MEMC-Net: Motion Estimation and Motion Compensation Driven Neural Network for Video Interpolation and Enhancement

Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang

Abstract—Motion estimation (ME) and motion compensation (MC) have been widely used for classical video frame interpolation systems over the past decades. Recently, a number of data-driven frame interpolation methods based on convolutional neural networks have been proposed. However, existing learning based methods typically estimate either flow or compensation kernels, thereby limiting performance on both computational efficiency and interpolation accuracy. In this work, we propose a motion estimation and compensation driven neural network for video frame interpolation. A novel adaptive warping layer is developed to integrate both optical flow and interpolation kernels to synthesize target frame pixels. This layer is fully differentiable such that both the flow and kernel estimation networks can be optimized jointly. The proposed model benefits from the advantages of motion estimation and compensation methods without using hand-crafted features. Compared to existing methods, our approach is computationally efficient and able to generate more visually appealing results. Furthermore, the proposed MEMC-Net architecture can be seamlessly adapted to several video enhancement tasks, e.g., super-resolution, denoising, and deblocking. Extensive quantitative and qualitative evaluations demonstrate that the proposed method performs favorably against the state-of-the-art video frame interpolation and enhancement algorithms on a wide range of datasets.

Index Terms—Motion Estimation, Motion Compensation, Convolutional Neural Network, Adaptive Warping

✦

1 INTRODUCTION

VIDEO frame interpolation aims to synthesize non-existent frames between original input frames, which has been applied to numerous applications such as video frame rate conversion [1], novel view synthesis [2], and frame recovery in video streaming [3], to name a few. Conventional approaches [4], [5] are generally based on motion estimation and motion compensation (MEMC), and have been widely used in various display devices [6]. A few deep learning based frame interpolation approaches [7], [8] have been developed to address this classical topic. In this paper, we analyze the MEMC-based and learning-based approaches of video frame interpolation and exploit the merits of both paradigms to propose a high-quality frame interpolation processing algorithm.

Conventional MEMC-based approaches entail both motion estimation [13] and motion compensation [14] for video interpolation. Motion estimation is used to determine the block-wise or pixel-wise motion vectors between two frames. The block-based methods [4] assume that the pixels within a block share the same motion and use search strategies [15], [16] and selection criteria [4], [17] to obtain the optimal motion vector. On the other hand, the methods based on pixel-based motion, i.e., optical flow, estimate a motion/flow vector for each pixel of the frames and thus entail heavy computational loads. The recent years have witnessed

significant advances in optical flow estimation via variational optimization [18], nearest neighbor field search [19], cost volume filtering [20], and deep convolutional neural networks (CNNs) [11], [21]. However, estimating optical flow remains a challenging problem due to fast-moving and thin objects, occlusion and dis-occlusion, brightness change and motion blur. To account for inaccurate flow and occluded pixels, motion compensated interpolation methods usually use sophisticated filters to reduce visual artifacts of the generated frames [14], [22]. In addition, these schemes do not perform well where objects in the intermediate frame are invisible in both forward and backward reference frames (e.g., some pixels cannot be compensated), and require further post-processing procedures to fill in missing or remove unreliable pixels [17], [23], [24], [25].

Numerous learning-based frame interpolation methods based on deep CNNs have been recently proposed [7], [8]. The training datasets for learning-based methods typically contain image triplets from raw video sequences, with the first and third frame feeding into the network as inputs and the intermediate second frame acting as ground truth [7], [8], [9] for output. By imposing loss functions such as L_p-norm on the difference between the network output and ground truth frame pixels, the model parameters can be iteratively updated via a gradient descent scheme.

