12-2017

Video Frame Interpolation via Adaptive Separable Convolution

Simon Niklaus
Portland State University, sniklaus@pdx.edu

Long Mai
Portland State University

Feng Liu
Portland State University, fliu@pdx.edu

Citation Details

Niklaus, S., Mai, L., & Liu, F. (2017). Video frame interpolation via adaptive separable convolution.

Video Frame Interpolation via Adaptive Separable Convolution

Simon Niklaus
Portland State University
sniklaus@pdx.edu

Long Mai
Portland State University
mtlong@cs.pdx.edu

Feng Liu
Portland State University
fliu@cs.pdx.edu

Abstract

Standard video frame interpolation methods first estimate optical flow between input frames and then synthesize an intermediate frame guided by motion. Recent approaches merge these two steps into a single convolution process by convolving input frames with spatially adaptive kernels that account for motion and re-sampling simultaneously. These methods require large kernels to handle large motion, which limits the number of pixels whose kernels can be estimated at once due to the large memory demand. To address this problem, this paper formulates frame interpolation as local separable convolution over input frames using pairs of 1D kernels. Compared to regular 2D kernels, the 1D kernels require significantly fewer parameters to be estimated. Our method develops a deep fully convolutional neural network that takes two input frames and estimates pairs of 1D kernels for all pixels simultaneously. Since our method is able to estimate kernels and synthesizes the whole video frame at once, it allows for the incorporation of perceptual loss to train the neural network to produce visually pleasing frames. This deep neural network is trained end-to-end using widely available video data without any human annotation. Both qualitative and quantitative experiments show that our method provides a practical solution to high-quality video frame interpolation.

1. Introduction

Traditional video frame interpolation methods estimate optical flow between input frames and synthesizing intermediate frames guided by optical flow [3]. However, their performance largely depends on the quality of optical flow, which is challenging to estimate accurately in regions with occlusion, blur, and abrupt brightness change.

Based on the observation that the ultimate goal of frame interpolation is to produce high-quality video frames and optical flow estimation is only an intermediate step, recent methods formulate frame interpolation [36] or extrapolation [14, 21, 58] as a convolution process. Specifi-

(a) Ground truth

(b) Niklaus *et al.* [36]

(c) Ours - $\mathcal{L}_1$
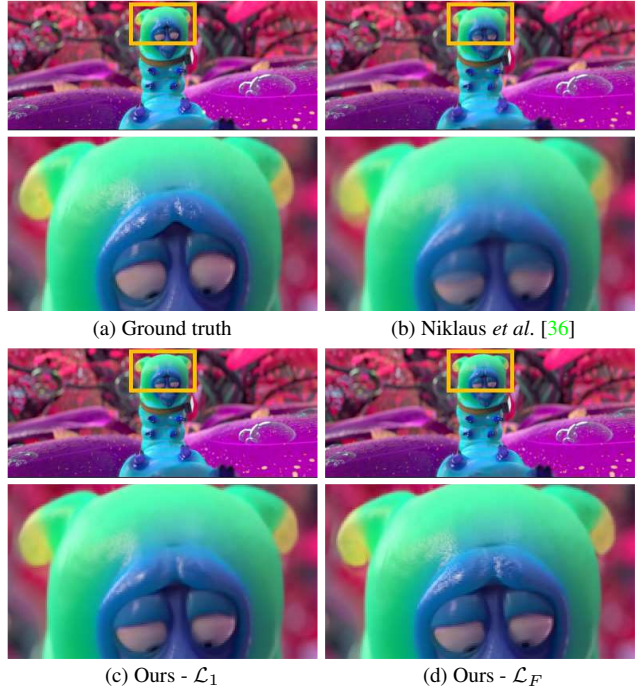
(d) Ours - $\mathcal{L}_F$

Figure 1: Video frame interpolation. Compared to the recent convolution approach that utilizes 2D kernels [36] (b), our separable convolution methods, especially the one with perceptual loss (d), incorporate 1D kernels that allow for full-frame interpolation and produce higher-quality results.

cally, they estimate spatially-adaptive convolution kernels for each output pixel and convolve the kernels with the input frames to generate a new frame. The convolution kernels jointly account for the two separate steps of motion estimation and re-sampling involved in traditional frame interpolation methods. In order to handle large motion, large kernels are required. For example, Niklaus *et al.* employ a neural network to output two 41×41 kernels for each output pixel [36]. To generate the kernels for all pixels in a 1080p video frame, the output kernels alone will require 26 GB of memory. The memory demand increases quadratically with the kernel size and thus limits the maximal motion to be handled. Given this limitation, Niklaus *et al.* trained a neural network to output the kernels pixel by pixel.

http://graphics.cs.pdx.edu/project/sepconv

This paper presents a spatially-adaptive separable convolution approach for video frame interpolation. Our work is inspired by the success of using separable filters to approximate full 2D filters for other computer vision tasks, like image structure extraction [41]. For frame synthesis, two 2D convolution kernels are required to generate an output pixel. Our approach approximates each of these with a pair of 1D kernels, one horizontal and one vertical. In this way, an $n \times n$ convolution kernel can be encoded using only $2n$ variables. This allows our method to employ a fully convolutional neural network that takes two video frames as input and produces the separable kernels for all output pixels at once. For a 1080p video frame, using separable kernels that approximate 41×41 ones only requires 1.27 GB instead of 26 GB of memory. Since our method is able to generate the full-frame output, we can incorporate perceptual loss functions [11, 22, 27, 42, 65] to further improve the visual quality of the interpolation results, as shown in Figure 1.

Our deep neural network is fully convolutional and can be trained end-to-end using widely available video data without any difficult-to-obtain meta data like optical flow. Our experiments show that our method is able to compare favorably to representative state-of-the-art interpolation methods both qualitatively and quantitatively on representative challenging scenarios and provides a practical solution to high-quality video frame interpolation.

