AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation

Hyeongmin Lee[1] Taeoh Kim[1] Tae-young Chung[1] Daehyun Pak[1] Yuseok Ban[2] Sangyoun Lee[1*]

[1]Yonsei University
{minimonia,kto,tato0220,koasing,syleee}@yonsei.ac.kr

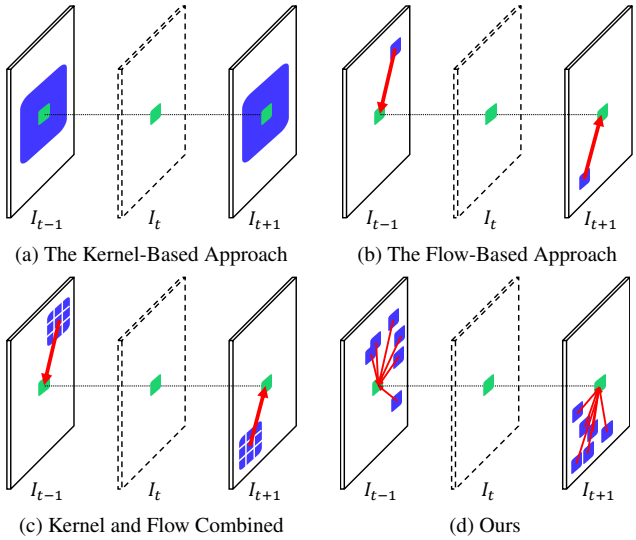
[2]Agency for Defense Development
ban@add.re.kr

Abstract

Video frame interpolation is one of the most challenging tasks in video processing research. Recently, many studies based on deep learning have been suggested. Most of these methods focus on finding locations with useful information to estimate each output pixel using their own frame warping operations. However, many of them have Degrees of Freedom (DoF) limitations and fail to deal with the complex motions found in real world videos. To solve this problem, we propose a new warping module named Adaptive Collaboration of Flows (AdaCoF). Our method estimates both kernel weights and offset vectors for each target pixel to synthesize the output frame. AdaCoF is one of the most generalized warping modules compared to other approaches, and covers most of them as special cases of it. Therefore, it can deal with a significantly wide domain of complex motions. To further improve our framework and synthesize more realistic outputs, we introduce dual-frame adversarial loss which is applicable only to video frame interpolation tasks. The experimental results show that our method outperforms the state-of-the-art methods for both fixed training set environments and the Middlebury benchmark. Our source code is available at https://github.com/HyeongminLEE/AdaCoF-pytorch.

1. Introduction

Synthesizing the intermediate frame when consecutive frames have been provided is one of the main research topics in the video processing area. Using a frame interpolation algorithm, we can obtain slow-motion videos from ordinary videos without using professional high-speed cameras. In addition, we can freely convert the frame rates of the videos so it can be applied to the video coding system. To interpolate the intermediate frame of a video requires an understanding of motion, unlike image pixel interpolation. Unfortunately, real world videos contain not only simple motions, but also large and complex ones, making the task significantly more difficult.



(a) The Kernel-Based Approach (b) The Flow-Based Approach

(c) Kernel and Flow Combined (d) Ours

Figure 1: Overall description of the main streams and our method. The blue parts of each figure represent the reference points for generating the target pixel.

Most of the approaches define video frame interpolation as a problem of finding reference locations in input frames which include information for estimating each output pixel value. This can be seen as a motion estimation process, because the task involves tracking the path of the target pixel. Therefore, each algorithm covers its own motion domain, and this area is directly related to the performance. To handle motion in real world videos, we need a generalized operation that can refer to *any number* of pixels in *any location* in the input frames. However, most of the existing approaches have a variety of limitations in Degrees of Freedom (DoF).

One is the kernel-based approach (Figure 1 (a)) [34, 35], which adaptively estimates the large-sized kernel for each pixel and synthesizes the intermediate frame by convolving the kernels with the input. This approach finds the proper reference location by assigning large weights to the pixels

of interest. However, it does not refer to *any location*, as it cannot deal with large motions beyond the kernel size. It is not efficient to keep the large size of the kernel even though the motion is small. The second approach is the flow-based approach (Figure 1 (b)) [20, 27], which estimates the flow vector directly pointing to the reference location for each output pixel. However, it cannot refer to *any number* of pixels because only one location is referred to in each input frame. Therefore, it is not suitable for complex motions and the result may suffer from lack of information when the input frame is of low-quality. Recently, methods of combining kernel-based and flow-based approaches are proposed to compensate for each other's limitations (Figure 1 (c)) [49, 3]. They multiply the kernels with the location pointed to by the flow vector. Therefore, they can refer to *any location* plus some additional neighboring pixels. However, this approach is not much different from the flow based approach as it uses significantly fewer reference points than the kernel-based one. In addition, there is room for improvement in terms of DoF because the shape of the kernel is a fixed square.

