# Video Frame Interpolation via Generalized Deformable Convolution

Zhihao Shi, Xiaohong Liu, *Graduate Student Member, IEEE*, Kangdi Shi, Linhui Dai, and Jun Chen, *Senior Member, IEEE*

***Abstract*—Video frame interpolation aims at synthesizing intermediate frames from nearby source frames while maintaining spatial and temporal consistencies. The existing deep-learning-based video frame interpolation methods can be roughly divided into two categories: flow-based methods and kernel-based methods. The performance of flow-based methods is often jeopardized by the inaccuracy of flow map estimation due to oversimplified motion models, while that of kernel-based methods tends to be constrained by the rigidity of kernel shape. To address these performance-limiting issues, a novel mechanism named generalized deformable convolution is proposed, which can effectively learn motion information in a data-driven manner and freely select sampling points in space-time. We further develop a new video frame interpolation method based on this mechanism. Our extensive experiments demonstrate that the new method performs favorably against the state-of-the-art, especially when dealing with complex motions. Code is available at https://github.com/zhshi0816/GDConvNet.**

***Index Terms*—Generalized deformable convolution, video frame interpolation.**

## I. Introduction

In RECENT years, owing to the hardware development and the availability of large-scale datasets, deep learning has achieved promising results in many computer vision and multimedia tasks [1] including, among others, super-resolution [2]–[4], optical flow estimation [5], [6], image dehazing [7], [8], action recognition [9], and video frame interpolation (VFI) [10]–[13]. VFI is a classic problem in the multimedia area and has received significant attention with the rapid growth of streaming videos. It aims at synthesizing intermediate frames from nearby sources while maintaining spatial and temporal consistencies. VFI has two main use cases; one is to perform error concealment at the decoder side [10], [11], and the other one is to increase the frame rate of a given video for better visual performance [12], [13]. In general, VFI methods can be roughly divided into two categories: flow-based methods and kernel-based methods.

Flow-based methods generate the value of each pixel in the target intermediate frame by finding an associated optical flow. Accurate estimation of the flow map is essential for producing desirable VFI results. However, in some cases with complex motions, it is hard to obtain an accurate flow map regardless whether traditional methods [14]–[16] or deep-learning-based methods [6], [17]–[19] are employed. Flow-based methods [1], [11], [20]–[22] typically adopt a linear model with the oversimplified assumption of uniform motion between neighboring frames. Recently, a more sophisticated approach was proposed in [23] for estimating motion trajectories, where the naive linear model is replaced by a more accurate quadratic model that can take advantage of latent motion information by simultaneously exploiting four consecutive frames. Nevertheless, it is conceivable that the complexities and irregularities of real-world motions cannot be completely captured by a simple mathematical model. Moreover, the pixel-level displacement performed in flow-based methods is inherently inadequate for handling diffusion and dispersion effects, especially when such effects are not negligible over the time interval between two consecutive frames.

Kernel-based methods directly generate the target intermediate frame by applying spatially-adaptive convolution kernels to the given frames. They circumvent the need for flow map estimation and consequently are not susceptible to the associated issues. On the other hand, the rigidity of the kernel shape [24], [25] severely limits the types of motions that such methods can handle. Indeed, one may need to choose a very large kernel size to ensure enough coverage, which is highly inefficient. As a partial remedy, reference [21] proposes adaptive deployment of convolution kernels guided by flow maps, but nevertheless, the receptive field is still constrained by the predetermined kernel shape. More recently, reference [12] introduces a new approach known as AdaCoF, which utilizes spatially-adaptive deformable convolution (DConv) to select suitable sampling points needed for synthesizing each target pixel. Although this approach eliminates the constraint on the kernel shape in the spatial domain, it does not fully exploit the degrees of freedom available in whole space-time.

In summary, flow-based methods and kernel-based methods have their respective limitations. For flow-based methods, even

Manuscript received August 20, 2020; revised November 28, 2020 and January 5, 2021; accepted January 8, 2021. Date of publication January 18, 2021; date of current version January 21, 2022. This work was supported by the Natural Sciences and Engineering Research Council of Canada through a Discovery Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ling-Yu Duan. *(Corresponding authors: Xiaohong Liu; Jun Chen.)*

The authors are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: shiz31@mcmaster.ca; liux173@mcmaster.ca; shik9@mcmaster.ca; dail5@mcmaster.ca; junchen@mail.ece.mcmaster.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMM.2021.3052419.

Digital Object Identifier 10.1109/TMM.2021.3052419

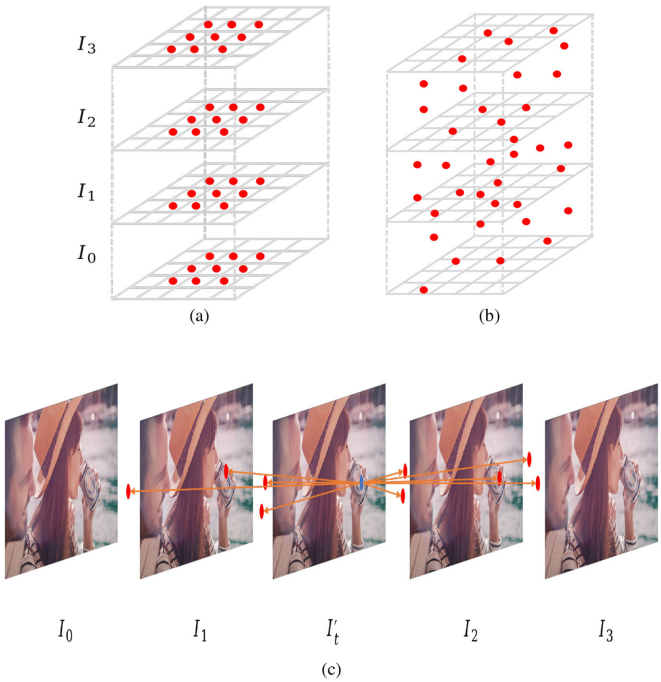



Fig. 1. Illustration of (a) conventional convolution with $3 \times 3 \times 4 = 36$ sampling points, (b) GDConv with the same number of sampling points, and (c) visualization of interpolating one frame with GDConv.

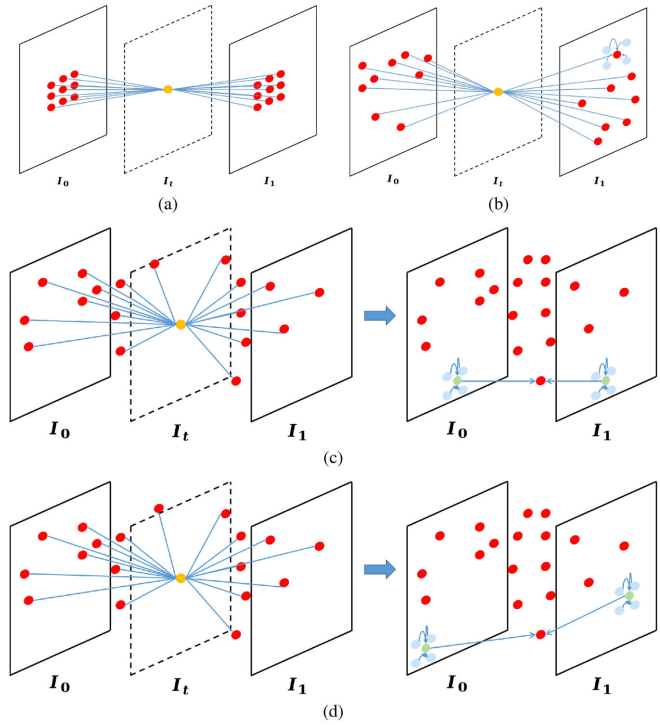



Fig. 2. Illustration of (a) conventional convolution, (b) AdaCoF, (c) basic GDConv, (d) advanced GDConv with $T = 1$. Here target pixels, sampling points, support points, and neighboring grid points are denoted by yellow, red, green, and blue dots, respectively. For AdaCoF, the value of each sampling point is specified via bilinear interpolation of its four neighboring grid points. For basic GDConv, the value of each sampling point is determined by its two support points via linear interpolation, or equivalently, by its eight associated grid points via trilinear interpolation. Advanced GDConv further removes the constraint that the support points need to be spatially aligned with the corresponding sampling point and allows more general numerical interpolation methods.

with the aid of sophisticated mathematical models, flow map estimation is still a challenging task due to the intricacies of inter-frame motion trajectories. For kernel-based methods, the predetermined kernel shape lacks the flexibility to cope with a great variety of motions in terms of range and pattern. While recent innovations have alleviated the rigidity issue to a certain extent, much remains to be done.

The main contribution of this paper is a new approach to VFI that overcomes the hurdles of the aforementioned methods and retains their desirable properties. The key mechanism underlying the proposed approach is generalized deformable convolution (GDConv). An illustration of the difference between conventional convolution and our GDConv in terms of the freedom to select sampling points can be found in Fig. 1(a) and (b). Fig. 1(c) provides a rough idea of how GDConv can be leveraged for VFI: each pixel (e.g., the blue one) in the target intermediate frame is synthesized based on the corresponding sampling points (the red ones). It is worth noting that as the sampling points are allowed to move freely in the continuous space-time, the receptive field of GDConv is basically unconstrained, making it possible to handle all kinds of motions (say, large motions). Moreover, GDConv does not directly adopt a predetermined mathematical model (e.g., linear or quadratic model) for motion estimation. Instead, it is trained to learn real-world motion trajectories and patterns via a data-driven approach. In our design, GDConv is encapsulated in a generalized deformable convolution module (GDCM). We integrate two GDCMs with several other modules, including the source extraction module (SEM), the context extraction module (CEM) and the post-processing module (PM), to form a generalized deformable convolution network (GDConvNet) for VFI. Our extensive experimental results demonstrate that owing to the effective design, the proposed GDConvNet performs favorably against the current state-of-the-art.