2. Related Work

Video frame interpolation is a classic topic in computer vision and video processing. Common frame interpolation approaches estimate dense motion, typically optical flow, between two input frames and then interpolate one or more intermediate frames guided by the motion [3, 53, 60]. The performance of these methods often depends on optical flow and special care, such as flow interpolation, is necessary to handle problems with optical flow [3]. Generic image-based rendering algorithms can also be used to improve frame synthesis results [33, 66]. Different from optical flow based methods, Meyer *et al.* developed a phase-based interpolation method that represents motion in the phase shift of individual pixels and generates intermediate frames by per-pixel phase modification [35]. This phase-based method often produces impressive interpolation results; however, it sometimes cannot preserve high-frequency details in videos with large temporal changes.

Our research is inspired by the success of applying deep learning to optical flow estimation [2, 12, 16, 19, 50, 51, 52], artistic style transfer [17, 22, 28], and image enhancement [6, 9, 10, 46, 47, 55, 57, 62, 65]. Our work belongs to the category of research that employs deep neural networks for view synthesis. Some of these methods render unseen views from input images for objects like faces and chairs, instead of complex real-world scenes [13, 26,

49, 59]. Flynn *et al.* developed a method that generates a novel image by projecting input images onto multiple depth planes and combining these depth planes to create the novel view [15]. Kalantari *et al.* proposed a view expansion method for light field imaging that uses two sequential convolutional neural networks to model the disparity and color estimation steps of view interpolation and trained these two networks simultaneously [23]. Xie *et al.* developed a neural network that synthesizes an extra view from a monocular video to convert it to a stereo video [54].

Recently, Zhou *et al.* developed an method that employs a convolutional neural network to estimate appearance flow and then uses this estimation to warp input pixels to create a novel view [64]. Their method can warp individual input frames and blend them together to produce a frame between the input ones. The deep voxel flow approach, a concurrent work to our paper, developed a deep neural network to output dense voxel flows optimized frame interpolation results [29]. Long *et al.* also developed a convolutional neural network to interpolate a frame in between two input ones; however, their method generates the interpolated frame as an intermediate step to estimate optical flow [30].

Our method is most relevant to the recent frame interpolation [36] or extrapolation [14, 21, 58] methods that combine motion estimation and frame synthesis into a single convolution step. These methods estimate spatially-varying kernels for each output pixel and convolve them with input frames to generate a new frame. Since these convolution methods require large kernels to handle large motion, they cannot synthesize all the pixels for a high-resolution video simultaneously, limited by the available memory. For example, the method from Niklaus *et al.* interpolates frames pixel by pixel. Although they employed a shift-and-stitch strategy to generate multiple pixels in each pass, the number of pixels that can be synthesized simultaneously is still limited [36]. Other methods only generate a relatively low-resolution image. Our work extends these algorithms by estimating separable 1D kernels to approximate 2D kernels which significantly reduces the required amount of memory. Consequently, our method can interpolate a 1080p frame in one pass. Moreover, our method also supports the incorporation of perceptual loss functions, which need to be constrained on a continuous image region, to improve the visual quality of the interpolated frames.

3. Video Frame Interpolation

To make this paper self-complete, we first briefly describe the recent adaptive convolution approach to video frame interpolation [36] and define notations. We then describe how we develop our separable convolution-based frame interpolation method.

Our goal is to interpolate a frame $\hat{I}$ temporally in the middle of the two input video frames I_1 and I_2. For each
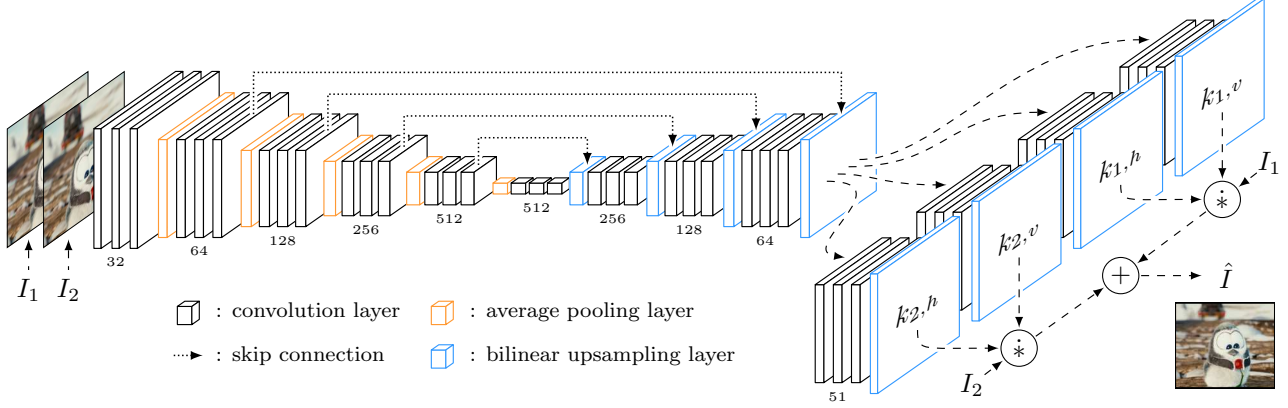
Figure 2: An overview of our neural network architecture. Given input frames I_1 and I_2, an encoder-decoder network extracts features that are given to four sub-networks that each estimate one of the four 1D kernels for each output pixel in a dense pixel-wise manner. The estimated pixel-dependent kernels are then convolved with the input frames to produce the interpolated frame $\hat{I}$. Note that $\dot{*}$ denotes a local convolution.

output pixel $\hat{I}(x, y)$, the convolution-based interpolation method estimates a pair of 2D convolution kernels $K_1(x, y)$ and $K_2(x, y)$ and uses them to convolve with I_1 and I_2 to compute the color of the output pixel as follows.

$$\hat{I}(x, y) = K_1(x, y) * P_1(x, y) + K_2(x, y) * P_2(x, y) \quad (1)$$

where $P_1(x, y)$ and $P_2(x, y)$ are the patches centered at (x, y) in I_1 and I_2. The pixel-dependent kernels K_1 and K_2 capture both motion and re-sampling information required for interpolation. To capture large motion, large-size kernels are required. Niklaus *et al.* used 41×41 kernels [36] and it is difficult to estimate them at once for all the pixels of a high-resolution frame simultaneously, due to the large amount of parameters and the limited memory. Their method thus estimates each individual pair of kernels pixel by pixel using a deep convolutional neural network.