The conventional MEMC-based methods are computationally efficient due to the block-wise setting [25], [26]. However, these block-based methods do not achieve the state-of-the-art results as hand-crafted features are typically used in the ME and MC stages. In contrast, the learning-based methods are developed based on the massive amount of raw video data. However, the state-of-the-art learning-

• Wenbo Bao, Xiaoyun Zhang and Zhiyong Gao are with the Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240. Email: {baowenbo|xiaoyun.zhang|zhiyong.gao}@sjtu.edu.cn
• Wei-Sheng Lai and Ming-Hsuan Yang are with the Department of Electrical Engineering and Computer Science, University of California, Merced, CA, 95340. Email:{wlai24|mhyang}@ucmerced.edu
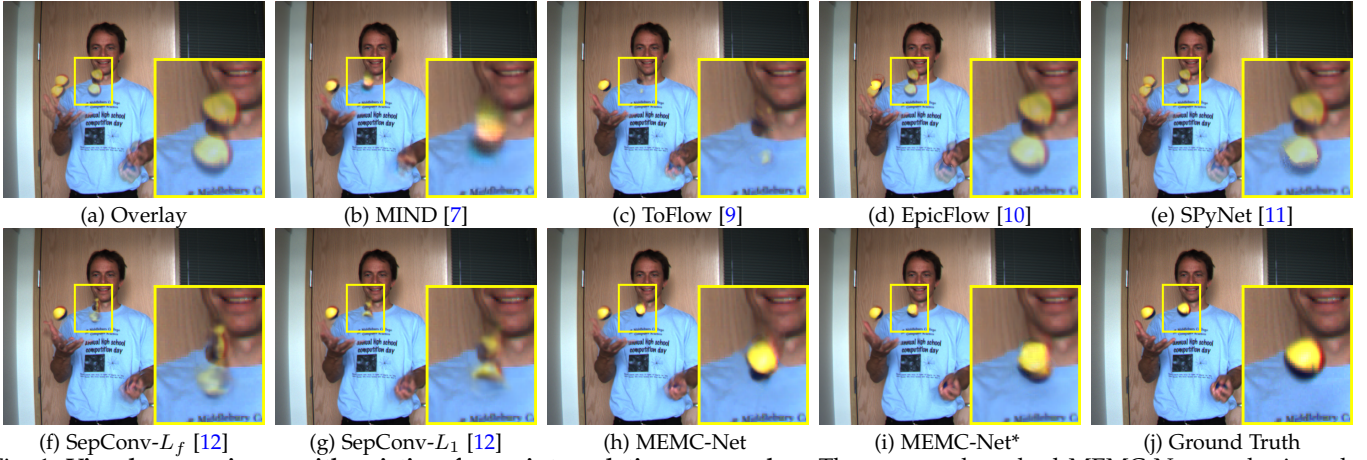
Fig. 1. **Visual comparisons with existing frame interpolation approaches.** The proposed method MEMC-Net synthesizes the intermediate frame with clear edges and shape. With the context information and residual blocks used, the improved model MEMC-Net* obtains better outcome with fine details around motion boundaries.

based approaches [9], [27] focus on motion estimation, which often leads to blurry results due to bilinear interpolation process. While other approaches [12], [28] are developed to consider the effect of interpolation kernels, such schemes are sensitive to large motion.

In this paper, we propose to exploit motion estimation and motion compensation in a neural network for video frame interpolation. Both the motion vectors and compensation filters are estimated through CNNs. We further propose an adaptive warping layer based on optical flow and compensation filters for synthesizing new pixels. This novel warping layer is fully differentiable such that the gradients can be back-propagated to both the ME and MC networks. To account for the occlusions, we estimate occlusion masks to adaptively blend the warped frames. Furthermore, the missing pixels in holes and unreliable pixels of the warped frames are processed by a post-processing CNN. Our entire model, MEMC-Net, is motivated by the architecture of conventional methods but realized via the most recent learning-based approaches. Fig. 1 shows an interpolated frame of our methods (MEMC-Net and MEMC-Net*) and existing algorithms [7], [9], [10], [11], [12], where the proposed methods predict the moving ball with clearer contours and sharper edges.

The contributions of this paper are summarized as follows:

(1) We propose a motion estimation and compensation driven neural network for robust and high-quality video frame interpolation.
(2) We integrate the optical flow warping with learned compensation filters into an *adaptive warping layer*. The proposed adaptive warping layer is fully differentiable and applicable to several video processing tasks, e.g., video super-resolution, video denoising, and video deblocking.
(3) We demonstrate that the proposed method performs favorably against the state-of-the-art frame interpolation algorithms on several benchmark datasets, including the Middlebury [29], UCF101 [30], and Vimeo90K [9] datasets. Our model requires less memory to predict the compensation filters and executes efficiently.