## II. Generalized Deformable Convolution Network

The overall architecture of GDConvNet is shown in Fig. 3. Given a video clip that consists of $T+1$ source frames[1] $I_0, I_1, \ldots, I_T$, the task of GDConvNet is to synthesize an intermediate frame $I_t$, $t \in [0, T]$. To this end, it first generates source features through SEM and extracts context maps $C_0, C_1, \ldots, C_T$ through CEM from $I_0, I_1, \ldots, I_T$. The input frames and context maps are then warped by two separate GDCMs according to the same source features. Finally, the warped frame $I'_t$ and the warped context map $C'_t$ are fed into the PM to produce the VFI result $\hat{I}_t$, which is an approximation[2] of $I_t$. The proposed network accomplishes the VFI task by employing a novel GDConv mechanism. Now we proceed to give a detailed

[1]For notional simplicity, we assume that the source frames are equally spaced in time. However, the proposed framework can in fact handle the unequal spacing case as well.

[2]The accuracy of this approximation can be evaluated by using objective image quality metrics (to be detailed later) or subjective criteria.

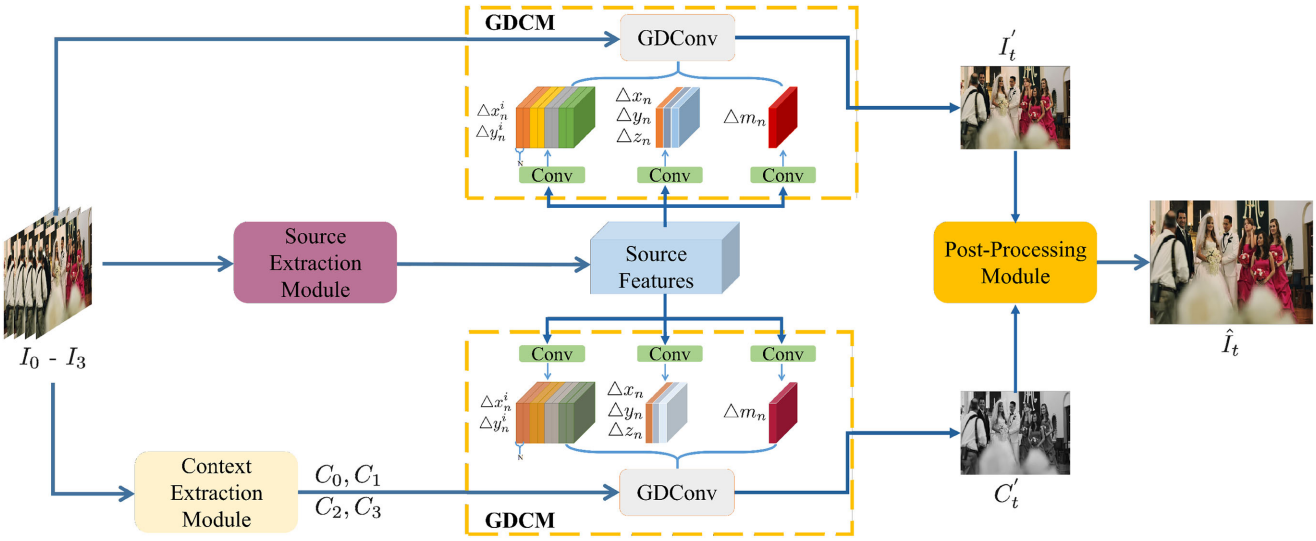



Fig. 3. Illustration of the architecture of GDConvNet with $T = 3$. Here $I_0, \ldots, I_3$ are input frames and $C_0, \ldots, C_3$ are their respective context maps; $\triangle x_n^i$ and $\triangle y_n^i$ are spatial offsets (horizontal and vertical) for support point; $\triangle x_n$, $\triangle y_n$ and $\triangle z_n$ are spatial offsets and temporal parameters for sampling points; $\triangle m_n$ is the modulation terms; $I'_t$ is a tentative prediction of the target frame $I_t$ while $C'_t$ denotes the predicted context map of $I_t$; $\hat{I}_t$ is the final output.

description of each module in Fig. 3, with a special emphasis on the GDCM where the GDConv mechanism is realized.

### A. Generalized Deformable Convolution Module

The input to the GDCM consists of the $T + 1$ source frames $I_0$, $I_1$, $\cdots$, $I_T$ (or the context maps $C_0$, $C_1$, $\cdots$, $C_T$) and the source features. As shown in Fig. 3, three different kinds of feature maps, which represent three different types of adaptive parameters, are generated through three different convolution layers, respectively. They are then fed to GDConv along with the source frames $I_0$, $I_1$, $\cdots$, $I_T$ (or the context maps $C_0$, $C_1$, $\cdots$, $C_T$) to synthesize $I'_t$ (or $C'_t$). Since the two GDCMs are almost identical, here we only describe the upper one in detail. Moreover, as the operations on the three color channels are the same, we simply regard $I_i$ as a single-channel image. For ease of exposition, we first give a brief review of VFI techniques based on conventional convolution [24] and AdaCoF [12], and then outline the improvements offered by the proposed GDConv.

Conventional convolution is employed in [24] for VFI. This can be formulated as:

$$I'_t(x, y) = \sum_{i=0}^{T} \sum_{m=1}^{M} W_m^i(x, y) \cdot I_i(x + x_m, y + y_m), \quad (1)$$

where $W_m^i(x, y)$ is a spatially-adaptive convolution weight, and $\{(x_m, y_m)\}_{m=1}^N$ is a collection of pre-defined convolution sampling offsets. Fig. 2(a) provides an illustration for the special case with $T = 1$, $M = 9$ and $\{(x_m, y_m)\}_{m=1}^M = \{(-1, -1), (-1, 0), \ldots, (1, 1)\}$. Ideally, the object (pixel) movement should be confined within the coverage of the convolution kernel. As such, in the presence of large motions, this approach is memory-inefficient due to the need for a large number of sampling points to ensure sufficient coverage.

The inefficiency of conventional convolution is largely a consequence of the pre-defined kernel shape (typically, a rectangular grid). AdaCoF [12] addresses this issue by adopting spatially-adaptive deformable convolution, resulting in the following formulation:

$$I'_t(x, y) = \sum_{i=0}^{T} \sum_{m=1}^{M} W_m^i(x, y) \cdot I_i(x + \triangle\alpha_m^i, y + \triangle\beta_m^i), \quad (2)$$

where $\{(\triangle\alpha_m^i, \triangle\beta_m^i)\}_{m=1}^M$ is a collection of adaptive sampling offsets. In the case where $\triangle\alpha_m^i$ and $\triangle\beta_m^i$ are not integers, $I_i(x + \triangle\alpha_m^i, y + \triangle\beta_m^i)$ is specified through bilinear interpolation. As a result of the introduction of adaptive sampling offsets, the kernel shape becomes adjustable, as shown in Fig. 2(b). For this reason, AdaCoF is able to cope with large motions using a relatively small number of sampling points. On the other hand, AdaCoF only exploits the degrees of freedom in the spatial domain. As a result, the sampling points are evenly split among the input frames. However, this is clearly suboptimal since the frames that are closer to the target intermediate frame in the temporal domain are more relevant and consequently should be allocated with more sampling points.

We shall develop a mechanism that enables flexible allocation of the sampling points across the input frames. In fact, we go one step further by allowing the sampling points to be freely distributed in whole space-time. The key idea is to associate each sampling point with an adaptive temporal parameter $z_n \in [0, T]$, leading to the following formulation:

$$I'_t(x, y) = \sum_{n=1}^{N} W_n(x, y) \cdot I(x + \triangle x_n, y + \triangle y_n, z_n). \quad (3)$$

Here, $I$ is a function (defined on a 3D space) obtained via a judicious extension of $I_0$, $I_1$, $\cdots$, $I_T$ to be detailed below (see Fig. 4 for an illustration of the special case in which

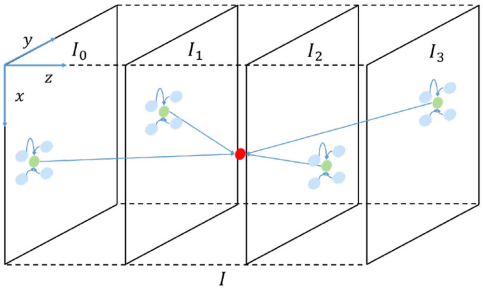



Fig. 4. Construction of function $I$ for the special case $T = 3$ with a sampling point $(x + \triangle x_n, y + \triangle y_n, z_n)$, its associated support points $(x + \triangle x_n^i, y + \triangle y_n^i, i), i \in \{0, 1, 2, 3\}$, and their neighboring grid points highlighted in red, green, and blue, respectively.