Our method addresses this problem by estimating a pair of 1D kernels that approximate a 2D kernel. That is, we estimate $\langle k_{1,v}, k_{1,h} \rangle$ and $\langle k_{2,v}, k_{2,h} \rangle$ to approximate K_1 as $k_{1,v} * k_{1,h}$ and K_2 as $k_{2,v} * k_{2,h}$. Thus, our method reduces the number of kernel parameters from n^2 to $2n$ for each kernel. This enables the synthesis of a high-resolution video frame in one pass and the incorporation of perceptual loss to further improve the visual quality of the interpolation results, as detailed in the following subsections.

3.1. Separable kernel estimation

We design a fully convolutional neural network that given input frames I_1 and I_2, estimates two pairs of 1D kernels $\langle k_{1,v}, k_{1,h} \rangle$ and $\langle k_{2,v}, k_{2,h} \rangle$ for each pixel in the output frame $\hat{I}$, as illustrated in Figure 2. We treat each color channel equally and apply the same 1D kernels to each of the RGB channels to synthesize the output pixel. Note that applying the estimated kernels to the input images is a local convolution and we implement it as a network layer of
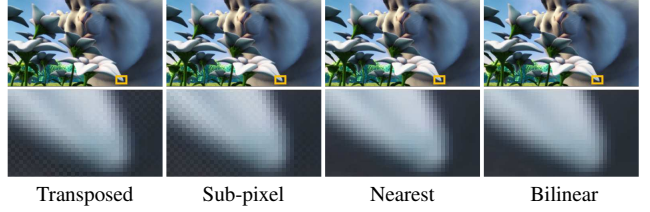

Figure 3: Checkerboard artifacts.

our neural network similar to a position-varying dynamics convolution layer in recent work [14, 21, 58]; therefore our neural network is end-to-end trainable.

Our neural network consists of a contracting component to extract features and an expanding part that incorporates upsampling layers to perform the dense prediction. We furthermore use skip connections [5, 31] to let the expanding layers incorporate features from the contracting part of the neural network, as shown in Figure 2. To estimate four sets of 1D kernels, we direct the information flow in the last expansion layer into four sub-networks, with each sub-network estimating one of the kernels. We could have modeled this jointly with a combined representation of the four kernels as well; however, we noticed a faster convergence during training when using four sub-networks.

We found stacks of 3×3 convolution layers together with Rectified Linear Units to be effective. Like Zhao *et al.* [63], we noticed that the average pooling performs well in the context of pixel-wise predictions and used it in our network accordingly. The upsampling layers in the expanding part can be implemented in various ways, such as transposed convolution, sub-pixel convolution, nearest-neighbor, and bilinear interpolation [10, 43, 61]. Odena *et al.* reported that checkerboard artifacts can occur for image generation tasks if the upsampling layers are not well selected [37]. Interestingly, while our method generates images by first estimating convolution kernels, these artifacts can still occur, as shown

3

in Figure 3. We followed the suggestions of Odena *et al.* and handled these artifacts by using bilinear interpolation to perform the upsampling in the decoder of our network.

Loss function. We consider two types of loss functions that measure the difference between an interpolated frame $\hat{I}$ and its corresponding ground truth I_{gt}. The first one is ℓ_1 norm based on per-pixel color difference, as defined below.

$$\mathcal{L}_1 = \left\| \hat{I} - I_{gt} \right\|_1 \tag{2}$$

Alternatively the ℓ_2 norm can be used; however, we found it often leads to blurry results, as also reported in other image generation tasks [18, 30, 34, 39, 45].

The second type of loss functions that this work explores is perceptual loss, which has often been found effective in producing visually pleasing images [11, 22, 27, 42, 65]. Perceptual loss is usually based on high-level features of images and is defined as follows.

$$\mathcal{L}_F = \left\| \phi(\hat{I}) - \phi(I_{gt}) \right\|_2^2 \tag{3}$$

where ϕ extracts features from an image. We tried various loss functions based on different feature extractors, such as SSIM loss [40] and feature reconstruction loss [22]. We empirically found that the feature reconstruction loss based on the `relu4_4` layer of the VGG-19 network [44] produces good results for our frame interpolation task.

3.2. Training

We initialized our neural network parameters using a convolution aware initialization method [1] and trained it using AdaMax [25] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of 0.001 and a mini-batch size of 16 samples. We chose a small mini-batch size since we experienced a degradation in the quality of the trained model as described by Keskar *et al.* [24] when using more samples per mini-batch. We used patches of size 128×128 instead of training on entire frames. This allows us to avoid patches that contain no useful information and leads to diverse mini-batches, which as described by Bansal *et al.* [4] improves training.

Training dataset. We extracted training samples from widely available videos, where each training sample consists of three consecutive frames with the middle frame serving as the ground truth. Since the video quality has a great influence on the quality of the trained model, we acquired video material from selected YouTube channels such as "Tom Scott", "Casey Neistat", "Linus Tech Tips", and "Austin Evans", whose videos consistently have a high-quality. Note that we downloaded these videos with a resolution of 1920×1080 but scale them to 1280×720 in order to reduce the influence of video compression.

Following Niklaus *et al.* [36], we did not use samples that span across video shot boundaries and discarded samples with a lack of texture. To increase the diversity of our training dataset, we avoided samples that are temporally close to each other. Instead of using the full frames, we randomly cropped 150×150 patches and selected those with sufficiently large motion. To compute the motion in each sample, we estimated the mean optical flow between the first and the last patch using SimpleFlow [48].