(4) We extend our network to the other video enhancement tasks including super-resolution, denoising, and deblocking as the model is general and applicable to motion compensation based tasks. Our methods obtain more favorable results against the state-of-the-art algorithms on each of these tasks.

2 RELATED WORK

In this section, we discuss the conventional MEMC-based and recent learning-based methods.

2.1 Conventional MEMC-based Methods

Fig. 2(a) shows the typical framework of conventional MEMC-based video frame interpolation methods. First, motion vectors between the forward and reference frames are estimated. Along the motion trajectories, pixels of the reference frames are used to interpolate the intermediate frame. Conventional ME methods use block-based algorithms such as the 3D recursive search [4], which are hardware-friendly and computationally efficient. The block-based methods typically divide the image frames into small pixel blocks and exploit certain search strategies such as spatial/temporal search [4], hierarchical search [31], based on selection criteria such as the minimum sum of absolute block difference to compute their motion vectors.

For motion compensated interpolation, overlapped blocks are usually utilized to cope with the erroneous motion vectors of pixel blocks [14]. Recently, several methods [24], [32] exploit optical flow for the truthfulness of flow fields. Compensation filters via image fusion [24] or overlapped patch reconstruction [32] are developed to deal with occlusion or blocky effects.

Aside from the motion estimation and motion compensation procedures, a post-processing step is often required to minimize artifacts and improve visual qualities [17], [23], [24], [25]. Due to relative motions and occlusion between objects with different depth, the estimated flow vectors may lead to incorrect interpolation results with hole regions. Kim et al. [25] utilize a hole interpolation method to restore missing pixels. On the other hand, Wang et al. [17] propose
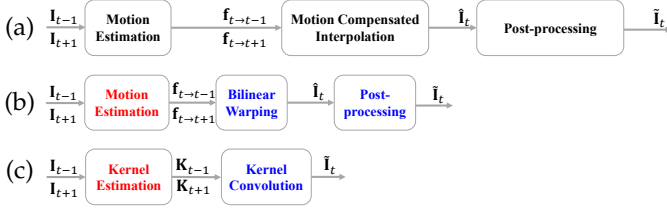
Fig. 2. **Frameworks of (a) the conventional MEMC-based approaches, (b) the flow-based and (c) the kernel-based models.** The **black**, red, and blue text boxes correspond to the conventional modules, network modules, and network layers respectively.

Fig. 3. **Frameworks of (a) the sequential MEMC-Net model and (b) our proposed MEMC-Net model.**

TABLE 1. **CNN-based frame interpolation methods**.

	Optical flow	Occlusion mask	Interpolation coefficients	Kernel size
Flow-based [9], [36]	✓	✓	fixed	2×2
Kernel-based [12], [28]	—	—	adaptive	41×41, 51×51
MEMC-Net (Ours)	✓	✓	adaptive	4×4

a trilateral filtering method to fill the holes and smooth the compensation errors in both the spatial and temporal domains. The proposed algorithm differs from the conventional MEMC methods in that we develop a data-driven end-to-end trainable model with deep features.

2.2 Learning-based Methods

Video frame interpolation based on deep learning algorithms can be categorized into the direct method, phase-based, flow-based, and kernel-based approaches. Long et al. [7] train a deep CNN to directly predict the interpolated frames. The outputs are usually blurry and contain fewer details as this deep model is not able to capture the multi-modal distribution of natural images and videos. The phase-based method [33] manipulates the pixel phase information within a multi-scale pyramid for frame interpolation. However, this approach is less effective in handling large motion in complicated scenes. In the following, we focus our discussion on recent flow-based and kernel-based methods.

Flow-based methods. With the advances in optical flow estimation by deep CNNs [11], [21], [34], [35], several methods based on end-to-end deep models have been developed for frame interpolation. These approaches either predict bidirectional flow [9] or use the bilinear warping operation to align input frames based on linear motion models [36], [37], [38]. To synthesize an output image, a common technique is to estimate an occlusion mask to adaptively blend the warped frames. As the bilinear warping blend neighbor pixels based on the sub-pixel shifts, the flow-based methods inevitably generate ghost or blurry artifacts when the input frames are not aligned well. The pipeline for the flow-based methods is illustrated in Fig. 2(b). Instead of using the fixed bilinear coefficients for interpolation, our approach learns spatially-varying interpolation kernels for each pixel. The learned kernels have larger spatial support (e.g., 4×4) than the bilinear interpolation and thus better account for occlusion and dis-occlusion.