$T = 3$). Note that $z_n$ is allowed to be any real number in $[0, T]$ to facilitate end-to-end training. If $z_n$ is an integer, we set $I(x + \triangle x_n, y + \triangle y_n, z_n) = I_{z_n}(x + \triangle x_n, y + \triangle y_n)$. (Following [12], [26], [27], in the case where $\triangle x_n$ and $\triangle y_n$ are not integers, $I_{z_n}(x + \triangle x_n, y + \triangle y_n)$ is specified via bilinear interpolation of four neighboring grid points.) It can be seen that (3) reduces to (2) when $N = (T + 1)M$ and each value in $\{0, 1, \ldots, T\}$ is taken by the same number of $z_n$. Now it remains to deal with non-integer valued $z_n$, which occurs when the associated sampling point is not exactly located on an input frame. One simple solution is to set $I(x + \triangle x_n, y + \triangle y_n, z_n)$ as $(\lceil z_n \rceil - z_n) \cdot (I(x + \triangle x_n, y + \triangle y_n, \lfloor z_n \rfloor) + (z_n - \lfloor z_n \rfloor) \cdot I(x + \triangle x_n, y + \triangle y_n, \lceil z_n \rceil)$. (See Fig. 2(c) for an illustration of the special case in which $T = 1$.) More generally, we attach a set of support points $(x + \triangle x_n^i, y + \triangle y_n^i, i)$, $i \in \{0, 1, \ldots, T\}$, to each sampling point $(x + \triangle x_n, y + \triangle y_n, z_n)$, and use their values $I(x + \triangle x_n^i, y + \triangle y_n^i, i)$ (denoted as $s_n^i$ for short), $i \in \{0, 1, \ldots, T\}$, and their relative positions, to specify $I(x + \triangle x_n, y + \triangle y_n, z_n)$ (denoted as $s_n$ for short) via a numerical interpolation function $G$:

$$s_n = G(\triangle x_n, \triangle y_n, z_n, \{s_n^i, \triangle x_n^i, \triangle y_n^i\}_{i=0}^T). \tag{4}$$

Illustrations of special cases with $T = 1$ and $T = 3$ can be found in Fig. 2(d) and Fig. 4, respectively. Note that each support point has its own adaptive spatial offset $(\triangle x_n^i, \triangle y_n^i)$, which is not necessarily the same as $(\triangle x_n, \triangle y_n)$. Moreover, there is considerable freedom in the choice of $G$ as long as the differentiability condition needed for end-to-end training is satisfied. We will discuss several candidate numerical interpolation methods in Section IV-C. Finally, inspired by modulated deformable convolution [27], we rewrite (3) in the following equivalent form:

$$I'_t(x, y) = \sum_{n=1}^{N} W_n \cdot I(x + \triangle x_n, y + \triangle y_n, z_n) \cdot \triangle m_n(x, y), \tag{5}$$

where $\triangle m_n(x, y) \in [0, 1]$ is an adaptive modulation term.

As illustrated in Fig. 3, three types of feature maps are generated in GDCM via three different convolution layers. The first $2(T + 1)N$ feature maps represent the spatial offsets (horizontal and vertical) for the support points (i.e., $\triangle x_n^i$, $\triangle y_n^i$), and the next $3N$ feature maps represent the spatial offsets and temporal parameters for the sampling points (i.e., $\triangle x_n$, $\triangle y_n$, $z_n$), and the last $N$ feature maps represent the modulation terms (i.e., $\triangle m_n$). We set the initial values of the adaptive parameters $\triangle x_n$, $\triangle y_n$, $z_n$, $\triangle m_n$, $\triangle x_n^i$ and $\triangle y_n^i$ as 0, 0, 0, 1, 0, and 0, respectively.

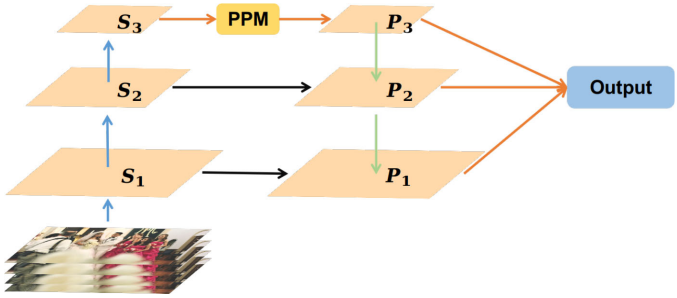



Fig. 5. Illustration of the architecture of SEM.

### B. Other Modules

Now we proceed to give a brief description of the remaining modules in the proposed GDConvNet.

*Source Extraction Module:* As shown in Fig. 5, we adopt the FPN backbone [28] to generate hierarchical features. In the bottom-up pathway, there are three levels (each consisting of two residual blocks and one convolution layer) and the associated feature maps (which are of different scales) are denoted as $S_1$, $S_2$, and $S_3$. The input $P_3$ to the top level of the top-down pathway is generated from $S_3$ through a pyramid pooling module [29]. $P_3$ is then upsampled and merged with $S_2$ via element-wise addition to generate $P_2$, which is further upsampled and merged with $S_1$ to generate $P_1$. Finally, $P_2$ and $P_3$ are upsampled and concatenated with $P_1$ to form the output.

*Context Extraction Module:* It is demonstrated in [30] that context information is very important for VFI. We use one convolution layer and two residual blocks [31] to sequentially extract contextual features. A SEblock [32] is then used to rearrange these feature maps, and finally its output is smoothed by a convolution layer.

*Post-Processing Module:* To refine the warped image, we adopt the GridDehazeNet architecture [33], where each row is associated with a different scale and contains five RDB blocks [34], while each column can be considered as a bridge connecting different scales through downsampling or upsampling modules. (which decrease or increase the size of feature maps by a factor of two.) Instead of employing the hard attention mechanism in [33], we use SEBlocks [32] to adaptively rebalance the incoming information flows at the junctions of GridDehazeNet.

## III. Understanding Generalized Deformable Convolution in VFI

In this section, we shall place generalized deformable convolution in a board context and explain why it is an effective mechanism for VFI.

### A. Related Works

Generalized deformable convolution is conceptually related to several existing ideas in the literature.

*Deformable Convolution:* There are many works on variants of conventional convolution with improved performance, including active convolution [35], dynamic filter [36], atrous convolution [37], among others. A culminating achievement of this line of research is deformable convolution [26], [27]. Our generalized deformable convolution degenerates to conventional deformable convolution [26], [27] if the temporal dimension is not present, and its basic form, shown in Fig. 2(c), can be viewed as a $3D$-version of deformable convolution.

*Non-Local Network:* In deep learning, non-locality means that the receptive field is not restricted to a certain local region and can capture long-range context information. The receptive field of conventional convolution is typically a fixed grid and consequently is local in nature. Significant efforts have been devoted to addressing this issue [29], [37]–[39]. Arguably the most successful one is [39], which takes all possible spatial positions into consideration. However, this comes at the cost of high memory usage. In contrast, generalized deformable convolution is memory-efficient as it is able to achieve non-local coverage and capture long-range context information with a relatively small kernel by adaptively and intelligently selecting sampling points in space-time.

*Attention Mechanism:* An attention mechanism enables differentiated treatment of different input features according to their relative importance, which has shown to yield significant performance gain in many vision tasks. Traditionally, it can be divided into spatial-wise attention [40] and channel-wise/temporal-wise attention [32]. Recently, there have also been attempts [41], [42] to combine these two types of attention. Nevertheless, in these approaches the spatial-wise and channel-wise/temporal-wise attention maps are still generated separately. It is interesting to note that generalized deformable convolution offers a natural way to consolidate these two types of attention by suitably modulating the sampling points at different locations in space-time.

*Non-Linearity:* The conventional approach to increasing the non-linearity of convolutional neural networks [43]–[45] is by stacking more non-linear modules [45], [46]. However, it has been recognized that a more effective approach is to allow the functionalities of constituent modules to be input-dependent [32], [40], [47]. From this perspective, generalized deformable convolution converts a linear convolution operation into a highly non-linear operation by adaptively adjusting its kernel according to the input, and by doing so it yields enhanced learning capabilities.

### B. Comparison With State-of-the-Art VFI Algorithms

The state-of-the-art VFI methods can be divided into two categories: flow-based methods and kernel-based methods. For illustrative purposes, we shall consider the simple scenario where two source frames $I_1$ and $I_2$ are used to predict one target frame $I_{1.5}$, unless specified otherwise.

*Flow-based:* These methods admit a common mathematical formulation as follows:

$$I'_{1.5\leftarrow 1}(x,y) = I_1(x+\triangle u^1, y+\triangle v^1), \tag{6}$$

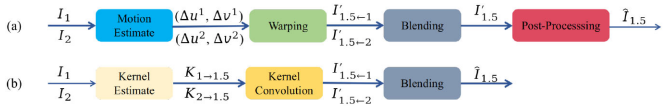



Fig. 6. Illustration of (a) flow-based VFI pipeline and (b) kernel-based VFI pipeline.

or

$$I'_{1.5\leftarrow 2}(x,y) = I_2(x+\triangle u^2, y+\triangle v^2), \tag{7}$$

where $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ are respectively optical flow fields from $I_{1.5}$ to $I_1$ and $I_2$, while $I'_{1.5\leftarrow 1}$ and $I'_{1.5\leftarrow 2}$ denote the warped images from each direction. The pipeline of flow-based methods is illustrated in Fig. 6(a). First, two input frames are used to estimate optical flow maps, typically with the help of traditional optical flow estimation methods [14]–[16] or convolution neural network [6], [17]–[19]. The input frames are then warped according to these optical flow maps. Finally, blending and post-processing operations are performed to generate the final output. The linear motion model is widely adopted in flow map estimation. However, this model is not accurate for describing accelerated and curvilinear motions. To handle such complex motions, a quadratic model is proposed in [23], where $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ are estimated based on four frames $I_0$, $I_1$, $I_2$, and $I_3$ instead of just $I_1$ and $I_2$. To understand the connection with our method, it is instructive to consider a special case of (5) with $N=1$, where $z_1=1$, $(\triangle x_1, \triangle y_1) = (\triangle x_1^1, \triangle y_1^1)$, or $z_1=2$, $(\triangle x_1, \triangle y_1) = (\triangle x_1^2, \triangle y_1^2)$:

$$\begin{aligned} I'_{1.5\leftarrow 1}(x,y) &= W_1 \cdot I(x+\triangle x_1, y+\triangle y_1, 1)\cdot \triangle m_1 \\ &= W_1 \cdot I_1(x+\triangle x_1^1, y+\triangle y_1^1)\cdot \triangle m_1, \end{aligned} \tag{8}$$