In this paper, we propose an operation that refers to *any number* of pixels and *any location* called Adaptive Collaboration of Flows (AdaCoF). To synthesize a target pixel, we estimate multiple flows, called offset vectors, pointing to the reference locations and sample them. Then the target pixel is obtained by linearly combining the sampled values. Our method is inspired by deformable convolution (DefConv) [8], but AdaCoF is significantly different from it in some points. First, DefConv has a shared weight for all positions, and it is not suitable for video because there are various motions in each position of a frame. Therefore, we allow the weights to be spatially adaptive. Second, AdaCoF is used as an independent module for frame warping, not for feature extraction as DefConv. Therefore, we obtain the weights as the outputs of a neural network, instead of training them as learnable parameters. Third, we add dilation for the starting point of the offset vectors to enforce the them to search a wider area. Lastly, we add an occlusion mask to utilize only one of the two input frames when one of the reference pixels is occluded. As shown in Figure 1 (d), it can refer to *any number* within *any location* in the input frames, because the sizes and shapes of the kernels are not fixed. Therefore, our method has the highest DoF compared to most of the other competitive algorithms, and therefore can deal with various complex motions in real world videos. To make the synthesized frames more realistic, we further train a discriminator to detect the generated frame given the output and one of the input frames. Then we train the generator to maximize the entropy of the discriminator using dual-frame adversarial loss. Experimental results on various benchmarks show the effectiveness of AdaCoF over the latest state-of-the-art approaches.

2. Related Work

Most of the classic video frame interpolation methods estimate the dense flow maps using optical flow algorithms [12, 19, 44, 46] and warp the input frames [1, 4, 47, 50]. Therefore, the performance of these approaches largely depends on optical flow algorithms. Also, optical flow based approaches have limitations in many cases, such as occlusions, large motion, and brightness changes. Although there are some approaches without using external optical flow modules [25, 29], they still have difficulty in dealing with these problems. Meyer *et al.* [32] regard video frames as linear combinations of wavelets with different directions and frequencies. This approach interpolates each wavelet's phase and magnitude. This method makes notable progress in both performance and running time. Their recent work also applies deep learning to this approach [31]. However, it still has limitations for large motions of high frequency components.

Recent work has demonstrated the success of applying deep learning in the field of computer vision [10, 14, 18, 21, 23, 41], which, in turn, inspires various deep learning based frame interpolation methods. As all we require for training neural networks are three consecutive video frames, learning based approaches are appropriate for this task. Long *et al.* [28] propose a CNN architecture that uses two input frames and directly estimates the intermediate frame. However, this type of approach often leads to blurry results. Some other methods focus on where to find the output pixel from the input frames, instead of directly estimating the image. This paradigm is based on the fact that at least one input frame contains the output pixel, even in the case of occlusion. Niklaus *et al.* [34] estimate a kernel for each location and obtains the output pixel by convolving it over input patches. Each kernel samples the proper input pixels by combining them selectively. However, this requires a lot of memory and estimating large kernels for every pixel is computationally expensive. Niklaus *et al.* [35] solve this problem by estimating each kernel from the outer product of two vectors. However, this approach cannot handle motions larger than the kernel size and it is still wasteful to estimate large kernels for small motions. Liu *et al.* [27] estimate a flow map that consists of vectors directly pointing to reference locations. They sample the proper pixels according to the flow map. However, as they assume that the forward and backward flows are the same, it is difficult to handle complex motions. Jiang *et al.* [20] propose a similar algorithm, but they estimate the forward and backward flows separately. They also improve the flow computation stage by defining the warping loss. However, it could be risky to get only one pixel value from each frame, especially when the input patches are of poor quality. To solve these problems, Reda *et al.* [38] and Bao *et al.* [3] combine kernel and flow map based approaches. They multiply

small-sized kernels with the locations pointed by the flow vectors. However, the reference points are still limited in a small area because the kernels maintain their square shape, which results in low DoF.

There are some approaches that use additional information to solve problems in video frame interpolation. Niklaus *et al.* [33] exploit the context informations extracted from ResNet-18 [18] to enable the informative interpolation and succeed in obtaining high-quality results. In addition, Bao *et al.* [2] use depth maps estimated from hourglass architecture [6] to solve the occlusion problems. Lastly, Liu *et al.* [26] obtain better performance with cycle consistency loss and additional edge maps. These approaches can be independently applied to many other algorithms, including our approach.

3. Proposed Approach

3.1. Video Frame Interpolation

Given consecutive video frames I_n and I_{n+1}, where $n \in \mathbb{Z}$ is a frame index, our goal is to find the intermediate frame I_{out}. All the information required to produce I_{out} can be obtained from I_n and I_{n+1}. Therefore, all we have to do is find the relations between them. We regard the relation as a warping operation $\mathcal{T}$ from I_n and I_{n+1} to I_{out}. For the forward and backward warping operations $\mathcal{T}_f$ and $\mathcal{T}_b$, we can consider I_{out} as a combination of $\mathcal{T}_f(I_n)$ and $\mathcal{T}_b(I_{n+1})$ as follows.

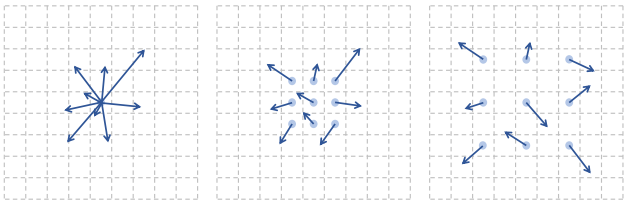
$$I_{out} = \mathcal{T}_f(I_n) + \mathcal{T}_b(I_{n+1}) \tag{1}$$

The frame interpolation task results in a problem of how the spatial transform $\mathcal{T}$ can be found. We employ a new operation called *Adaptive Collaboration of Flows (AdaCoF)* for $\mathcal{T}$, which convolve the input image with adaptive kernel weights and offset vectors for each output pixel.