Kernel-based methods. Instead of relying on pixel-wise optical flow, frame interpolation can be formulated as convolution operations over local patches [39], [40]. Niklaus et al. [28] propose the AdaConv model to estimate spatially-adaptive convolutional kernels for each output pixel. We show the pipeline of kernel-based methods in Fig. 2(c). In these methods, a large kernel size is used
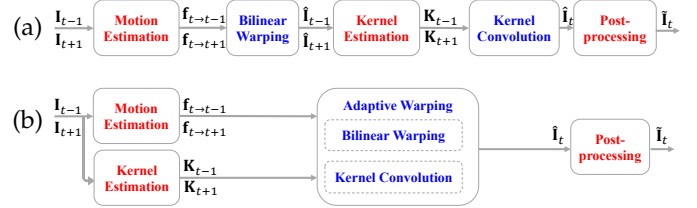
to handle large motion, which requires a large amount of memory to process high-resolution images. For an input frame of $H \times W$ pixels, the AdaConv model needs to estimate $H \times W \times R \times R$ coefficients for interpolation, where R is the size of the local kernels. To reduce memory requirements, the SepConv method [12] assumes that the convolutional kernels are separable and uses a pair of 1D kernels (one vertical and one horizontal kernel) to approximate the 2D kernels. This strategy significantly reduces the memory consumption from $O(R^2)$ to $O(2R)$ and further improves interpolation results. However, both the AdaConv and SepConv methods cannot handle motion larger than the *pre-defined* kernel size. While our approach also learns adaptive local kernels for interpolation, the proposed method is not limited by the assumption of *fixed* motion range as optical flow warping is integrated. That is, our method uses smaller kernels, requires a low amount of memory, and performs robustly to frames with large motion. We list the main difference with flow-based methods [9], [36] and kernel-based approaches [12], [28] in Table 1.

3 MOTION ESTIMATION AND MOTION COMPENSATION DRIVEN NEURAL NETWORK

In this section, we describe the design methodology of the proposed MEMC-Net framework, adaptive warping layer, and flow projection layer used in our model.

3.1 MEMC-Net Framework

Following the conventional MEMC-based and recent learning-based methods, there are different ways to design a MEMC-Net model for video frame interpolation.

A straightforward method is to combine the motion estimation, motion compensation, and post-processing sequentially. That is, the reference frames are first aligned with the motion estimation, bilinear warping is applied to account for large motion, and small convolutional kernels for the warped frames are estimated to synthesize a final frame. As in the conventional MEMC-based framework, a post-processing network is also added to the sequential model to reduce the possible pixel outliers. Fig. 3(a) illustrates this
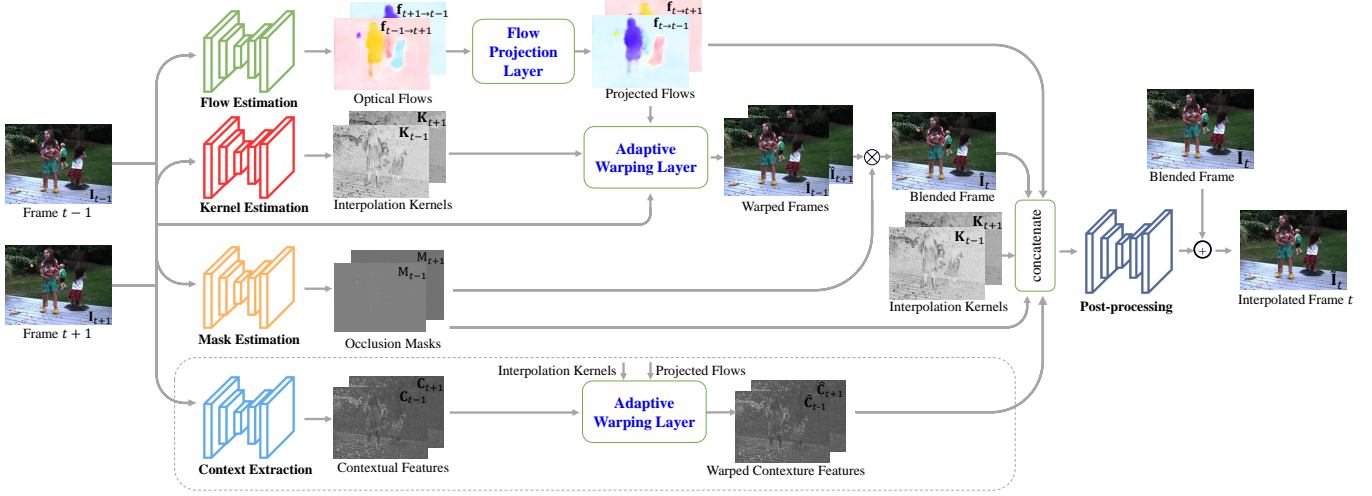
Fig. 4. **Network architecture of the proposed MEMC-Net and MEMC-Net*.** The context extraction module and its generated contextual features and warped contextual features are for MEMC-Net*.