Or

$$\begin{aligned} I'_{1.5\leftarrow 2}(x,y) &= W_1 \cdot I(x+\triangle x_1, y+\triangle y_1, 2)\cdot \triangle m_1 \\ &= W_1 \cdot I_2(x+\triangle x_1^2, y+\triangle y_1^2)\cdot \triangle m_1. \end{aligned} \tag{9}$$

One can readily recover (6) and (7) from (8) and (9) by setting $W_1 = \triangle m_1 = 1$ and interpreting $(\triangle x_1^i, \triangle y_1^i)$ as $(\triangle u^i, \triangle v^i), i=1,2$. Similarly to the case with $(\triangle u^1, \triangle v^1)$ and $(\triangle u^2, \triangle v^2)$ in [23], the estimation of the offsets $(\triangle x_1^1, \triangle y_1^1)$ and $(\triangle x_1^2, \triangle y_1^2)$ can also benefit from more than two source frames. More importantly, in our method, the offset estimation does not directly resort to any predetermined mathematical model and is carried out in a completely data-driven manner. As such, it can cope with real-world motions more flexibly and accurately. Furthermore, for the general version of our method, the number of sampling points can be set to be greater than 1 (i.e., $N>1$), which, together with the freedom in choosing the space-time coordinates of the sampling points and the relaxation of the constraint $(\triangle x_n, \triangle y_n) = (\triangle x_n^i, \triangle y_n^i)$, makes it possible to capture complex diffusion and dispersion effects. Finally, we would like to point out that the space-time numerical interpolation operation in our method plays a role similar to that of the blending operation in some existing flow-based methods [13], [20], [22] (see also Fig. 6(a)), but requires fewer parameters, as it is performed at the sampling point level.

*Kernel-based:* These methods [24], [25], [43] generate two sets of spatially-adaptive convolution kernels and use them to convolve with source frame patches to get the predicted target frames $I'_{1.5\leftarrow 1}$, $I'_{1.5\leftarrow 2}$ from two sides, which are then blended at the pixel level to get final VFI result:

$$\begin{aligned}\hat{I}_{1.5}(x,y) &= I'_{1.5\leftarrow 1}(x,y) + I'_{1.5\leftarrow 2}(x,y) \\ &= K_1(x,y) * I_1(x,y) + K_2(x,y) * I_2(x,y). \quad (10)\end{aligned}$$

The pipeline of kernel-based methods is shown in Fig. 6(b). Note that in the presence of complex motions, the technique in [24] and [25] need to adopt large kernels (specifically, the size of convolutional kernels used in [24] and [25] are $41 \times 41$ and $51 \times 51$, respectively) to ensure sufficient coverage, which is inflexible and memory-inefficient. AdaCoF [12] addresses this issue by adopting deformable convolution. Nevertheless, the sampling points in AdaCoF are only spatially adaptive. In contrast, the proposed method can make more effective use of the sampling points by freely exploring space-time (not just in the spatial domain). As such, it often suffices to employ small kernels, even when dealing with very complex motions. Our method also has the additional advantage of blending images at the sampling point level (in the form of space-time numerical interpolation), which is more efficient than blending at the pixel level in kernel-based methods.

## IV. Four-Frame VFI Experiments

Due to its flexibility, our method can leverage an arbitrary number of frames for VFI. Here we focus on the four-frame VFI case. The experimental results for two-frame VFI will be presented in Section V.

### A. Implementation Details

We use four source frames $I_0$, $I_1$, $I_2$, and $I_3$ to synthesize the target frame $I_{1.5}$. In GDConv, the number of sampling points for each warped pixel is set to 25. The loss function, the training dataset, and the training strategy are described below.

*Loss Function:* In addition to the supervision provided at the output end, we introduce intermediate supervision to ensure proper training of the GDCM (which is the key component of GDConvNet). Note that without intermediate supervision, we have no direct control of the training of the GDCM due to the fact that the downstream post-processing module, which is a relatively large and complex network, tends to dilute the impact of the supervisory signal. The overall loss function can be formulated as:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_r + \lambda \mathcal{L}_w \\ &= \sum_x ||\hat{I}_t(x) - I_{GT}(x)||_1 + \lambda \sum_x ||I'_t(x) - I_{GT}(x)||_1, \quad (11)\end{aligned}$$

where $I_{GT}$ is the ground-truth frame, and $\lambda$ is a hyper-parameter to balance the warped loss $\mathcal{L}_w$ and the refined loss $\mathcal{L}_r$. (Experimentally, we found that $\lambda = 0.5$ yields the best performance.) We use the $\ell_1$ norm instead of the $\ell_2$ norm because the latter is known to produce blurry results in image synthesis tasks. Following [12], [21], [22], [48], we use the Charbonnier Function $\Phi(x) = \sqrt{x^2 + \epsilon^2}$ to smoothly approximate the $\ell_1$ norm and set $\epsilon = 10^{-6}$.

*Training DataSet:* The Vimeo90 k Septuplet training dataset [13] is used to train our model. This training dataset is composed of 64 612 seven-frame sequences with a resolution of $256 \times 448$. We use the first, the third, the fifth, and the seventh frames (corresponding to $I_0$, $I_1$, $I_2$, and $I_3$ in our notation, respectively) of each sequence to predict the fourth one (corresponding to $I_{1.5}$). We randomly crop image patches of size $256 \times 256$ for training. Horizontal and vertical flipping, as well as temporal order reversal, are performed for data augmentation.

*Training Strategy:* Different from [13], [21], [22], our network can be trained from scratch without relying on any pre-trained model. We adopt the Adam optimizer [49], where $\beta_1$ and $\beta_2$ are set as the default values 0.9 and 0.999, respectively. We set the training batch size as 8 and train our network for 14 epochs (nearly 11 300 iterations) in total. The initial learning rate is set as $10^{-3}$, and the learning rate is reduced by a factor of two every 4 epochs for the first 8 epochs and by a factor of five every 2 epochs for the last 6 epochs. The training is carried out on four NVIDIA GTX 1080Ti GPUs, and takes about 58 hours to converge.

### B. Evaluation Datasets

The following three datasets are used for performance evaluation.

*Vimeo90 K Septuplet Test Set [13]:* This dataset consists of 7824 video sequences, each with 7 frames. As in the case of the Vimeo90 K Septuplet training dataset, the first, the third, the fifth, and the seventh frames of each sequence are leveraged to synthesize the fourth one. The image resolution of this dataset is $256 \times 448$.

*Gopro Dataset [50]:* This dataset is composed of 33 high-resolution videos recorded by hand-held cameras. The frame rate of each video is 240 fps, and the image resolution is $720 \times 1280$. The dataset was released in an image format, consisting of a total of 35 782 images. We successively group every 25 consecutive images as a test sequence, and resize the images to $360 \times 480$. Finally, 1392 test sequences are selected. For each sequence, the first, the ninth, the seventeenth, and the twenty-fifth frames (corresponding to $I_0$, $I_1$, $I_2$, and $I_3$, respectively) are used to synthesize the thirteenth frame (corresponding to $I_{1.5}$). This dataset is rich with non-linear camera motions and dynamic object motions, posing significant challenges to VFI methods in these respects.

*Adobe240 Dataset [51]:* This dataset consists of 133 240 fps videos in total, where the resolution of each video is $720 \times 1280$. These videos are recorded by hand-held cameras, and mainly contain outdoor scenes. Different from the Gopro dataset, this dataset is released in a video format. We extract 7479 non-overlapped test sequences, each with 25 frames. This dataset is rich with large motions. Indeed, it has the largest average pixel displacement among the three datasets under consideration according to Table I. Therefore, it can be used to examine the strength of a VFI method in handling such motions.

TABLE I
THE STATISTICS OF PIXEL DISPLACEMENT WITHIN DIFFERENT DATASETS. THIS TABLE SHOWS THE AVERAGE PIXEL DISPLACEMENT, THE PERCENTAGE OF PIXELS WITH DISPLACEMENT LARGER THAN 15, 20, AND 25 RESPECTIVELY FOR THREE DATASETS

| Dataset | avg disp. | $> 15$ | $> 20$ | $> 25$ |
|---|---|---|---|---|
| Vimeo90k Dataset | 6.1 | 9.1% | 5.0% | 3.0% |
| Gopro Dataset | 6.1 | 7.0% | 2.7% | 1.0% |
| Adobe240 Dataset | 8.2 | 13.0% | 9.0% | 6.1% |

### C. Numerical Interpolation Methods

As described in Section II-A, a numerical interpolation function $G$ is used to specify the value $s_n = I(x + \triangle x_n, y + \triangle y_n, z_n)$ of a sampling point in accordance with its position and the corresponding support points $s_n^i = I(x + \triangle x_n^i, y + \triangle y_n^i, i)$, $i \in \{0, 1, \ldots, T\}$, when it does not exactly lie on an input frame (i.e., when $z_n$ is not an integer). In principle, any numerical interpolation function satisfying the differentiability condition can be leveraged for this purpose. However, different numerical interpolation functions may generate different values for the same sampling point and consequently lead to different final outputs. Therefore, it is important to understand how the choice of the numerical interpolation function affects the overall system performance. To this end, we investigate the following representatives: linear interpolation, 3D and 1D versions of inverse-distance-weighted interpolation, and polynomial interpolation.