Occlusion reasoning. Let both the input and output image sizes be $M \times N$. In the case of occlusion, the target pixel will not be visible in one of the input images. Therefore we define occlusion map $V \in [0, 1]^{M \times N}$ and modify Equation (1) as follows.

$$I_{out} = V \odot \mathcal{T}_f(I_n) + (J - V) \odot \mathcal{T}_b(I_{n+1}), \tag{2}$$

where $\odot$ is a pixel-wise multiplication and J is an $M \times N$ matrix of ones. For the target pixel (i, j), $V(i, j) = 1$ implies that the pixel is visible only in I_n and $V(i, j) = 0$ implies that it is visible only in I_{n+1}.



(a) $d = 0$ (b) $d = 1$ (c) $d = 2$

Figure 2: Illustration of the offset vectors of AdaCoF under various dilations.

3.2. Adaptive Collaboration of Flows

Let the frame warped from I be $\hat{I}$. When we define $\mathcal{T}$ as a classic convolution, we can write $\hat{I}$ as follows.

$$\hat{I}(i, j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l} I(i + k, j + l), \tag{3}$$

where F is the kernel size and $W_{k,l}$ are the kernel weights. The input image I is considered to be padded so that the original input and output size are equal. Deformable convolution [8] adds offset vectors $\Delta p_{k,l} = (\alpha_{k,l}, \beta_{k,l})$ to the classic convolution as follows.

$$\hat{I}(i, j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l} I(i + k + \alpha_{k,l}, j + l + \beta_{k,l}) \tag{4}$$

AdaCoF, unlike the classic deformable convolutions, does not share the kernel weights over the different pixels. Therefore the notation for the kernel weights $W_{k,l}$ should be written as follows.

$$\hat{I}(i, j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l}(i, j) I(i + k + \alpha_{k,l}, j + l + \beta_{k,l}) \tag{5}$$

The offset values $\alpha_{k,l}$ and $\beta_{k,l}$ may not be integer values. In other words, $(\alpha_{k,l}, \beta_{k,l})$ could point to an arbitrary location, not only the grid point. Therefore, the pixel value of I for any location has to be defined. We use bilinear interpolation to obtain the values of non-grid location as DCNs [8]. It also makes the module differentiable; therefore, the whole network can be trained end-to-end.

Dilation. We found that dilating the starting point of the offset vectors helps AdaCoF to explore wider area as shown in Figure 2. Therefore, we add dilation term $d \in \{0, 1, 2, ...\}$ to the operation as follows.

$$\hat{I}(i, j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l}(i, j) I(i + dk + \alpha_{k,l}, j + dl + \beta_{k,l}) \tag{6}$$

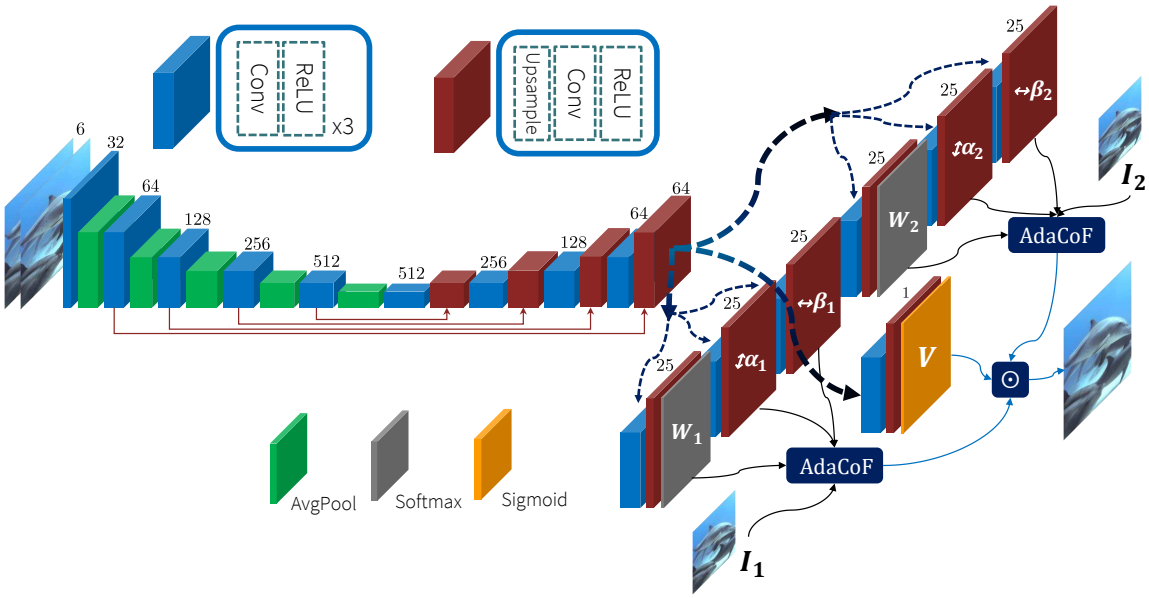


Figure 3: The neural network architecture. The model consists of three main parts: the U-Net, sub-networks, and Adaptive Collaboration of Flows (AdaCoF). The U-Net architecture extracts features from the input image. Then the sub-networks estimates the parameters needed for AdaCoF from the extracted features. The output's height and width of each sub-network are the same as that of the input. Each parameter group for an output pixel is obtained as a 1D vector along the channel axis. The AdaCoF part synthesizes the intermediate frame using the input frames and parameters.