sequential model. However, according to our experiments, the warped frames ($\hat{\mathbf{I}}_{t-1}$ and $\hat{\mathbf{I}}_{t+1}$) are usually of low quality due to the imperfect optical flow estimated by existing methods. Consequently, the lateral kernel estimation, kernel convolution, and post-processing are not able to generate visually pleasing results from the corrupted frames.

In this paper, we develop a novel algorithm to simultaneously estimate the flow and compensation kernels with respect to the original reference frames. This approach requires frame interpolation to be carried out within a warping layer based on both the flow and compensation kernel. This new warping layer is expected to tightly couple the motion estimation and kernel estimation networks so that both networks can be optimized through the enormous video data. Fig. 3(b) shows the proposed framework for video frame interpolation.

We present the network architecture of the proposed MEMC-Net for the video frame interpolation in Fig. 4. In this work, we propose a novel *adaptive warping layer* to assemble the bilinear warping and kernel convolution in one single step. The layer takes in the optical flow, interpolation kernel to warp the input frame pixels. For the video frame interpolation task, since the intermediate frame is not originally available, we estimate the flow between the forward and backward reference frames, and then project it to simulate the flow between the intermediate and reference frames. This operation is achieved by our proposed *flow projection layer*.

The adaptive warping and the flow projection layers are the two major technical innovations of our algorithm. We summarize the benefits of the proposed layer from two aspects. First, the conventional MEMC-based approaches rely on hand-crafted features (e.g., SIFT [41] for motion estimation or Gaussian-like weighting maps [42] for motion compensation), while the proposed adaptive warping layer allows us to extract *data-driven* features for joint motion estimation and motion compensation. Therefore, the proposed model has a better *generalization capability* to handle various scenarios for video frame interpolation and enhancement tasks. Second, the adaptive warping layer tightly integrates

two learning-based methodologies, namely the flow-based and kernel-based ones, and inherits their merits in that:

1) Compared to the flow-based methods [9], [36] that rely on simple bilinear coefficients, our method is able to improve the *interpolation accuracy* by using data-driven kernel coefficients.

2) Compared to the kernel-based approaches [12], [28], our method obtains higher *computational efficiency* by largely reducing the kernel size through pre-aligning the pixels with learned optical flows.

We present the forward inference and back-propagation details of the two novel layers in Section 3.2 and Section 3.3 respectively. The pipeline of our method in Fig. 4, as well as the detailed network configuration of the used motion estimation, kernel estimation, mask estimation, and post-processing networks, are described in Section 4. We will also introduce an additional context extraction network toward the enhanced MEMC-Net* model.

3.2 Adaptive Warping Layer

The proposed adaptive layer warps images or features based on the given optical flow and local convolutional kernels.