We composed our dataset from the extracted samples by randomly selecting $250,000$ of them without replacement. The random selection was guided by the estimated mean optical flow, making sure that samples with a large flow magnitude were more likely to be included. Overall, 10% of the pixels in the resulting training dataset have a flow magnitude of at least 17 pixels and 5% of them have a magnitude of at least 23 pixels. The largest motion is 39 pixels.

Data augmentation. We performed data augmentation on the fly during training. While each sample in the training dataset is of size 150×150 pixels, we used patches with a size of 128×128 pixels for training. This makes it possible to perform data augmentation by random cropping, preventing the network from learning spatial priors that potentially exist in the training dataset. We augmented the motion of each sample by shifting the cropped windows in the first and last frames while leaving the cropped window of the ground truth unchanged. By doing this systematically and shifting the cropped windows of the first and last frames in opposite directions, the ground truth will still be sound. We found that performing shifts by up to 6 pixels works well, which augments the flow magnitude by approximately 8.5 pixels. We also randomly flipped the cropped patches horizontally or vertically and randomly swap their temporal order, which makes the motion within the training dataset symmetric and prevents the network from being biased.

3.3. Implementation details

Below we discuss implementation details with respect to speed, boundary handling, and hyper-parameter selection.

Computational efficiency. We used Torch [8] to implement our convolutional neural network. To achieve a high computational efficiency and allow our network to directly render the interpolated frame, we wrote our own layer in CUDA that applies the estimated 1D kernels. If applicable, we used implementations based on cuDNN [7] for the other layers of the network to further improve the speed. With a Nvidia Titan X (Pascal), our system is able to interpolate a 1280×720 frame in 0.5 seconds as well as a 1920×1080 frame in 0.9 seconds. Training our network takes about 20 hours using four Nvidia Titan X (Pascal).

Boundary handling. Due to the utilized convolution-based interpolation formulation, the input needs to be padded such that boundary pixels can be processed. We tried zero padding, reflective padding, and padding by repetition. Empirically, we found padding by repetition to work well and

used it accordingly. Note that boundary handling is not needed during training, where an output with a reduced size is still acceptable.

Hyper-parameter selection. We used a validation dataset in order to select reasonable hyper-parameters for our network architecture as well as for the training. This validation dataset is disjoint from the training dataset but has been created in a similar manner.

Besides common parameters such as the learning rate, our model comes with a crucial domain-specific hyper-parameter, which is the size of the 1D kernels for interpolation. We found kernels of size 51 pixels to work well, which we attribute to the largest flow magnitude in the dataset, 39 pixels, together with 8.5-pixels of extra motion from augmentation. While increasing the kernel size is desirable to handle larger motion, restricted by the flow in our dataset, we did not observe improvements with larger kernels.

Another important hyper-parameter for our method is the number of pooling layers. Pooling layers have a great influence on the receptive field [32] of a convolutional neural network, which in our context is related to the aperture problem in motion estimation. A larger number of pooling layers increases the receptive field to potentially handle large motion; on the other hand, the largest flow magnitude in the training dataset provides an upper bound for the number of useful pooling layers. Empirically, we found using five pooling layers produces good interpolation results.

4. Experiments

We compare our method with representative state-of-the-art methods and evaluate them both qualitatively and quantitatively. For the optical flow based methods, we selected MDP-Flow2 [56], which currently achieves the lowest interpolation error in the Middlebury benchmark and Deep-Flow2 [52], which is the neural network based approach with the lowest interpolation error [3]. We follow recent frame interpolation papers [29, 35] and use the algorithm from the Middlebury benchmark [3] to synthesize frames from the estimated optical flow. We also compare our method with the phase-based frame interpolation method from Meyer *et al.* [35] as well as the AdaConv method based on adaptive convolution from Niklaus *et al.* [36] as alternatives to optical flow based methods. For all these methods, we use the code or trained models from the original papers. Please refer to our video for more results.

4.1. Loss functions

Our method incorporates two types of loss functions: $\mathcal{L}_1$ loss and feature reconstruction loss $\mathcal{L}_F$. To examine their effect, we trained two versions of our neural network. For the first one, we only used $\mathcal{L}_1$ loss and refer to this network as "$\mathcal{L}_1$" for simplicity in this paper. For the second one, we
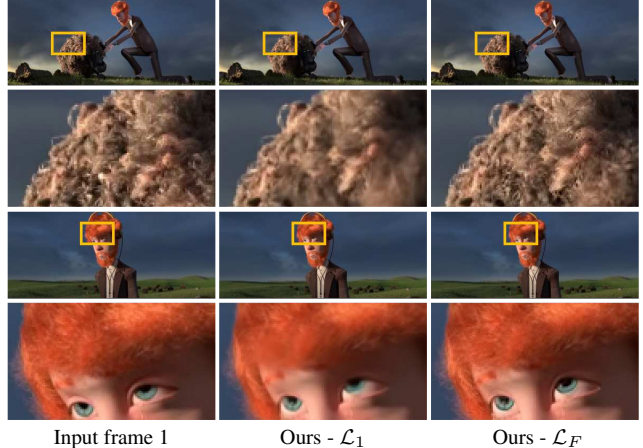

| Input frame 1 | Ours - $\mathcal{L}_1$ | Ours - $\mathcal{L}_F$ |

Figure 4: The effect of loss functions.

used both $\mathcal{L}_1$ loss and $\mathcal{L}_F$ loss and refer to this network as "$\mathcal{L}_F$" for simplicity. We tried different training schemes, including using linear combinations of $\mathcal{L}_1$ and $\mathcal{L}_F$ with different weights, and first training the network with $\mathcal{L}_1$ loss and then fine tuning it using $\mathcal{L}_F$ loss. We found that the latter leads to the best visual quality and used this scheme accordingly. As shown in Figure 4, incorporating $\mathcal{L}_F$ loss leads to sharper images with more high frequency details. This is in line with the findings in recent work on image generation and super resolution [11, 22, 27, 42, 65].

