, which makes for an effective image
 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 reso-
lution 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 \( \phi \).

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

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

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

This delayed padding has two positive effects. First, it im-
proves the computational efficiency. Using an Nvidia V100,
the original SepConv implementation takes 0.027 seconds
to interpolate a frame at a resolution of \( 512 \times 512 \) pixels. In
comparison, it takes 0.018 seconds with the delayed padding.
At a resolution of \( 1024 \times 1024 \) pixels, it takes 0.083 seconds
with the original padding and 0.065 seconds with our pro-
posed delayed padding. Second, it improves the quality of the
interpolated results since the neural network \( \phi \) 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 in-
terpolation 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 publication 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 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.

$$
\hat{I} = \frac{\varphi(\hat{I}_1, K_{1,h}, K_{1,v}) + \varphi(\hat{I}_2, K_{2,h}, K_{2,v})}{\varphi(1, K_{1,h}, K_{1,v}) + \varphi(1, K_{2,h}, K_{2,v})}
$$

(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
where we force it to predict coefficients that agree with the work to estimate coefficients that account for the true motion.

yields an improvement of \( -18 \) dB on the Middlebury benchmark 

Supervising the kernel prediction using this contextual loss 

Since each pixel in the contextual space not only describes 

mimize not only the difference between the prediction and the ground truth in color space but also in the contextual space 

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.

### 3.5. 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 \( \psi \) (we have found \texttt{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).

\[
\mathcal{L}_{\text{Ctx}} = \| \langle \hat{I}, \alpha \cdot \hat{I}_\psi \rangle - \langle I_{gt}, \alpha \cdot \psi(I_{gt}) \rangle \|_1 \tag{6}
\]

where

\[
\hat{I} = \varphi(I_1, K_{1,h}, K_{1,v}) + \varphi(I_2, K_{2,h}, K_{2,v}) \tag{7}
\]

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

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 known ground truth.

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.

### 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
we simplified the augmentation pipeline and refrained from implementing.

can we compare to other frame interpolation techniques? 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ootnote{https://media.xiph.org/video/derf}. 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 the 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...
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 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], SoftSplat [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...
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
[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. 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 Røthe, 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, Zihan Wang, and Jose Caballero. Frame Interpolation With Multi-Scale Deep Loss Functions and Generative Adversarial Networks. arXiv/1711.06045, 2017. 2

[55] Thïjs 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, Rue 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, Michael 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, Wenshu 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