Forward pass. Let $\mathbf{I}(\mathbf{x}) : \mathbb{Z}^2 \rightarrow \mathbb{R}^3$ denote the RGB image where $\mathbf{x} \in [1, H] \times [1, W]$, $\mathbf{f}(\mathbf{x}) := (u(\mathbf{x}), v(\mathbf{x}))$ represent the optical flow field and $\mathbf{k}^l(\mathbf{x}) = [k_{\mathbf{r}}^l(\mathbf{x})]_{H \times W}$ ($\mathbf{r} \in [-R+1, R]^2$) indicate the interpolation kernel where R is the kernel size. The adaptive warping layer synthesizes an output image by:

$$\hat{\mathbf{I}}(\mathbf{x}) = \sum_{\mathbf{r} \in [-R+1, R]^2} k_{\mathbf{r}}(\mathbf{x}) \mathbf{I}(\mathbf{x} + \lfloor \mathbf{f}(\mathbf{x}) \rfloor + \mathbf{r}), \quad (1)$$

where the weight $k_{\mathbf{r}} = k_{\mathbf{r}}^l k_{\mathbf{r}}^d$ is determined by both the learned interpolation kernel $k_{\mathbf{r}}^l$ and bilinear coefficient $k_{\mathbf{r}}^d$. We train a kernel estimation network to predict the weights for the interpolation kernels. For each 2D spatial location in the image grid $[1, W] \times [1, H]$, the kernel estimation network generates a 16-channel feature vector. We then map the feature vector into a 4×4 square matrix as the
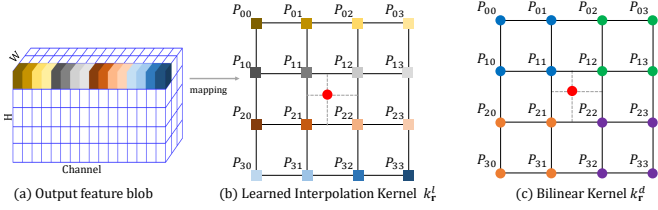
Fig. 5. **Learned interpolation kernel** $k_{\mathbf{r}}^l$ **and bilinear kernel** $k_{\mathbf{r}}^d$. The $k_{\mathbf{r}}^l$ is re-organized from the output feature blob generated by kernel estimation network.

kernel coefficients for sampling the local patch. As shown in Fig. 5(a) and (b), the colors on the feature vector and the patch pixels show the mapping of the 16 channels. The red point in Fig. 5(b) indicates the sub-pixel location shifted by the optical flow.

On the other hand, the bilinear coefficient (Fig. 5(c)) is defined by:

$$
k_{\mathbf{r}}^d = \begin{cases} [1 - \theta(u)][1 - \theta(v)], & \mathbf{r}_u \le 0, \mathbf{r}_v \le 0, \\ \theta(u)[1 - \theta(v)], & \mathbf{r}_u > 0, \mathbf{r}_v \le 0, \\ [1 - \theta(u)]\theta(v), & \mathbf{r}_u \le 0, \mathbf{r}_v > 0, \\ \theta(u)\theta(v), & \mathbf{r}_u > 0, \mathbf{r}_v > 0, \end{cases} \tag{2}
$$

where $\theta(u) = u - \lfloor u \rfloor$ denotes the fractional part of a float point number, and the subscript u, v of the 2-D vector $\mathbf{r}$ represent the horizontal and vertical components, respectively. The bilinear coefficient allows the layer to back-propagate the gradients to the optical flow estimation network. In this case, we aim to compute a local interpolation kernel that combines the bilinear coefficients and the learned coefficients from the kernel prediction network. To apply the bilinear coefficients to kernels of any size, we first compute the bilinear coefficients for the nearest four neighbor pixels, i.e., P_{11}, P_{12}, P_{21}, and P_{22}, and then replicate the coefficients to the pixels at the same corner. Therefore, the pixels with the same color in Fig. 5(c) have the same bilinear coefficient. Finally, we multiply the bilinear coefficients with the learned kernel coefficients as our local adaptive kernels.