4.2. Visual comparison

We examine how our separable convolution approach handles challenging cases of video frame interpolation.

The top row in Figure 5 shows an example where the delicate butterfly leg makes it difficult to estimate optical flow accurately, causing artifacts in the flow-based results. Since the leg motion is also large, the phase-based approach cannot handle it well either and produces ghosting artifacts. The result from AdaConv appears blurry. Both our results are sharp and free from ghosting artifacts.

The second row shows an example of a busy street. As people are moving in opposing directions, there is significant occlusion. Both our methods handle occlusion better than the others. We attribute this to the convolution approach and the use of 1D kernels with fewer parameters.

In the third row, we show an example of a stage where the rightmost spotlight is being turned on. This violates the brightness constancy assumption of optical flow methods, leading to visible artifacts in the frame interpolation results. The last row shows an example with shallow depth of field, which is common in professional videos. The blurry background makes flow estimation difficult and compromises the flow-based frame interpolation results. For these examples, the other methods, including ours, work well.

Kernels. We examine how the kernels estimated by our $\mathcal{L}_F$ method compare to those from AdaConv. We show some
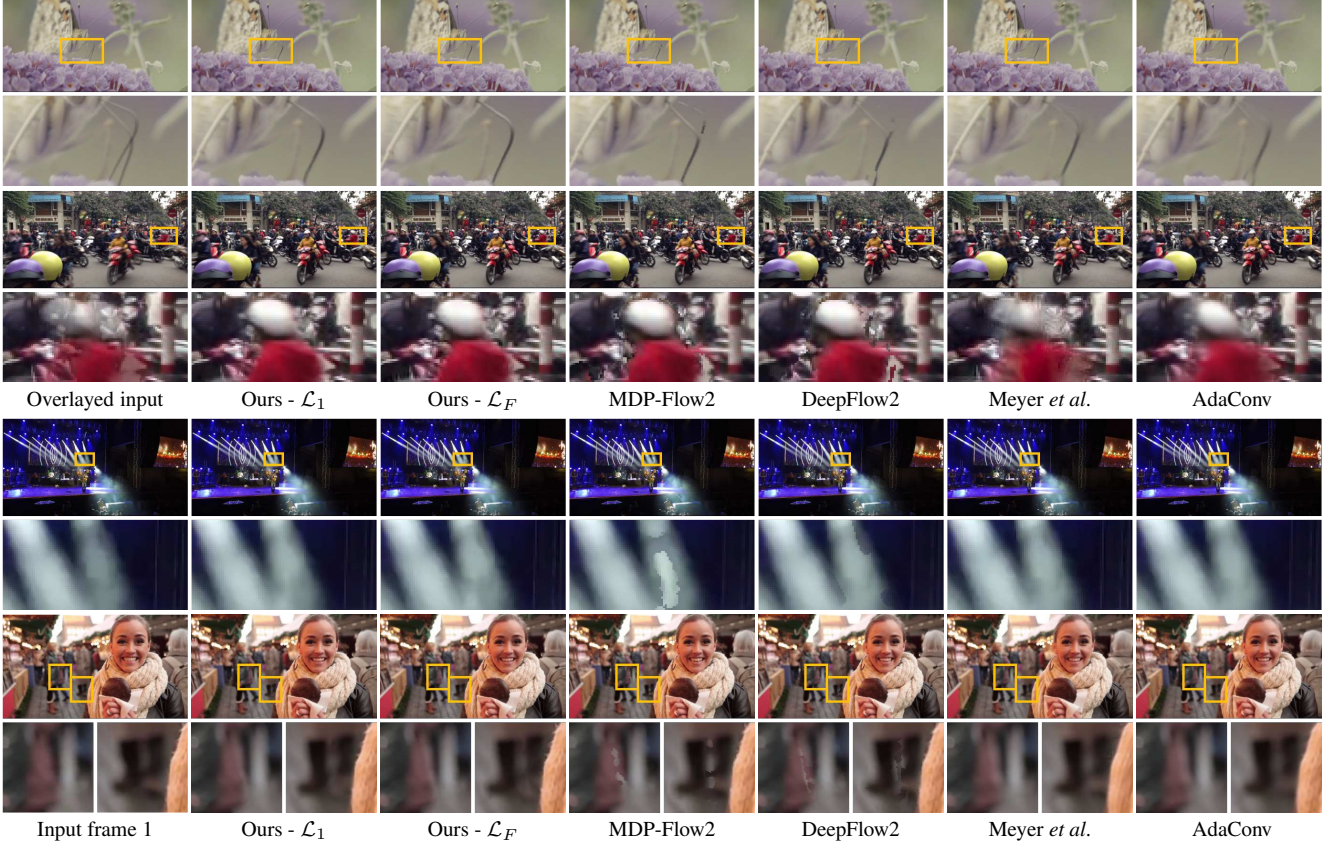
Overlayed input	Ours - $\mathcal{L}_1$	Ours - $\mathcal{L}_F$	MDP-Flow2	DeepFlow2	Meyer *et al.*	AdaConv

Input frame 1	Ours - $\mathcal{L}_1$	Ours - $\mathcal{L}_F$	MDP-Flow2	DeepFlow2	Meyer *et al.*	AdaConv

Figure 5: Visual comparison among frame interpolation methods.