3.3. Network Architecture

We design a fully convolutional neural network which estimates the kernel weights $W_{k,l}$, offset vectors $(\alpha_{k,l}, \beta_{k,l})$, and occlusion map V. Therefore, any video frames size can be used as the input. Furthermore, because each module of the neural network is differentiable, it is end-to-end trainable. Our neural network starts with the U-Net architecture, which consists of encoder, decoder, and skip connections [39]. Each processing unit basically contains 3×3 convolution and ReLU activation. For the encoder part, we use average pooling to extract the features. And for the decoder part, we use bilinear interpolation for the upsampling. After the U-Net architecture, the seven sub-networks finally estimate the outputs ($W_{k,l}$, $\alpha_{k,l}$, $\beta_{k,l}$ for each frame and V). We use sigmoid activation for V to satisfy $V \in [0,1]^{M \times N}$. Moreover, as the weights $W_{k,l}$ for each pixel have to be non-negative and must add up to 1, softmax layers are used for the constraints. More specific architectures of the network are described in Figure 3.

3.4. Objective Functions

Loss Function. First, we have to reduce a difference between the model output I_{out} and ground truth I_{gt}. We use ℓ_1 norm for the loss as follows.

$$\mathcal{L}_1 = \|I_{out} - I_{gt}\|_1 \tag{7}$$

The ℓ_2 norm can be used, but it is known that the ℓ_2 norm-based optimization leads to blurry results in most of the image synthesis tasks [16, 28, 30, 43]. Following Liu *et al.* [27], we use the Charbonnier Function $\Phi(x) = (x^2 + \epsilon^2)^{1/2}$ for optimizing ℓ_1 norm, where $\epsilon = 0.001$.

Perceptual Loss. Perceptual loss has been found to be effective in producing visually more realistic outputs [11, 21, 51]. We add the perceptual loss with the feature extractor $\mathcal{F}$ from `conv4_3` of ImageNet pretrained VGG16 network.

$$\mathcal{L}_{vgg} = \|\mathcal{F}(I_{out}) - \mathcal{F}(I_{gt})\|_2 \tag{8}$$

Dual-Frame Adversarial Loss. It is known that training the networks with adversarial loss [15] can lead to results of higher quality and sharpness, instead of increasing mean squared error [24, 5]. This could be applied to video frame interpolation tasks. However, simply applying it to the single output frame does not consider the temporal consistency and leads to a disparate result compared to the input frames. What we want is to make the synthesized frame appear natural among the adjacent frames, not the other real images. Therefore, we concatenate the generated frame and one of the input frames in the temporal order and train the discriminator C to distinguish which of the two is the generated frame with the following loss.

$$-\mathcal{L}_C = \log(C([I_n, I_{out}])) + \log(1 - C([I_{out}, I_{n+1}])), \tag{9}$$

	Middlebury		UCF101		DAVIS	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ours-*fb*	32.879	0.956	33.449	0.967	24.787	0.828
Ours-*kb*	34.762	0.972	34.689	0.973	25.802	0.854
Ours-*ws*	35.412	0.976	34.901	0.973	26.623	0.866
Ours-*woocc*	35.471	0.975	34.907	0.973	26.482	0.863
Ours-*sdc*	34.973	0.972	34.673	**0.974**	26.367	0.866
Ours-*vgg*	**35.694**	**0.977**	**34.973**	0.973	**26.773**	**0.869**
Ours	**35.715**	**0.978**	**35.063**	**0.974**	**26.636**	**0.868**

Table 1: Result of ablation study on warping operations.

where $[\cdot]$ is concatenation. Then we train the main network to maximize the uncertainty, i.e., entropy, of the discriminator with the following loss. This idea is inspired by some prior works [9, 13].

$$\begin{aligned}\mathcal{L}_{adv} =& C([I_n, I_{out}]) \log(C([I_n, I_{out}])) \\ &+ C([I_{out}, I_{n+1}]) \log(C([I_{out}, I_{n+1}]))\end{aligned} \tag{10}$$

Thus, the network is intended to generate an output that is realistic compared to the adjacent input frames.

We finally combine above losses to compose two versions of objective function: distortion-oriented loss ($\mathcal{L}_d$) and perception-oriented loss ($\mathcal{L}_p$) as follows.

$$\mathcal{L}_d = \mathcal{L}_1, \tag{11}$$

$$\mathcal{L}_p = \lambda_1 \mathcal{L}_1 + \lambda_{vgg} \mathcal{L}_{vgg} + \lambda_{adv} \mathcal{L}_{adv}, \tag{12}$$

For the perception-oriented version, we first train the network with $\mathcal{L}_d$ then fine-tune it with $\mathcal{L}_p$.

4. Experiments

4.1. Experimental Settings

Learning Strategy. We train our neural network using AdaMax optimizer [22], where $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate is initially 0.001 and decays half every 20 epochs. The batch size is 4 and the network is trained for 50 epochs.

Training Dataset. We use Vimeo90K [49] dataset for training. It contains 51,312 triplets of 256×448 video frames. To augment the dataset, we randomly crop 256×256 patches from the original images. We also eliminate the biases due to the priors by flipping horizontally, vertically and swapping the order of frames for the probability 0.5.