Backward pass. We compute the gradient with respect to the optical flow and interpolation kernels, respectively. The derivative with respect to the optical flow field $\mathbf{f}$ is computed by (using the horizontal component u for example):

$$
\frac{\partial \hat{\mathbf{I}}(\mathbf{x})}{\partial u(\mathbf{x})} = \sum_{\mathbf{r}} k_{\mathbf{r}}^l(\mathbf{x}) \cdot \mathbf{I}\left(\mathbf{x} + \lfloor \mathbf{f}(\mathbf{x}) \rfloor + \mathbf{r}\right) \cdot \frac{\partial k_{\mathbf{r}}^d}{\partial u}, \tag{3}
$$

where

$$
\frac{\partial k_{\mathbf{r}}^d}{\partial u} = \begin{cases} -[1 - \theta(v)], & \mathbf{r}_u \le 0, \mathbf{r}_v \le 0, \\ [1 - \theta(v)], & \mathbf{r}_u > 0, \mathbf{r}_v \le 0, \\ -\theta(v), & \mathbf{r}_u \le 0, \mathbf{r}_v > 0, \\ \theta(v), & \mathbf{r}_u > 0, \mathbf{r}_v > 0. \end{cases} \tag{4}
$$

The derivative with respect to the vertical component v can be derived in a similar way.

The derivative with respect to the interpolation kernel $k_{\mathbf{r}}^l$ is:

$$
\frac{\partial \hat{\mathbf{I}}}{\partial k_{\mathbf{r}}^l(\mathbf{x})} = k_{\mathbf{r}}^d(\mathbf{x}) \cdot \mathbf{I}\left(\mathbf{x} + \lfloor \mathbf{f}(\mathbf{x}) \rfloor + \mathbf{r}\right). \tag{5}
$$

The integration with the spatially-varying kernels alleviates the limitation of bilinear interpolation to synthesize pixel values from a broader neighborhood. In addition, this approach facilitates the warping layer to perform more robustly to inaccurate optical flow and better account for occlusion.

3.3 Flow Projection Layer

As the intermediate frame is not available, we transform the flow between the forward and backward reference frames and then project it to simulate the flow between the intermediate frame and the reference frames. Let $\mathbf{f}_{t \to t-1}(\mathbf{x})$ be the motion vector field of frame $\mathbf{I}_t$ to $\mathbf{I}_{t-1}$. Similarly, $\mathbf{f}_{t-1 \to t+1}(\mathbf{y})$ represents the motion vector field of frame $\mathbf{I}_{t-1}$ to $\mathbf{I}_{t+1}$. Note that we use $\mathbf{y}$ to index the 2-D coordinate at time step $t-1$, as distinguished to $\mathbf{x}$ at t. Our flow projection layer is designed to transform an estimated flow $\mathbf{f}_{t-1 \to t+1}(\mathbf{y})$ to $\mathbf{f}_{t \to t-1}(\mathbf{x})$. Here we assume that the local motion between consecutive frames is linear and invert the flow between $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t+1}$ to approximate the intermediate flow fields.

As there may exist multiple flow vectors projected to the same location in the intermediate frame, we average all the projected flow vectors at the same location. On the other hand, there may exist holes where no flow is projected. Thus, we use the outside-in strategy [29] to fill-in these holes in the intermediate frame. We denote the set of flow vectors mapped to location $\mathbf{x}$ of time step t by $\mathcal{S}(\mathbf{x}) := \{\mathbf{y} : \text{round}\left(\mathbf{y} + \mathbf{f}_{t-1 \to t+1}(\mathbf{y})/2\right) = \mathbf{x}, \forall\, \mathbf{y} \in [1, H] \times [1, W]\}$ and denote the 4-directional nearest available flow vectors of a hole by $\mathcal{N}(\mathbf{x}) := \{\mathbf{x}' : |\mathcal{S}(\mathbf{x}')| > 0\}$. The forward pass of the proposed projection layer is defined by:

$$
\mathbf{f}_{t \to t-1}(\mathbf{x}) = \begin{cases} \dfrac{-1}{|\mathcal{S}(\mathbf{x})|} \displaystyle\sum_{\mathbf{y} \in \mathcal{S}(\mathbf{x})} \dfrac{\mathbf{f}_{t-1 \to t+1}(\mathbf{y})}{2}, & \text{if } |\mathcal{S}(\mathbf{x})| > 0, \\ \dfrac{1}{|\mathcal{N}(\mathbf{x})|} \displaystyle\sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \mathbf{f}_{t \to t-1}(\mathbf{x}'), & \text{if } |\mathcal{S}(\mathbf{x})| = 0. \end{cases} \tag{6}
$$