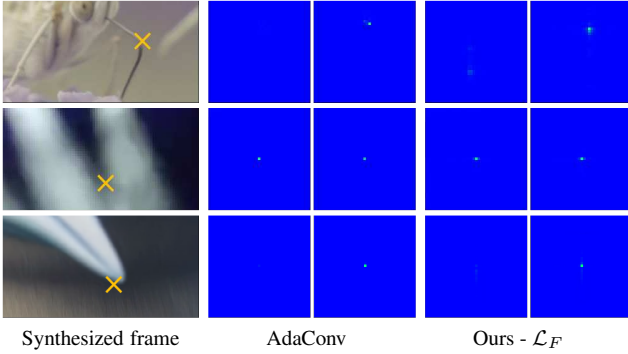| Synthesized frame	AdaConv	Ours - $\mathcal{L}_F$

Figure 6: Comparison of the estimated kernels.

representative kernels in Figure 6. Note that we convolve each pair of 1D kernels from our method to produce its equivalent 2D kernel for comparison. As our kernels are larger than those from AdaConv, we cropped the boundary values off for better visualization as they are all zeros.

In the butterfly example, we show the kernels for a pixel on the leg. AdaConv only takes color from the corresponding pixel in the second input frame. While our method takes color mainly from the same pixel in the second input, it also takes color from the corresponding pixel in the first input frame. Since the color of that pixel remains the same in the two input frames, both methods produce proper results.

Notice how both methods capture the motion encoded as the offset of the non-zero kernel values to the kernel center.

The second example shows kernels for a pixel in the lit area where the brightness changes between two input frames. Both methods output the same kernels that, due to the lack of motion, only have non-zero values in the center. Therefore, the output color is estimated as the average color of the corresponding pixels in the input frames.

The last example shows a pixel in an occluded area due to the leaf moving up. This area is only visible in the second input frame and both methods produce kernels that correctly choose to only sample from the second frame. They thus produce good results and are able to handle occlusion appropriately, unlike methods that explicitly have to establish a correspondence between pixels of the input frames.

4.3. Quantitative evaluation

We quantitatively evaluate our method on the interpolation set of the Middlebury optical flow benchmark [3]. Note that we did not fine-tune our models in any way. The results are shown in Table 1. Our $\mathcal{L}_1$ model and our $\mathcal{L}_F$ model perform particularly well in the regions with discontinuous motion. In terms of overall average, our $\mathcal{L}_1$ model achieves state-of-the-art results. Notice that our $\mathcal{L}_F$ model performs inferior to our $\mathcal{L}_1$ model in this quantitative evaluation due

	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.
Ours - $\mathcal{L}_1$	**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
Ours - $\mathcal{L}_F$	2.60	5.00	1.19	3.87	5.50	2.07	4.38	4.29	2.73	5.78	7.16	3.94	**10.1**	**12.7**	3.39	5.98	11.4	2.42	**6.85**	**15.5**	1.78	6.90	10.8	1.65	5.81	9.04	2.40
MDP-Flow2	2.89	5.38	1.19	**3.47**	5.07	**1.26**	3.66	6.10	2.48	**5.20**	7.48	**3.14**	10.2	12.8	3.61	6.13	11.8	2.31	7.36	16.8	**1.49**	7.75	12.1	1.69	5.83	9.69	2.15
DeepFlow2	2.99	5.65	1.22	3.88	5.79	1.48	**3.62**	6.03	**1.34**	5.38	7.44	3.22	11.0	13.8	3.67	5.83	11.2	2.25	7.60	17.4	1.50	7.82	12.2	1.77	6.02	9.94	**2.06**
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

Table 1: Evaluation on the Middlebury benchmark. *disc.*: regions with discontinuous motion. *unt.*: textureless regions.

	Cross-validation				Video: See You Again			
	$250,000$ samples at 150×150				$2,801$ samples at 960×540			
	MAE	RMSE	PSNR	SSIM	MAE	RMSE	PSNR	SSIM
Ours - $\mathcal{L}_1$	**3.66**	**7.37**	**32.92**	**0.941**	**2.03**	**4.28**	**41.31**	**0.968**
Ours - $\mathcal{L}_F$	4.01	7.84	32.37	0.934	2.11	4.40	40.88	0.965
MDP-Flow2	3.72	7.40	32.47	0.940	2.21	5.01	40.50	0.961
DeepFlow2	3.89	7.82	32.16	0.935	2.09	4.83	40.52	0.965
Meyer *et al.*	10.45	17.16	26.05	0.705	2.60	5.36	38.17	0.944
AdaConv	5.34	10.14	30.16	0.885	2.14	4.44	40.06	0.967

Table 2: More extensive quantitative evaluation.

to its loss function that optimizes for perceptual quality.

For a more extensive quantitative evaluation, we performed a cross-validation and additionally assessed the interpolation capabilities of the different methods on a popular video. The results are shown in Table 2. For the former, we performed a 10-fold cross-validation on our training dataset for both of our methods and let the other methods directly interpolate the $250,000$ samples that each have a resolution of 150×150 pixels. Please note that this experiment is mainly to evaluate how our method can generalize. We did not adjust the parameters of the other methods or fine-tune them, which might limit their performance in this cross-validation experiment and we included them as baselines. For the latter, we obtained the video "See You Again" from Wiz Khalifa which currently is the most viewed video on YouTube. We processed this video at a size of 960×540 since this resolution is the largest that all methods and their reference implementations support. We withheld every other frame and used the remaining frames to interpolate the withheld ones. In this way, every method interpolated $2,801$ frames. Across these two additional experiments, our $\mathcal{L}_1$ model performs best regardless of the incorporated error metric. Like in the evaluation on the Middlebury benchmark, our $\mathcal{L}_F$ model quantitatively performs inferior to our $\mathcal{L}_1$ model due to the nature of the different loss functions that they were optimized with.

4.4. User study

We conducted a user study to further compare the visual quality of the frame interpolation results from our $\mathcal{L}_F$ method with our $\mathcal{L}_1$ method as well as the other four methods. We recruited 15 participants, who are graduate or undergraduate students in Computer Science and Statistics. This study used all 8 examples of the Middlebury testing set. On each example, our $\mathcal{L}_F$ result was compared to the other 5 results one by one. In this way, each participant

compared 40 pairs of results. We developed a web-based system for the study. In each trial, the website only shows one result and supports participants to switch back and forth between two results using the arrow keys on the keyboard, allowing them to easily examine the difference between the results. The participants were asked to select the better result for each trial. The temporal order as well as the order in which the two results appear were randomized.