*1) Linear Interpolation:* This is one of the simplest interpolation methods. It can be be formulated as:

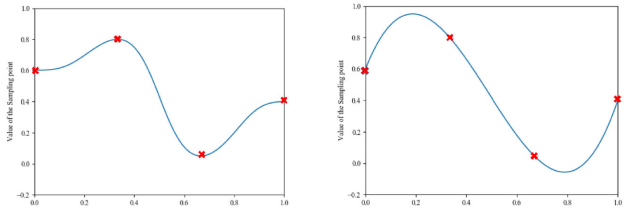
$$s_n = \sum_{i=0}^{T} \max(0, 1 - |z_n - i|) \cdot s_n^i. \tag{12}$$

Note that even if $T > 1$, only two adjacent support points are taken into consideration in (12) for interpolating $s_n$. (The maximum operation suppresses the contribution of other support points.) We regard this interpolation method as the baseline in comparisons.

*2) 3D Version of Inverse-Distance-Weighted Interpolation (3D Inv):* In contrast to linear interpolation, this method makes use of all support points (see Fig. 4) as follows:

$$s_n = \frac{\sum_{i=0}^{T} w_i \cdot s_n^i}{\sum_{i=0}^{T} w_i}, \tag{13}$$

where $w_i = 1/((d_x^i)^2 + (d_y^i)^2 + (d_z^i)^2)$, $d_x^i = |\triangle x_n - \triangle x_n^i|/H$, $d_y^i = |\triangle y_n - \triangle y_n^i|/W$, and $d_z^i = |z_n - i|/T$. The quantitative comparisons in Table II indicate that leveraging all support points instead of just two adjacent points yields better performance. Table III shows the means of $(d_x^i)^2$, $(d_y^i)^2$, and $(d_z^i)^2$ (averaged over $i$), denoted as $(d_x)^2$, $(d_y)^2$ and $(d_z)^2$, respectively. It is clear that $(d_x)^2$ and $(d_y)^2$ are about two orders of magnitude smaller than $(d_z)^2$. This implies that it might suffice to set the weights based on the temporal information alone, which naturally suggests the following interpolation method.




(a) (b)

Fig. 7. Illustration of (a) 1D version of inverse distance weighted interpolation and (b) polynomial interpolation with support points highlighted in red. Here $s_n^0 = 0.6$, $s_n^1 = 0.8$, $s_n^2 = 0.05$, and $s_n^3 = 0.4$, respectively.

*3) 1D Version of Inverse-Distance-Weighted Interpolation (1D Inv):* Setting $w_i = 1/(d_z^i)^2$ in (13) leads to the 1D version of inverse distance weighted interpolation (see Fig. 7(a) for an example with $T = 3$). The quantitative results of this interpolation method are shown in Table II. Somewhat surprisingly, the 1D version slightly outperforms its 3D counterpart. The reason is that focusing on the dominant dimension enables more effective use of the training data and consequently yields more accurate VFI results. This suggests that it might be possible to further improve the performance by employing more advanced 1D interpolation methods.

*4) Polynomial Interpolation (Poly):* This method uses a polynomial function of degree $T$ to perform interpolation. More specifically, we have:

$$G = a_0 + a_1 z_n + \cdots + a_T z_n^T, z \in [0, T], \tag{14}$$

where the coefficients $a_0, a_1, \cdots$, and $a_T$ can be uniquely determined by jointly solving $T + 1$ linear equations $G|_{z_n = i} = s_n^i$, $i \in \{0, 1, \ldots, T\}$. It should be emphasized that sampling points and their associated support points are still selected in 3D space-time even if a 1D interpolation method is adopted; as such, the overall method is intrinsically 3D.

Fig. 7(b) provides an example of polynomial interpolation with $T = 3$. In contrast to 1D Inv, polynomial interpolation is able to generate values beyond the upper and lower limits of $s_n^i$, $i \in \{0, 1, \ldots, T\}$. This extra freedom might be the reason why polynomial interpolation leads to 0.5 dB improvement over 1D Inv as shown in Table II.

To provide supporting evidence for our conjecture, we count the number of sampling points whose values are beyond the upper or lower limit. As shown in Table IV, for the upper GDCM used to synthesize intermediate frame $I_t'$, there are 10.8% and 22.6% sampling points beyond the upper limit and lower limit respectively. As for the lower GDCM used to predict the context map $C_t'$, 9.3% and 10.0% points are beyond the upper limit and the lower limit, respectively. We then clamp those values to their associated limits and reevaluate the model on the test datasets. As shown in Table II, indeed, forcing the values of sampling points to stay in the range set by support points jeopardizes the performance.

Fig. 8 provides visual examples of the results. It can be seen that compared to 1D inv, standard polynomial interpolation provides a better reconstruction in the texture regions, which usually contain a fair amount of sampling points beyond limits. In

TABLE II
QUANTITATIVE COMPARISONS OF GDCONVNET WITH DIFFERENT NUMERICAL INTERPOLATION METHODS ON VIEMO-90 K TEST DATASET, GOPRO DATASET, AND ADOBE240 DATASET

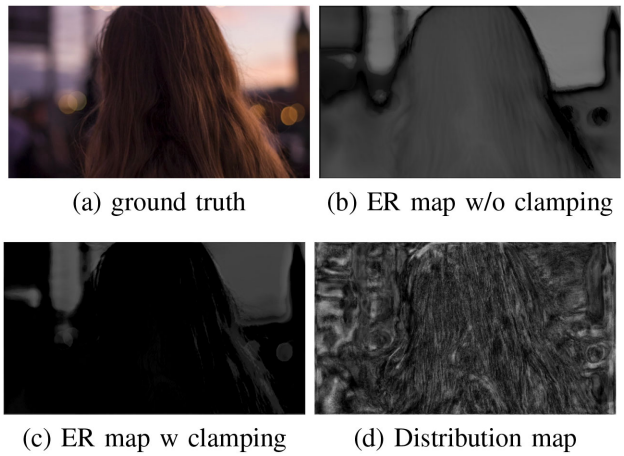
| Method | #Parameters | Vimeo-90k | | Gopro | | Adobe240 | |
|---|---|---|---|---|---|---|---|
| | (million) | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| Ours-Linear | 5.1 | 34.96 | 0.9534 | 30.06 | 0.9092 | 34.20 | 0.9422 |
| Ours-3D Inv | 5.1 | 35.01 | 0.9535 | 30.12 | 0.9099 | 34.27 | 0.9427 |
| Ours-1D Inv | 5.1 | 35.08 | 0.9541 | 30.16 | 0.9099 | 34.36 | 0.9436 |
| Ours-Poly | 5.1 | **35.58** | **0.9580** | **30.49** | **0.9180** | **34.53** | **0.9456** |
| Ours-Poly-clamping | 5.1 | 35.10 | 0.9548 | 30.18 | 0.9072 | 34.33 | 0.9442 |

TABLE III
MEAN OF THE SQUARED DISTANCE

| $(d_x)^2$ | $(d_y)^2$ | $(d_z)^2$ |
|---|---|---|
| 0.0025 | 0.0010 | 0.2009 |

TABLE IV
THE STATISTICAL DISTRIBUTION OF SAMPLING POINTS BEYOND LIMITS

| Method | Beyond Upper | Beyond Lower |
|---|---|---|
| Upper GDCM | 10.8% | 22.6% |
| Lower GDCM | 9.3% | 10.0% |



(a) ground truth (b) ER map w/o clamping

(c) ER map w clamping (d) Distribution map

Fig. 8. Visualization of (a) ground truth, (b) error residual (ER) map generated by GDConvNet with standard polynomial, (c) error residual map generated by GDConvNet with clamped polynomial, and (d) distribution map of sampling points beyond limits. The error residual map is calculated by $ER = MSE_{1\_inv} - MSE_{poly}$, where $MSE_{1\_inv}$ denotes the mean squared error map between the ground truth and the result generated by GDConvNet with 1D Inv, and $MSE_{poly}$ is similarly defined for polynomial interpolation.

contrast, clamped polynomial interpolation performs considerably worse than the standard one in these regions. Similar phenomena can be observed for images in different datasets. In summary, polynomial interpolation is able to generate sampling points beyond upper and lower limits, and these sampling points contribute positively to the synthesis of the texture regions of the images, which helps to improve the overall performance.

### D. Comparison With the State-of-the-Art

We compare our best-performing GDConvNet (Ours-Poly) with the state-of-the-art VFI algorithms on the aforementioned three evaluation datasets. Specifically, the following ones are chosen for comparison: the phase-based method (Phase) [52], separable adaptive convolution (SepConv) [25], deep voxel flow (DVF) [19], SuperSlomo (Slomo) [20], quadratic video interpolation (QVI) [23], and adaptive collaboration of flows (AdaCoF) [12]. Since these methods just use two frames ($I_1$, $I_2$) to synthesize the target frame,[3] we also provide a degraded version of our method (Ours-Poly*) with 4 frames ($I_0$, $I_1$, $I_2$, $I_3$) for offset generation and 2 frames ($I_1$, $I_2$) for target frame prediction. For fair comparison, DVF, Slomo, QVI, and AdaCoF are retrained on our training dataset. As the SepConv training code is not available, we choose to directly evaluate the original SepConv model.

In Table V, we quantitatively compare our method with the state-of-the-art methods on the evaluation datasets under two well-known objective image quality metrics, PSNR and SSIM. It can be seen that although it suffers from some performance degradation with respect to Ours-Poly, Ours-Poly* still performs on par with QVI (which is 6 times as large as Ours-Poly* in terms of model size) and surpasses other methods by a visible margin. As for Ours-Poly, it shows a significant improvement over its degraded counterpart due to the complete freedom in exploiting the given frames, and ranks consistently at the top in Table V (except for the Gopro dataset on which it comes in a close second in terms of the SSIM value). Overall, our method has a clear advantage under joint consideration of cost and performance.

Fig. 9 shows some qualitative comparisons. It can be seen that our method produces clearer and sharper results. For example, on the first row, our method is capable of generating smooth edges around the hand compared with that of Phase, DVF, SepConv, Slomo, QVI, and AdaCoF.