Computational issue. Our approach is implemented using PyTorch [36]. To implement the AdaCoF layer, we used CUDA and cuDNN [7] for the parallel processing. We set the kernel size 5×5 and all the weights, offsets and occlusion map require 0.94 GB of memory for a 1080p video frame. It is about 70% demand compared to Niklaus *et al.* [35]. Using RTX 2080 Ti GPU, it takes 0.21 seconds to synthesize a 1280×720 frame.

	Middlebury		UCF101		DAVIS	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$F = 1$	32.879	0.956	33.449	0.967	24.787	0.828
$F = 3$	35.212	0.975	34.728	0.973	26.535	0.867
$F = 5$	35.715	0.978	**35.063**	**0.974**	26.636	0.868
$F = 7$	35.927	0.979	34.974	**0.974**	**26.987**	**0.873**
$F = 9$	**36.019**	**0.980**	35.012	0.973	**27.029**	**0.875**
$F = 11$	**36.094**	**0.981**	**35.024**	**0.974**	26.941	**0.873**

Table 2: Experimental result on kernel size F.

	Middlebury		UCF101		DAVIS	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$d = 0$	35.489	0.977	35.032	**0.974**	**26.710**	**0.870**
$d = 1$	**35.715**	**0.978**	**35.063**	**0.974**	26.636	0.868
$d = 2$	**35.876**	**0.980**	**35.099**	**0.974**	**26.910**	**0.870**

Table 3: Experimental result on dilation d.

Evaluation settings. The test datasets used for the experiments are the Middlebury dataset [1], some randomly sampled sequences from UCF101 [42] and the DAVIS dataset [37]. We evaluate each algorithm by measuring PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity) [45] for all test datasets. For all the tables in this section, the **red** numbers mean the best performance and the **blue** numbers mean the second best performance.

4.2. Ablation Study

We analyze the contributions of each module in terms of five keywords: warping operation, perceptual loss, kernel size, dilation and adversarial loss.

Warping Operation. To verify that higher DoF leads to better performance, we fix the backbone network and replace AdaCoF with some other warping operations of lower DoF. We train all versions of warping operation with $\mathcal{L}_d$ and the kernel sizes are fixed to be 5 except for Ours-*fb*.

- Ours-*fb*: To compare AdaCoF with flow-based approaches, we set the kernel size to be 1.
- Ours-*kb*: SepConv [35] is one of the most representative kernel-based approaches. However, because it does not contain an occlusion map, the comparison is not fair. Therefore, we train a new network of SepConv with an occlusion map.
- Ours-*sdc*: To compare our algorithm with kernel and flow combined approaches, we exploit Spatially Displaced Convolution (SDC) [38] instead of AdaCoF.
- Ours-*ws*: One of the differences between deformable convolution and AdaCoF is that our algorithm does not share the weights over all locations of images. Therefore, we compare it with the weight shared version.

	AVERAGE		Mequon		Schefflera		Urban		Teddy		Backyard		Basketball		Dumptruck		Evergreen	
	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE
MDP-Flow2 [48]	5.83	0.87	2.89	0.59	3.47	0.62	3.66	1.24	5.20	0.94	10.20	0.98	6.13	1.09	7.36	0.70	7.75	0.78
DeepFlow [46]	5.97	0.86	2.98	0.62	3.88	0.74	3.62	0.86	5.39	0.99	11.00	1.04	5.91	1.02	7.14	0.63	7.80	0.96
SepConv [35]	5.61	0.83	2.52	**0.54**	3.56	0.67	4.17	1.07	5.41	1.03	10.20	0.99	5.47	0.96	6.88	0.68	6.63	0.70
SuperSlomo [20]	5.31	0.78	2.51	0.59	3.66	0.72	**2.91**	0.74	5.05	0.98	9.56	0.94	5.37	0.96	6.69	0.60	6.73	0.69
CtxSyn [33]	5.28	0.82	**2.24**	**0.50**	**2.96**	**0.55**	4.32	1.42	**4.21**	**0.87**	9.59	0.95	5.22	0.94	7.02	0.68	6.66	0.67
CyclicGen [26]	**4.20**	**0.73**	**2.26**	0.64	3.19	0.67	**2.76**	**0.72**	4.97	0.95	**8.00**	0.91	**3.36**	0.87	**4.55**	**0.53**	**4.48**	**0.52**
TOF-M [49]	5.49	0.84	2.54	0.55	3.70	0.72	3.43	0.92	5.05	0.96	9.84	0.97	5.34	0.98	6.88	0.72	7.14	0.90
DAIN [2]	4.86	**0.71**	2.38	0.58	3.28	0.60	3.32	**0.69**	**4.65**	**0.86**	**7.88**	**0.87**	4.73	0.85	6.36	0.59	6.25	0.66
MEMC-Net [3]	5.00	0.74	2.39	0.59	3.36	0.64	3.37	0.80	4.84	0.88	8.55	**0.88**	4.70	0.85	6.40	0.64	6.37	0.63
AdaCoF (Ours)	**4.75**	**0.73**	2.41	0.60	**3.10**	**0.59**	3.48	0.84	4.84	0.92	8.68	0.90	**4.13**	**0.84**	**5.77**	**0.58**	**5.60**	**0.57**

Table 4: Evaluation results on the Middlebury benchmark.