Figure 7 shows the result of this study. For each hypothesis that users prefer the frames interpolated by our $\mathcal{L}_F$ method over those produced by one of the baselines, we get a p-value < 0.01 and can thus confirm them. Note that the participants preferred our $\mathcal{L}_1$ result over our $\mathcal{L}_F$ result on the Basketball example, shown in Figure 8. We attribute this to the introduced discontinuity to the basketball.

4.5. Comparison with AdaConv

Our method builds upon AdaConv [36] by estimating 1D kernels instead of 2D kernels and developing a dedicated encoder-decoder neural network to estimate the kernels for all the pixels in a frame at once. This provides a few advantages. First, our method is over 20 times faster than AdaConv when interpolating a 1080p video. Second, as shown in the previous quantitative comparisons (Table 1 and Table 2), our method produces numerically better results. Third, our methods, especially $\mathcal{L}_F$, often generates visually more appealing results than AdaConv as shown in Figure 5, 10, and in our study. We attribute these advantages to the separable convolution. First, it allows us to synthesize the full frame at once and to use perceptual loss that has recently been shown effective in producing visually pleasing results [11, 22, 27, 42, 65]. Second, 1D kernels require significantly fewer parameters, which enforces a useful constraint to obtain good kernels. Third, our method is able to use a larger kernel than AdaConv and can thus handle larger motion. As shown in Figure 9, AdaConv cannot capture the motion of the cars and generates blurry results.

4.6. Discussion

By using different loss functions, we effectively optimized our model for different goals. While our $\mathcal{L}_1$ approach is able to provide better numerical results as reported in the quantitative evaluation in Table 1 and 2, our $\mathcal{L}_F$ approach achieves higher visual quality as shown in the user study where perceptual quality has been evaluated.
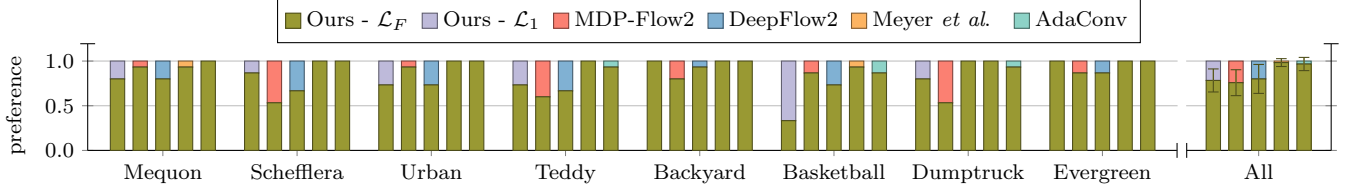
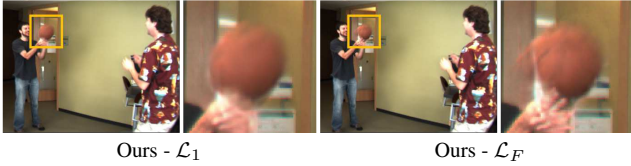Figure 7: User study result. The error bars denote the standard deviation.

Ours - $\mathcal{L}_1$ Ours - $\mathcal{L}_F$

Figure 8: Example where users prefer our $\mathcal{L}_1$ result.

AdaConv   Ours - $\mathcal{L}_F$ Direct - $\mathcal{L}_F$

Figure 9: Comparison with AdaConv and direct synthesis.

Overlayed input   Long *et al.*
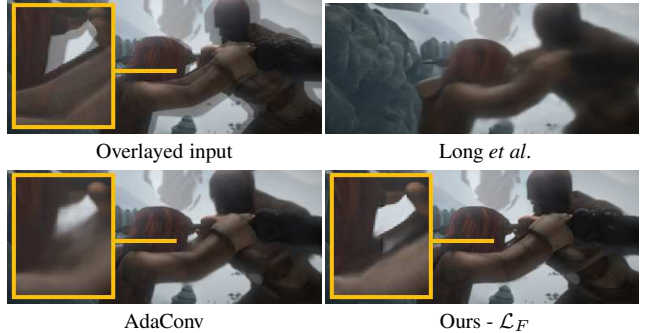

AdaConv   Ours - $\mathcal{L}_F$

Figure 10: Comparison with Long *et al.* [30].

tinue synthesizing frames at $t = 0.25$ and $t = 0.75$, or train a new model from scratch that returns frames at a different temporal offset. Both of these solutions are not ideal and are not as flexible as optical flow based interpolation. In the future, we plan to enhance our neural network to explicitly handle the temporal offset as a control variable.

5. Conclusion

This paper presents a practical solution to high-quality video frame interpolation. The presented method combines motion estimation and frame synthesis into a single convolution process by estimating spatially-adaptive separable kernels for each output pixel and convolving input frames with them to render the intermediate frame. The key to make this convolution approach practical is to use 1D kernels to approximate full 2D ones. The use of 1D kernels significantly reduces the number of kernel parameters and enables full-frame synthesis, which in turn supports the use of perceptual loss to further improve the visual quality of the interpolation results. Our experiments show that our method compares favorably to state-of-the-art interpolation results both quantitatively and qualitatively and produces high-quality frame interpolation results.

One question that has so far been left unanswered is how interpolation via separable convolution compares to directly synthesizing frames using a neural network. Therefore, we adapted our network in order to obtain a baseline for direct synthesis. Specifically, we used one sub-network after the encoder-decoder and let it directly estimate the interpolated frame instead of the kernel coefficients. We furthermore added Batch Normalization [20] layers after each block of convolution layers, which improves the quality of this direct synthesis network. We trained this model in the same way we trained our $\mathcal{L}_F$ method. As shown in Figure 9, the direct synthesis leads to blurry results. Additionally, we compare our approach with the image matching method from Long *et al.* [30] that performs direct synthesis to produce a middle frame as an intermediate result. As shown in Figure 10, our result is sharper. This is consistent with the findings in Zhou *et al.* [64] where they argue that synthesizing images from scratch is difficult.