### E. Ablation Study

In our ablation studies, we adopt polynomial interpolation and consider a simplified version of GDConvNet in which the CEM and the associated GDCM, as well as the PM, are removed. This simplification greatly reduces the training time and, more importantly, enables us to focus on the most essential aspects of GDConvNet.

*1) Generalized Deformable Convolution Module:* In order to validate the effectiveness of our design, we compare the proposed GDConv with DConv (more precisely, spatially-adaptive

[3] Although 4 frames are employed in QVI, only 2 of them are directly involved in predicting the target frame.

TABLE V
QUANTITATIVE COMPARISONS OF DIFFERENT VFI METHODS ON VIMEO90 K SEPTULET TEST SET, GOPRO DATASET AND ADOBE240 DATASET, WHERE THE FIRST PLACE AND SECOND PLACE ARE HIGHLIGHTED IN RED AND BLUE, RESPECTIVELY

| Method | #Parameters | Vimeo90K | | Gopro | | Adobe240 | |
|---|---|---|---|---|---|---|---|
| | (million) | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| Ours-Poly | 5.1 | **35.58** | **0.9580** | **30.49** | 0.9180 | **34.53** | **0.9456** |
| Ours-Poly* | 5.1 | 35.01 | 0.9558 | 30.12 | 0.9100 | 34.12 | 0.9422 |
| AdaCoF | 21.8 | 33.92 | 0.9453 | 28.45 | 0.8734 | 33.17 | 0.9305 |
| QVI | 29.2 | 35.19 | 0.9563 | 30.24 | **0.9230** | 33.06 | 0.9393 |
| Slomo | 39.6 | 33.73 | 0.9453 | 28.50 | 0.8827 | 31.94 | 0.9264 |
| SepConv | 21.6 | 33.65 | 0.9435 | 28.66 | 0.8798 | 33.41 | 0.9349 |
| DVF | 3.8 | 30.79 | 0.8912 | 25.13 | 0.7633 | 22.33 | 0.6159 |
| Phase | – | 30.52 | 0.8854 | 26.17 | 0.8135 | 31.20 | 0.8930 |

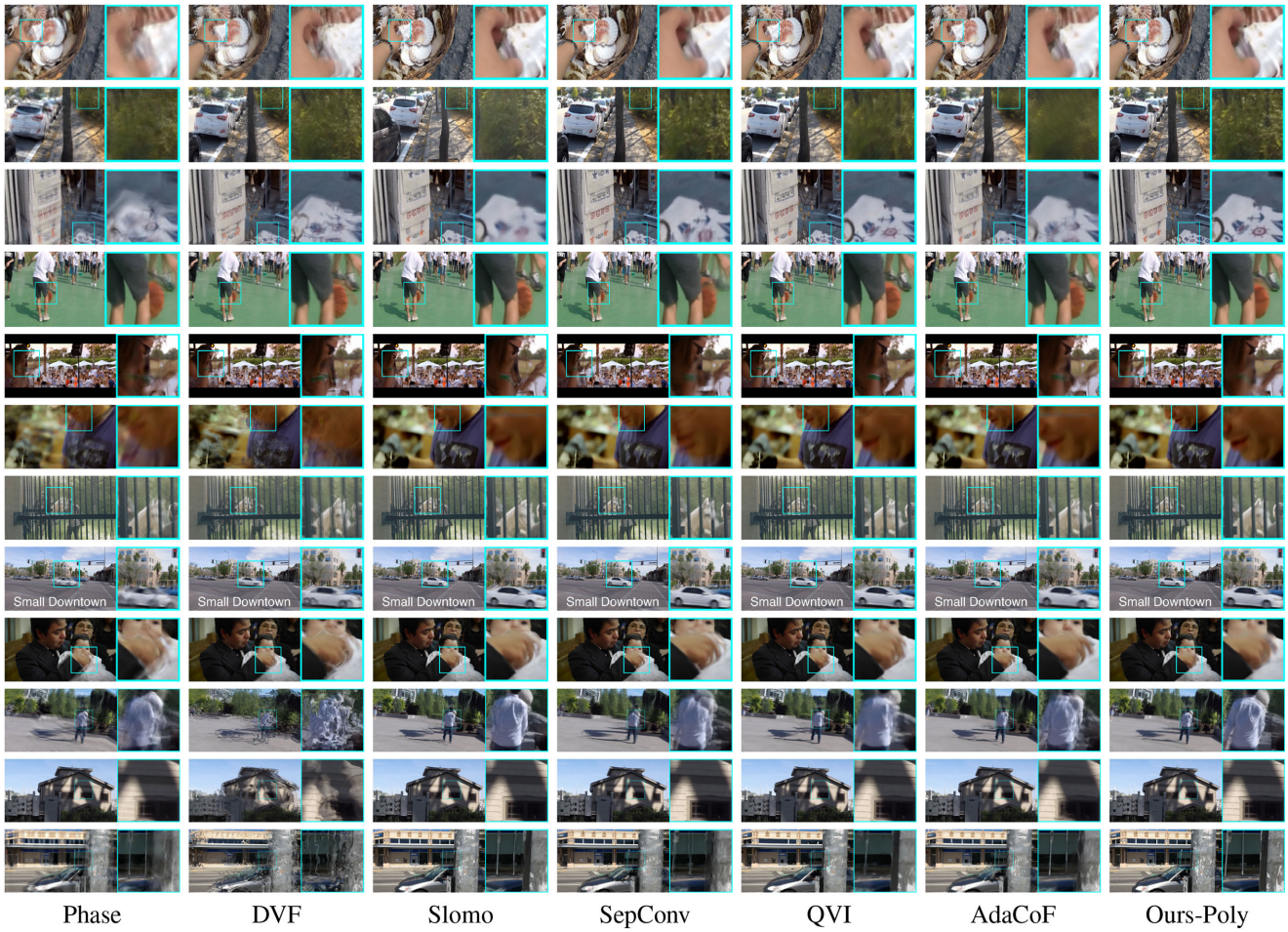



Fig. 9. Qualitative comparisons of different VFI algorithms.

DConv or modulated DConv) adopted by [12], as well as several variants of GDConv.

*Superiority of GDConv over DConv:* As mentioned earlier, the proposed GDConv is able to fully exploit the given source frames in accordance with their relevance to the target intermediate frame in terms of temporal distance. In contrast, the performance of DConv is limited by the inflexibility in choosing the number of sampling points from each source frame. For instance, consider the case where 4 consecutive frames are used for VFI and the convolution kernel size is set to 3. DConv is constrained to select 9 sampling points from each frame. This is inefficient from the perspective of resource allocation since the source frames closer to the target intermediate frame in time are conceivably more informative and should receive more attention. In this sense, the proposed GDConv is more desirable as it is endowed with complete freedom to select sampling points in space-time. Specifically, in GDConv, the number of sampling points in each frame is adjustable according to the significance of that frame in synthesis. More importantly, sampling points are not even required to lie exactly on the source frames, and are

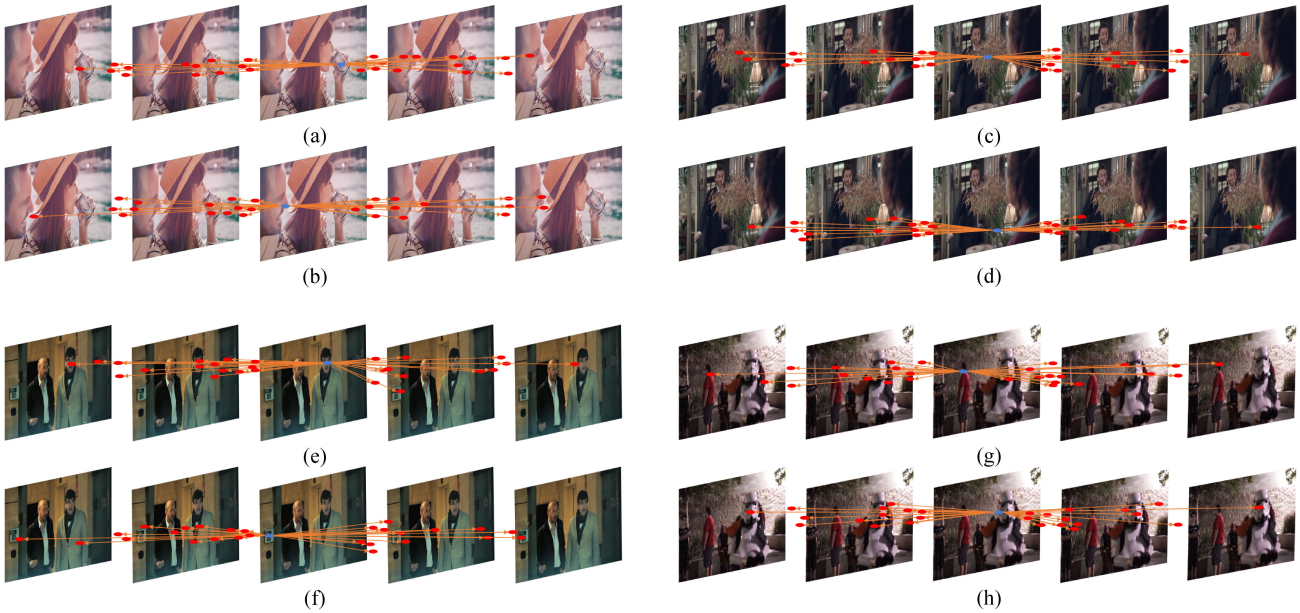



Fig. 10. Visualization of sampling points in GDCM when $t = 1.5$. Here ((a), (b)), ((c), (d)), ((e), (f)), and ((g), (h)) illustrate two different pixels in a same target intermediate frame and their associated sampling points respectively. It can be seen that sampling points are not exclusively located between $I_1$ and $I_2$. Indeed, there are some between $I_1$ and $I_2$, and some between $I_2$ and $I_3$. This indicates that the information from $I_1$ and $I_2$ is more significant for synthesizing $I_t$, but $I_0$ and $I_3$ also contribute to the synthesized result. .