	Middlebury		UCF101		DAVIS	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Overlapping	27.968	0.879	30.445	0.935	21.922	0.740
Phase Based [32]	31.117	0.933	32.454	0.953	23.465	0.800
MIND [28]	31.346	0.943	32.437	0.963	25.570	0.852
SepConv [35]	35.521	0.977	34.735	0.973	26.258	0.861
DVF [27]	34.340	0.971	34.465	0.972	25.880	0.858
SuperSlomo [20]	34.234	0.972	34.055	0.970	25.699	0.858
Ours	**35.715**	**0.978**	**35.063**	**0.974**	**26.636**	**0.868**
Ours +	**36.139**	**0.981**	**35.048**	**0.974**	**27.070**	**0.874**

Table 5: Evaluation result with fixed train dataset.

- Ours-*woocc*: AdaCoF without occlusion map. The intermediate frame is obtained by simply averaging the outputs from the forward and backward warping.

As shown in Table 1, our warping operation outperforms the other ones with lower DoFs. Especially, we can find that the PSNR gap between Ours-*sdc* and Ours is larger than the gap between Ours-*kb* and Ours-*sdc*. It means that breaking the square-shaped kernels to be any shape is more crucial than allowing the kernels to move freely.

Perceptual Loss. We add perceptual loss $\mathcal{L}_{vgg}$ introduced in Section 3.4 without adversarial loss. We set $\lambda_{vgg} = 0.01$. The row of Ours-*vgg* in Table 1 shows that the PSNR generally decreases and increases only for DAVIS datasets. This implies that the perceptual loss improves the robustness for hard sequences with large and complex motions.

Kernel Size. We train the network with various kernel sizes $F \in \{1, 3, 5, 7, 9, 11\}$ which means that F^2 offset vectors are used. As shown in Table 2, the larger kernel size generally leads to better performance and the PSNR saturates as F increases. Especially, the saturation is earlier for the UCF101 dataset because it contains relatively small motion and low-resolution sequences so that there is no room for the performance increase.

Dilation. In Section 3.3, we add dilation to the AdaCoF operation to enforce the offset vectors to start from a wider area. We check the effect of dilation by training the network with $F = 5$ and $d \in \{0, 1, 2\}$. $d = 0$ means that the offset vectors start from the same location. Table 3 shows that the larger dilation generally leads to better results. As we can see from the 4$^{\text{th}}$ - 7$^{\text{th}}$ columns of Figure 6, the offset vectors tend to spread more in the case of large motion. Therefore, dilation provides the effect of *better initialization* for them. Figure 6 will be covered in more detail in Section 4.5.

(a) Ours-$\mathcal{L}_d$ (b) Ours-$\mathcal{L}_p$

(c) WGAN-GP (d) TGAN

Figure 4: The result of adding adversarial losses.

Adversarial Loss. For the visually more convincing results, we first train the network with $\mathcal{L}_d$ for 50 epochs and fine-tune it for 10 epochs with $\mathcal{L}_p$ which is introduced in Section 3.4. We set $\lambda_1 = 0.01, \lambda_{vgg} = 1, \lambda_{adv} = 0.005$. For the comparison, we train the version of changing $\mathcal{L}_{adv}$ to be WGAN-GP loss [17] and TGAN loss [40]. Then we visually compare them with the result of the proposed dual-frame adversarial loss (Ours-$\mathcal{L}_p$). According to Figure 4, fine-tuning the network with adversarial losses increase the sharpness of the results. However, WGAN-GP and TGAN loss cause some artifacts to the output image, while our loss preserves the structures of the frames.

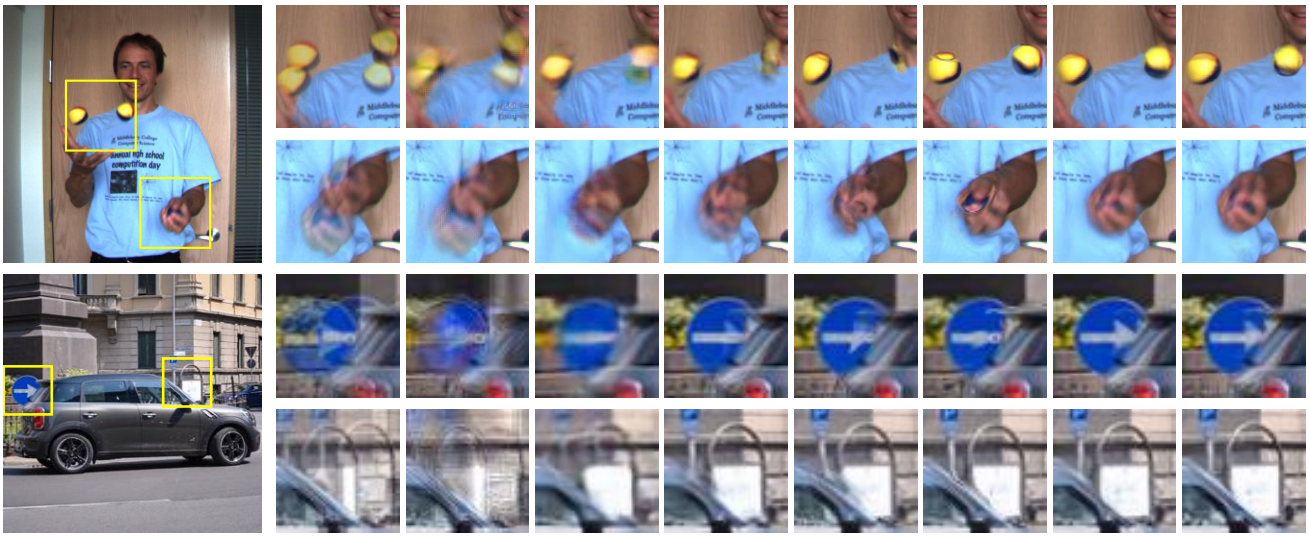


Figure 5: Visual comparison of sample sequences with large motions (1st - 2nd row) and visual comparison of sample sequences with occlusion (3rd - 4th row). There are occluded areas in the front and back of the car.