The amount of motion that our method can handle is limited by the kernel size, which is 51 pixels in our system. While this is larger than the recent AdaConv method [36], we plan to handle even larger motion by borrowing a multi-scale approach from optical flow research [38].

Like AdaConv, our approach currently interpolates a frame at $t = 0.5$ in the middle of the two input frames. We cannot produce a frame at an arbitrary time between the input ones. To address this, we could either recursively con-

References

[1] A. Aghajanyan. Convolution aware initialization. *arXiv/1702.06295*, 2017. 4

[2] C. Bailer, B. Taetz, and D. Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*, pages 4015–4023, 2015. 2

[3] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 1, 2, 5, 6

[4] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. PixelNet: Representation of the pixels, by the pixels, and for the pixels. *arXiv/1702.06506*, 2017. 4

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. 3

[6] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399, 2012. 2

[7] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cuDNN: Efficient primitives for deep learning. *arXiv/1410.0759*, 2014. 4

[8] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 4

[9] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, pages 576–584, 2015. 2

[10] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016. 2, 3

[11] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016. 2, 4, 5, 7

[12] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 2

[13] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015. 2

[14] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, pages 64–72, 2016. 1, 2, 3

[15] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016. 2

[16] D. Gadot and L. Wolf. PatchBatch: A batch augmented loss for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4236–4245, 2016. 2

[17] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016. 2

[18] R. Goroshin, M. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, pages 1234–1242, 2015. 4

[19] F. Güney and A. Geiger. Deep discrete flow. In *Asian Conference on Computer Vision*, volume 10114, pages 207–224, 2016. 2

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, volume 37, pages 448–456, 2015. 8

[21] X. Jia, B. D. Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016. 1, 2, 3

[22] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, volume 9906, pages 694–711, 2016. 2, 4, 5, 7

[23] N. K. Kalantari, T. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 35(6):193:1–193:10, 2016. 2

[24] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv/1609.04836*, 2016. 4

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 4

[26] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. 2

[27] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv/1609.04802*, 2016. 2, 4, 5, 7

[28] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, volume 9907, pages 702–716, 2016. 2

[29] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. *arXiv/1702.02463*, 2017. 2, 5

[30] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, volume 9910, pages 434–450, 2016. 2, 4, 8

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 3

[32] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 4898–4906, 2016. 5

[33] D. Mahajan, F. Huang, W. Matusik, R. Ramamoorthi, and P. N. Belhumeur. Moving gradients: A path-based method

for plausible image interpolation. *ACM Trans. Graph.*, 28(3):42:1–42:11, 2009. 2

[34] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations*, 2016. 4

[35] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1418, 2015. 2, 5

[36] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017. 1, 2, 3, 4, 5, 7, 8

[37] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. http://distill.pub/2016/deconv-checkerboard. 3

[38] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv/1611.00850*, 2016. 8

[39] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv/1412.6604*, 2014. 4

[40] K. Ridgeway, J. Snell, B. Roads, R. S. Zemel, and M. C. Mozer. Learning to generate images with perceptual similarity metrics. *arXiv/1511.06409*, 2015. 4

[41] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning separable filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2754–2761, 2013. 2

[42] M. S. M. Sajjadi, B. Schölkopf, and M. Hirsch. EnhanceNet: Single image super-resolution through automated texture synthesis. *arXiv/1612.07919*, 2016. 2, 4, 5, 7

[43] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 3

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv/1409.1556*, 2014. 4

[45] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning*, volume 37, pages 843–852, 2015. 4

[46] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 769–777, 2015. 2

[47] P. Svoboda, M. Hradis, D. Barina, and P. Zemcík. Compression artifacts removal using convolutional neural networks. *arXiv/1605.00366*, 2016. 2

[48] M. W. Tao, J. Bai, P. Kohli, and S. Paris. SimpleFlow: A non-iterative, sublinear optical flow algorithm. *Computer Graphics Forum*, 31(2):345–353, 2012. 4

[49] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3D models from single images with a convolutional network. In *European Conference on Computer Vision*, volume 9911, pages 322–337, 2016. 2

[50] D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. *arXiv/1601.07532*, 2016. 2

[51] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep End2End Voxel2Voxel prediction. In *CVPR Workshops*, pages 402–409, 2016. 2

[52] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 2, 5

[53] M. Werlberger, T. Pock, M. Unger, and H. Bischof. Optical flow guided TV-L 1 video interpolation and restoration. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 6819, pages 273–286, 2011. 2

[54] J. Xie, R. B. Girshick, and A. Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, volume 9908, pages 842–857, 2016. 2

[55] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 350–358, 2012. 2

[56] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012. 5

[57] L. Xu, J. S. J. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798, 2014. 2

[58] T. Xue, J. Wu, K. L. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2016. 1, 2, 3

[59] J. Yang, S. E. Reed, M. Yang, and H. Lee. Weakly-supervised disentangling with recurrent transformations for 3D view synthesis. In *Advances in Neural Information Processing Systems*, pages 1099–1107, 2015. 2

[60] Z. Yu, H. Li, Z. Wang, Z. Hu, and C. W. Chen. Multi-level video frame interpolation: Exploiting the interaction among different levels. *IEEE Trans. Circuits Syst. Video Techn.*, 23(7):1235–1248, 2013. 2

[61] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pages 2018–2025, 2011. 3

[62] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, volume 9907, pages 649–666, 2016. 2

[63] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *arXiv/1612.01105*, 2016. 3

[64] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, volume 9908, pages 286–301, 2016. 2, 8

[65] J. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, volume 9909, pages 597–613, 2016. 2, 4, 5, 7

[66] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. A. J. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004. 2