TABLE VI
COMPARISONS OF DCONV, GDCONV WITH DIFFERENT NUMBERS OF SAMPLING POINTS, AND SOME VARIANTS OF GDCONV

| Method | Sampling points | PSNR | SSIM |
|---|---|---|---|
| DConv | 25 | 32.82 | 0.9236 |
| GDConv | 1 | 33.40 | 0.9342 |
| GDConv | 9 | 33.98 | 0.9414 |
| GDConv | 25 | **34.20** | **0.9436** |
| GDConv | 36 | 34.17 | 0.9430 |
| Variant (a) | 25 | 32.99 | 0.9274 |
| Variant (b) | 25 | 33.24 | 0.9310 |
| Variant (c) | 25 | 33.92 | 0.9410 |
| Variant (d) | 25 | 34.06 | 0.9418 |
| Variant (e) | 25 | **34.20** | **0.9436** |

allowed to be anywhere in the spatio-temporal domain specified by their associated parameters $\triangle x_n$, $\triangle y_n$ and $z_n$ (see Fig. 10 for some visual results). This mechanism is especially important for VFI since it is better suited to cope with complex and irregular inter-frame motions. In Table VI, we provide quantitative comparisons of DConv and GDConv. Here the number of input source frames is 4. In GDConv, the number of sampling points is set to 36. For fair comparison, the kernel size in DConv is chosen to be 3; thus, there are $3 \times 3 \times 4 = 36$ sampling points in total, as well. It is evident that the proposed GDConv achieves better performance in terms of the PSNR and SSIM metrics.

*Importance of Spatio-Temporal Freedom:* We consider the following 4 variants of GDConv to illustrate the importance of spatio-temporal freedom for sampling points.

a) No spatio-temporal freedom: $(\triangle x_n^i, \triangle y_n^i), i \in \{0, 1, 2, 3\}$, are identical and fixed to be a distinct point in a $5 \times 5$ grid $\{(-2, -2), (-2, -1), \ldots, (2, 2)\}$, and $z_n = 1.5$.

b) Limited spatial freedom, no temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, are identical but adaptive, and $z_n = 1.5$.

c) Limited spatial freedom, complete temporal freedom: $(\triangle x_n^i, \triangle y_n^i), i \in \{0, 1, 2, 3\}$, are identical but adaptive, and $z_n$ is adaptive.

d) Complete spatial freedom, no temporal freedom: $(\triangle x_n^i, \triangle y_n^i), i \in \{0, 1, 2, 3\}$, can be different from each other and are individually adaptive, and $z_n = 1.5$.

e) Complete spatio-temporal freedom: $(\triangle x_n^i, \triangle y_n^i)$, $i \in \{0, 1, 2, 3\}$, can be different from each other and are individually adaptive, and $z_n$ is also adaptive.

The results of the experiment are shown in Table VI. One can easily find that the performance rises progressively with the availability of every additional degree of freedom. It is worth noting that the temporal parameter $z_n$ is better interpreted as being effective time instead of physical time. Indeed, forcing $z_n = 1.5$ limits the degrees of freedom and jeopardizes the performance.

*Choice of the Number of Sampling Points:* We further investigate how to choose the number of sampling points in GDConv. As shown in Table VI, as the number of sampling points increases, the performance improves initially, but becomes saturated eventually. In particular, using more than 36 sampling points does not further enhance the quality of synthesized frames.

*2) Input Length and Offset Generation:* So far, except for the degraded version in Section IV-D, we have assumed that

TABLE VII
COMPARISONS FOR DIFFERENT NUMBERS OF REFERENCE FRAMES (WITH THE NUMBER OF GENERATION FRAMES SET TO BE THE SAME AS THAT OF REFERENCE FRAME)

| Reference Frames | PSNR | SSIM |
|---|---|---|
| $I_1$, $I_2$ | 33.69 | 0.9416 |
| $I_0$, $I_1$, $I_2$ | 33.97 | 0.9427 |
| $I_0$, $I_1$, $I_2$, $I_3$ | **34.20** | **0.9436** |

TABLE VIII
COMPARISONS FOR DIFFERENT NUMBERS OF GENERATION FRAMES (WITH THE REFERENCE FRAMES FIXED TO BE $I_1$ AND $I_2$)

| Generation Frames | PSNR | SSIM |
|---|---|---|
| $I_1$, $I_2$ | 33.69 | 0.9416 |
| $I_0$, $I_1$, $I_2$ | 33.84 | 0.9418 |
| $I_0$, $I_1$, $I_2$, $I_3$ | **34.05** | **0.9434** |

all 4 source frames $I_0$, $I_1$, $I_2$, and $I_3$ participate in generating offsets (as well as $z_n$ and $\triangle m_n$) and in predicting the target intermediate frame $I_{1.5}$. It is interesting to study how the proposed method performs if one only utilizes a subset of source frames. In fact, our framework is flexible enough to allow the use of different subsets of source frames for offset generation and frame prediction separately. For clarity, we shall refer to source frames used for generating offsets as generation frames and those directly involved in predicting the target intermediate frame as reference frames. For example, if we use $I_0$, $I_1$, $I_2$ to generate offsets for $I_1$ and $I_2$, which are subsequently leveraged to predict $I_{1.5}$, then $I_0$, $I_1$, $I_2$ are generation frames while the latter two are reference frames. We first study the scenario with the same subset of source frames used for both purposes. It is clear from Table VII that the VFI result improves progressively with the increase in the number of reference frames (as well as generation frames). We further investigate the scenario where reference frames and generation references are not necessarily the same. Specifically, we fix $I_1$ and $I_2$ to be reference frames, and consider various combinations of generation frames. It can be seen from Table VIII that increasing the number of generation frames leads to better performance. This is consistent with a similar finding regarding flow-based methods: namely, it is profitable to have three or more generation frames as that opens the door for exploiting higher-order approximation of motion trajectories (instead of relying on linear approximation, which is basically the only available choice in the case with just two generation frames). Finally, comparing the corresponding rows in Table VII and Table VIII reveals that VFI can also benefit from an increase in the number of reference frames (when the number of generation frames is fixed).

### F. Failure Case Analysis

Our method is trained in a purely data-driven manner to learn motion estimation. As such, it is able to handle complex motion patterns that cannot be characterized by simple mathematical models. On the other hand, the success of our method depends critically on the quality of the training dataset, which should ideally contain extensive motion patterns to ensure sufficient coverage. The performance of our method tends to degenerate when the motion patterns encountered in the evaluation dataset deviate significantly from those in the training dataset. Fig. 11 provides some examples where object motions are atypical with respect to the training dataset. It can be seen that the VFI results produced by our method are somewhat blurry (albeit still slightly better than those of QVI, which is the best known mathematical-model-based method).

(a) Ours (b) QVI

Fig. 11. Visualization of failure cases.

## V. TWO-FRAMES VFI EXPERIMENTS

As described earlier, our method is able to handle an arbitrary number of frames. To substantiate this claim, here we conduct two-frame VFI experiments (i.e., using $I_0$ and $I_1$ to predict $I_{0.5}$).

### A. Implementation Details

We adopt polynomial interpolation (or linear interpolation) and set the number of sampling points for each warped pixel to be 25 in GDConv. The training dataset and the training strategy are described below.

*Training Dataset:* The Vimeo90 k interpolation training dataset [13] is used to train our model. This training dataset is composed of 51 312 triplets with resolution $256 \times 448$. We use the first frame and the third frame (corresponding to $I_0$ and $I_1$, respectively) of each triplet to predict the second one (corresponding to $I_{0.5}$). We randomly crop image patches of size $256 \times 256$ for training. Horizontal and vertical flipping, as well as temporal order reversal, are performed for data augmentation.

*Training Strategy:* This is the same as the four-frame case, except that we train our network for 20 epochs in total. The initial learning rate remains to be $10^{-3}$, and the learning rate is reduced by a factor of two every 4 epochs for the first 12 epochs and by a factor of five every 4 epochs for the last 8 epochs. The whole training process takes about 3 days on our hardware.

### B. Evaluation Datasets

Following [21], we evaluate the proposed GDConvNet on three public datasets (Vimeo90 k Interpolation Test Set [13],

TABLE IX
QUANTITATIVE COMPARISONS ON VIMEO90 K INTERPOLATION TEST SET, UCF101 DATASET AND MIDDLEBURY-OTHER DATASET, WHERE THE FIRST PLACE AND SECOND PLACE ARE HIGHLIGHTED IN RED AND BLUE, RESPECTIVELY

| Method | #Parameters | UCF101 | | Vimeo90K | | Middlebury |
|---|---|---|---|---|---|---|
| | (million) | PSNR | SSIM | PSNR | SSIM | IE |
| MIND | 7.60 | 33.93 | 0.9661 | 33.50 | 0.9429 | 3.35 |
| DVF | 3.80 | 34.12 | 0.9631 | 31.54 | 0.9462 | 7.75 |
| ToFlow | 1.07 | 34.58 | 0.9667 | 33.73 | 0.9682 | 2.51 |
| SepConv-Lf | 21.6 | 34.69 | 0.9655 | 33.45 | 0.9674 | 2.44 |
| SepConv-L1 | 21.6 | 34.78 | 0.9669 | 33.79 | 0.9702 | 2.27 |
| MEMC-Net | 70.3 | 34.96 | 0.9682 | 34.29 | 0.9739 | 2.12 |
| DAIN | 24.0 | **34.99** | **0.9683** | **34.71** | **0.9756** | **2.04** |
| AdaCoF | 21.8 | **34.99** | 0.9682 | 33.43 | 0.9677 | 2.43 |
| Ours | 5.6 | **35.16** | **0.9683** | **34.99** | **0.9750** | **2.03** |

UCF101 Test Dataset [53], and Middlebury-Other Dataset [54]) and compare it with the state-of-the-art.