4.3. Quantitative Evaluation

We compare our method with simply overlapped results and several competing algorithms including Phase Based [32], MIND [28], SepConv [35], DVF [27], and SuperSlomo [20]. We evaluate two versions of our algorithm. One is the basic version of $F = 5, d = 1$ (Ours) and the other is the version of $F = 11, d = 2$ (Ours +). For a fair comparison, we fix the training environment. We implement the competing algorithms and train them with the train dataset introduced in Section 4.1 commonly for 50 epochs. We measure PSNR and SSIM of each algorithm for the three test datasets. The results are shown in Table 5. According to the table the kernel-based approach (SepConv) generally perform better than the flow-based ones (DVF, SuperSlomo). Finally, our method outperforms the other algorithms for all test datasets by a high margin. We also upload our result to Middlebury Benchmark [1] and compare it with the other recent state-of-the-art algorithms. As reported in Table 4, AdaCoF ranks 2nd in both IE (Interpolation Error) and NIE (Normalized Interpolation Error) among all published methods in Middlebury website. In addition, CyclicGen [26], which ranks 1st in IE, uses additional edge maps for sharper results and the cycle consistency loss is orthogonally applicable to our method. Also, DAIN [2], which ranks 1st in NIE, use pre-trained optical flow estimator and depth maps while our method does not require any additional information. Lastly, our approach shows better performance for data with dynamic motions such as Basketball, Dumptruck and Evergreen.

4.4. Visual Comparison

Because the video frame interpolation task does not have a fixed answer, the evaluations based on PSNR and SSIM are not perfect by themselves. Therefore we quantitatively evaluate the methods by comparing each result. Especially, we check how our method and other state-of-the-art algorithms handle the two main obstacles which make motions complex in real world videos: large motion and occlusion.

Large motion. When the reference point is located far away, the search area has to be expanded accordingly. Therefore the large motion problem is one of the most challenging obstacles in video frame interpolation research. The first and second rows of Figure 5 show the estimated results of various approaches including our method. The results of MIND, SepConv tend to be blurry and DVF, SuperSlomo suffer from some artifacts. Compared to the other competing algorithms, our approach better synthesizes fast moving objects. In addition, the perception-oriented AdaCoF (Ours-$\mathcal{L}_p$) mitigate the motion blurs of the objects.

Occlusion. Most of the objects in the intermediate frame appear in both adjacent frames. However, in case of occlusion, the object does not appear in one of the frames. Therefore, the appropriate frame has to be selected for each case, which makes the problem more difficult. In the third and fourth rows of Figure 5, a car causes occlusion in its front and back. Comparing the estimated images on occluded areas, our method handles the occlusion problems better than the other approaches.

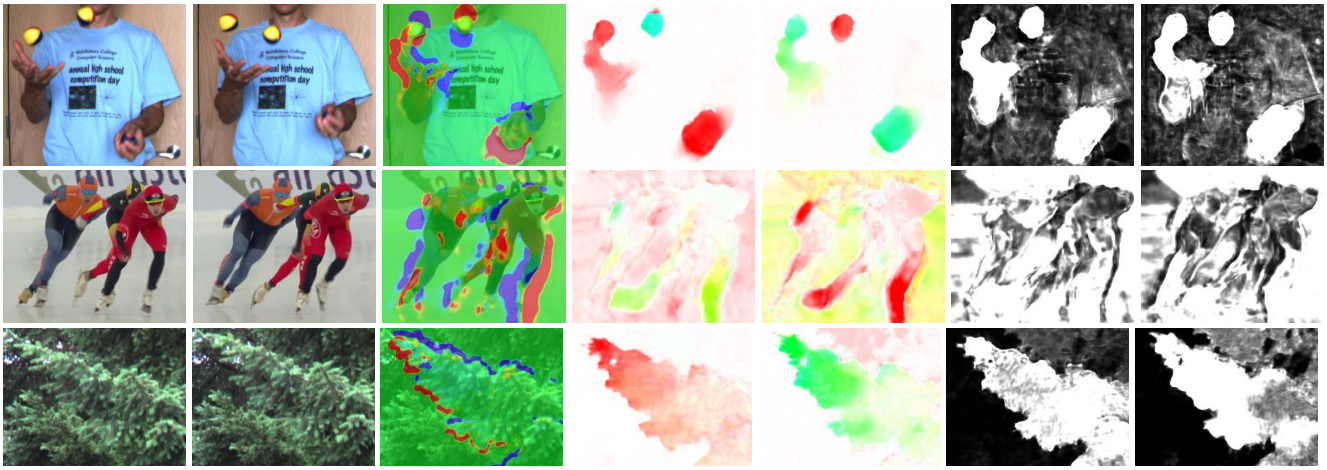


Figure 6: Various visualizations of the network outputs.