*Vimeo90 k Interpolation Test Set [13]:* This dataset consists of 3782 video sequences, each with 3 frames. As in the case of the Vimeo90 K interpolation training dataset, the first frame and the third frame of each sequence are leveraged to synthesize the second one. The image resolution of this dataset is $256 \times 448$.

*UCF101 Test Dataset [53]:* The UCF101 dataset contains 379 triplets with a large variety of human actions. The image resolution of this dataset is $256 \times 256$.

*Middlebury-Other Dataset [54]:* The Middlebury-Other dataset is another commonly used benchmark for VFI, which contains 12 triplets in total. Most of the images in this dataset are of resolution $640 \times 480$. Again, we use the first frame and the third frame to predict the second one.

### C. Experimental Results

We compare our GDConvNet with the state-of-the-art VFI algorithms on the aforementioned datasets. Specifically, the following ones are chosen for comparison: MIND [55], DVF [19], SepConv [25], CtxSyn [30], ToFlow [13], SuperSlomo [20], MEMC-Net [22], DAIN [21], and AdaCoF [12].

In Table IX, we quantitatively compare our method with the state-of-the-art on Vimeo90 k and UCF101 under PSNR and SSIM, while Interpolation Error [44] (IE) is used as the performance measure for the Middlebury-Other dataset. It can be seen that the proposed method performs favorably against those under consideration. Overall, our method has a clear advantage under joint consideration of cost and performance. In particular, although DAIN [21] also shows very competitive performance, its model size is about 5 times that of our model. In addition, our method can be trained from scratch, while DAIN [21] needs to rely on a pre-trained model.

## VI. CONCLUSION

In this paper, a new mechanism named generalized deformable convolution is proposed to tackle the VFI problem. This mechanism unifies the essential ideas underlying flow-based and kernel-based methods and resolves some performance-limiting issues. It should be noted that the proposed mechanism is largely generic in nature, and is potentially applicable to a wide range of problems, especially those involving video data (e.g., video super-resolution, enhancement, and quality mapping). Exploring such applications is an endeavor well worth undertaking.

## ACKNOWLEDGMENT

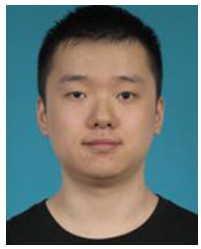
The authors would like to thank Prof. Duan and the anonymous reviewers for their valuable comments and suggestions. They also want to thank Prof. T. Davidson for proofreading the manuscript.

## REFERENCES

[1] G.-J. Qi, H. Larochelle, B. Huet, J. Luo, and K. Yu, "Guest editorial: Deep learning for multimedia computing," *IEEE Trans. Multimedia*, vol. 17, no. 11, pp. 1873–1874, Nov. 2015.
[2] X. Yang *et al.*, "DRFN: Deep recurrent fusion network for single-image super-resolution with large factors," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 328–337, Feb. 2019.
[3] Z. He *et al.*, "MRFN: Multi-receptive-field network for fast and accurate single image super-resolution," *IEEE Trans. Multimedia*, vol. 22, no. 4, pp. 1042–1054, Apr. 2020.
[4] X. Liu, K. Shi, Z. Wang, and J. Chen, "Exploit camera raw data for video super-resolution via hidden Markov model inference," *IEEE Trans. Image Process.*, to be published, doi: 10.1109/TIP.2021.3049974.
[5] P. Hu, G. Wang, and Y.-P. Tan, "Recurrent spatial pyramid CNN for optical flow estimation," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2814–2823, Oct. 2018.
[6] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8934–8943.
[7] C. Li *et al.*, "PDR-Net: Perception-inspired single image dehazing network with refinement," *IEEE Trans. Multimedia*, vol. 22, no. 3, pp. 704–716, Mar. 2020.
[8] Y. Song, J. Li, X. Wang, and X. Chen, "Single image dehazing using ranking convolutional neural network," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1548–1560, Jun. 2018.
[9] K. Zhu, R. Wang, Q. Zhao, J. Cheng, and D. Tao, "A cuboid CNN model with an attention mechanism for skeleton-based action recognition," *IEEE Trans. Multimedia*, vol. 22, no. 11, pp. 2977–2989, Nov. 2020.
[10] M. Usman *et al.*, "Frame interpolation for cloud-based mobile video streaming," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 831–839, May 2016.
[11] M. Ma, O. C. Au, L. Guo, S.-H. G. Chan, and P. H. Wong, "Error concealment for frame losses in MDC," *IEEE Trans. Multimedia*, vol. 10, no. 8, pp. 1638–1647, Dec. 2008.

[12] H. Lee *et al.*, "AdaCoF: Adaptive collaboration of flows for video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5316–5325.
[13] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.
[14] C. Zhang *et al.*, "Refined TV-l 1 optical flow estimation using joint filtering," *IEEE Trans. Multimedia*, vol. 22, no. 2, pp. 349–364, Feb. 2020.
[15] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Proc. Tech. Appl. Image Understanding*, vol. 281, 1981, pp. 319–331.
[16] B. D. Lucas *et al.*, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, 1981.
[17] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4161–4170.
[18] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[19] E. Ilg *et al.*, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2462–2470.
[20] H. Jiang *et al.*, "Super SloMo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9000–9008.
[21] W. Bao *et al.*, "Depth-aware video frame interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3703–3712.
[22] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang, "MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2019.2941941.
[23] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang, "Quadratic video interpolation," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019.
[24] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 670–679.
[25] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 261–270.
[26] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.
[27] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets v2: More deformable, better results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9308–9316.
[28] T.-Y. Lin *et al.*, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.
[29] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.
[30] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1701–1710.
[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[33] X. Liu, Y. Ma, Z. Shi, and J. Chen, "GriddehazeNet: Attention-based multi-scale network for image dehazing," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7314–7323.
[34] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.
[35] Y. Jeon and J. Kim, "Active convolution: Learning the shape of convolution for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4201–4209.
[36] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 667–675.
[37] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, J. M. Combes, A. Grossmann, and P. Tchamitchian, Eds., Berlin, Germany: Springer, 1990, pp. 286–297.
[38] H. Zhao *et al.*, "PSANet: Point-wise spatial attention network for scene parsing," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 267–283.
[39] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.
[40] T. Wang *et al.*, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 270–12 279.
[41] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "BAM: Bottleneck attention module," *Brit. Mach. Vis. Conf.*, 2018.
[42] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
[47] M. Jaderberg *et al.*, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[48] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4463–4471.
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int. Conf. Learn. Representations*, 2015.
[50] S. Nah, T. Hyun Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3883–3891.
[51] S. Su *et al.*, "Deep video deblurring for hand-held cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1279–1288.
[52] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1410–1418.
[53] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.
[54] S. Baker *et al.*, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.
[55] G. Long *et al.*, "Learning image matching by simply watching video," in *Proc. Eur. Conf. Comput. Vis*., 2016, pp. 434–450.

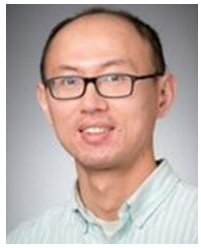
**Zhihao Shi** received the B.E. degree in communication engineering from Zhengzhou University, Zhengzhou, China, in 2018. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada. His research interests include image dehazing or deraining and video super-resolution.



**Xiaohong Liu** (Graduate Student Member, IEEE) received the B.E. degree in communication engineering from Southwest Jiaotong University, Chengdu, China, in 2014 and the M.A.Sc. degree in electrical and computer engineering from the University of Ottawa, Ottawa, ON, Canada, in 2016. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada. His research interests include video super-resolution, image dehazing or deraining, and image forgery detection. He was the recipient of the Ontario Graduate Scholarship in 2019, the NSERC Postgraduate Scholarship-Doctoral, and the Borealis AI Global Fellowship Award in 2020. He is a reviewer for several journals, including the IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE SIGNAL PROCESSING LETTERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON BROADCASTING, and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

**Kangdi Shi** received the B.E. degree in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2017. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada. His research interests include image classification, video super-resolution, and video coding.

**Linhui Dai** received the B.E. degree (Hons.) from Electrical Engineering, McMaster University, Hamilton, ON, Canada. where he is currently working toward the M.A.Sc. program with the Department of Electrical and Computer Engineering, McMaster University, supervised by Dr. Jun Chen. His research interests include image matting, low-level vision, and machine-learning algorithms. He was the recipient of the 2019 Vector Institute Scholarship.



**Jun Chen** (Senior Member, IEEE) received the B.E. degree in communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2004 and 2006, respectively. He was a Postdoctoral Research Associate with Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Champaign, IL, USA, from September 2005 to July 2006, and a Postdoctoral Fellow with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, from July 2006 to August 2007. Since September 2007, he has been with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada, where he is currently a Professor. His research interests include information theory, machine learning, wireless communications, and signal processing. Dr. Chen was the recipient of the Josef Raviv Memorial Postdoctoral Fellowship in 2006, the Early Researcher Award from the Province of Ontario in 2010, the IBM Faculty Award in 2010, the ICC Best Paper Award in 2020, and the JSPS Invitational Fellowship in 2021. He held the title of the Barber-Gennum Chair in Information Technology from 2008 to 2013 and the Joseph Ip Distinguished Engineering Fellow from 2016 to 2018. He was an Associate Editor for the the IEEETRANSACTIONS ON INFORMATION THEORYfrom 2014 to 2016. He is currently the Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.