4.5. Offset Visualization

Our method estimates some parameters from the input images: the kernel weights $W_{k,l}$, the offset vectors $(\alpha_{k,l}, \beta_{k,l})$, and the occlusion map V. To check whether the parameters behave as intended, we visualize them in various ways. Further, because the network is trained by self-supervised learning, the visualizations can be obtained without any supervision. Therefore, they can be used for some other tasks in motion estimation research.

Occlusion map. The third column of Figure 6 shows the occlusion map V. To handle occlusion, the proper frame has to be selected in each case. For example, the pixels in the red area cannot be found in the second frame. Therefore the network decides to consider only the first frame, not the second one. The blue area can be explained in the same way for the second frame, and the green area means that there is no occlusion.

Mean Flow map. The fourth and fifth columns of Figure 6 show the weighted sum of the backward and forward offset vectors for each pixel. We call them Mean Flow F_m and they can be calculated by the following equation.

$$\Delta p_{k,l} = (\alpha_{k,l}, \beta_{k,l}) \tag{13}$$

$$F_m(i,j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l}(i,j) \Delta p_{k,l} \tag{14}$$

This means the overall tendency of the offset vectors. Therefore they might behave like a forward/backward optical flow and the figures prove it. This can be used as dense optical flow and can also be obtained from the other flow-based algorithms such as DVF and SuperSlomo.

Variance Flow map. The sixth and seventh columns of Figure 6 are the weighted variance of the backward and forward offset vectors. We call them Variance Flow map F_v and they can be calculated by the following equation.

$$F_v(i,j) = \sum_{k=0}^{F-1} \sum_{l=0}^{F-1} W_{k,l}(i,j) (F_m(i,j) - \Delta p_{k,l})^2 \tag{15}$$

The large value for this map means that the offset vectors for the pixel are more spread out so that it can refer to more pixels. According to the figure, more challenging locations such as large motions and occluded areas have larger variance values. Therefore, it can be used as a kind of uncertainty map for some motion estimation tasks. Unlike Mean Flow map, it can only be obtained through our method.

5. Conclusion

In this paper, we point out that the DoF of the warping operation to deal with various complex motions is one of the most critical factors in video frame interpolation. Then we propose a new operation called Adaptive Collaboration of Flows (AdaCoF). This method is the most generalized because all of the previous approaches are special versions of AdaCoF. The parameters needed for the AdaCoF operation are obtained from a fully convolutional network which is end-to-end trainable. Our experiments show that our method outperforms most of the competing algorithms even in several challenging cases such as those with large motion and occlusion. We visualize the network outputs to check whether they behave as intended and that the visualized maps are meaningful, so they can be used for other motion estimation tasks.

Acknowledgement This research was supported by R&D program for Advanced Integrated-intelligence for Identification (AIID) through the National Research Foundation of KOREA(NRF) funded by Ministry of Science and ICT (NRF-2018M3E3A1057289).

References

[1] Simon Baker, Daniel Scharstein, JP 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. 2, 5, 7

[2] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3703–3712, 2019. 3, 6, 7

[3] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2, 6

[4] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994. 2

[5] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, 2018. 4

[6] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Advances in neural information processing systems*, pages 730–738, 2016. 3

[7] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 5

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 3

[9] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423, 2017. 5

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016. 2

[11] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*, pages 658–666, 2016. 4

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 5

[14] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 4

[16] Ross Goroshin, Michael F Mathieu, and Yann LeCun. Learning to linearize under uncertainty. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1234–1242. Curran Associates, Inc., 2015. 4

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 6

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 3

[19] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, page 6, 2017. 2

[20] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2, 6, 7

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 2, 4

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 4

[25] Hongbin Liu, Ruiqin Xiong, Debin Zhao, Siwei Ma, and Wen Gao. Multiple hypotheses bayesian frame rate up-conversion by adaptive fusion of motion-compensated interpolations. *IEEE transactions on circuits and systems for video technology*, 22(8):1188–1198, 2012. 2

[26] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang. Deep video frame interpolation using cyclic frame generation. In *AAAI Conference on Artificial Intelligence*, 2019. 3, 6, 7

[27] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 4, 6, 7

[28] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016. 2, 4, 6, 7

[29] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving gradients: a path-based method for plausible image interpolation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 42. ACM, 2009. 2

[30] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations (ICLR)*, 2016. 4

[31] Simone Meyer, Abdelaziz Djelouah, Brian McWilliams, Alexander Sorkine-Hornung, Markus Gross, and Christopher Schroers. Phasenet for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[32] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2, 6, 7

[33] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3, 6

[34] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2

[35] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2, 5, 6, 7

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017. 5

[37] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016. 5

[38] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–733, 2018. 2, 5

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 4

[40] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2830–2839, 2017. 6

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[43] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015. 4

[44] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[45] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[46] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013. 2, 6

[47] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. Optical flow guided tv-l 1 video interpolation and restoration. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 273–286. Springer, 2011. 2

[48] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012. 6

[49] 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

[50] Zhefei Yu, Houqiang Li, Zhangyang Wang, Zeng Hu, and Chang Wen Chen. Multi-level video frame interpolation: Exploiting the interaction among different levels. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(7):1235–1248, 2013. 2

[51] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, pages 597–613. Springer, 2016. 4