The backward pass computes the derivative with respect to the input optical flow $\mathbf{f}_{t-1 \to t+1}(\mathbf{y})$:

$$
\frac{\partial \mathbf{f}_{t \to t-1}(\mathbf{x})}{\partial \mathbf{f}_{t-1 \to t+1}(\mathbf{y})} = \begin{cases} \dfrac{-1}{2|\mathcal{S}(\mathbf{x})|}, & \text{for } \mathbf{y} \in \mathcal{S}(\mathbf{x}) \text{ if } |\mathcal{S}(\mathbf{x})| > 0, \\ 0, & \text{for } \mathbf{y} \notin \mathcal{S}(\mathbf{x}) \text{ or } |\mathcal{S}(\mathbf{x})| = 0. \end{cases} \tag{7}
$$

We use a graph to illustrate the outside-in strategy in Fig. 6. We use a *soft* blending way in the proposed flow projection layer by averaging the 4-directional available flow vectors from the neighboring non-hole regions. The spatial position at X has its 4-directional non-hole neighbors A, B, C, and D. Therefore, the flow vector $\mathbf{f}_{\mathrm{X}}$ is approximated by $\mathbf{f}_{\mathrm{X}} = (\mathbf{f}_{\mathrm{A}} + \mathbf{f}_{\mathrm{B}} + \mathbf{f}_{\mathrm{C}} + \mathbf{f}_{\mathrm{D}})/4$. An alternative is to fill in the flow holes with zero vectors, which is only suitable for stationary objects. We show an example in Fig. 7 to compare the two strategies. The outside-in strategy can reduce interpolation artifacts significantly.
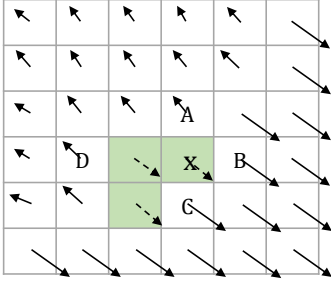
Fig. 6. **Outside-in strategy for filling the flow holes.** The green regions indicate a hole, where the flow vectors are approximated by the average of 4-directional available flow vectors from the non-hole regions.

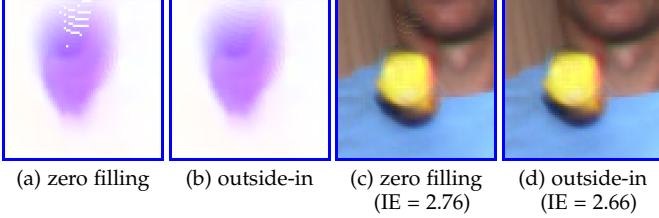| (a) zero filling | (b) outside-in | (c) zero filling (IE = 2.76) | (d) outside-in (IE = 2.66) |

Fig. 7. **Effectiveness of the used outside-in strategy for hole filling.** (a) and (b) are the flow maps by zero filling and outside-in strategy. (c) and (d) are the generated frames by them. Less artifact is generated by the outside-in hole filling strategy. IE is short for interpolation error. The lower, the better.

4 VIDEO FRAME INTERPOLATION

We provide an overview of the proposed MEMC-Net in Fig. 4 and describe the detailed architecture design of each component below.

Motion estimation. Given two input frames $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t+1}$, we first estimate the forward flow $\mathbf{f}_{t-1 \rightarrow t+1}$ and backward flow $\mathbf{f}_{t+1 \rightarrow t-1}$ by passing $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t+1}$ into the flow estimation network twice with a reverse order. In this work, we use the FlowNetS [21] model for optical flow estimation. Then we use the proposed flow projection layer as described by Eq.(6) to project the forward flow $\mathbf{f}_{t-1 \rightarrow t+1}$ and backward flow $\mathbf{f}_{t+1 \rightarrow t-1}$ into $\mathbf{f}_{t \rightarrow t-1}$ and $\mathbf{f}_{t \rightarrow t+1}$ for the intermediate frame, respectively.

Kernel estimation. We use the U-Net [43] as our kernel estimation network, which has an encoder with five max-pooling layers, a decoder with five un-pooling layers, and skip connections from the encoder to the decoder. The kernel prediction network takes two video frames as input and generates R^2 coefficient maps, denoted by $\mathbf{K}_{t-1}$ and $\mathbf{K}_{t+1}$. We then reshape the coefficient maps to $R \times R$ convolutional kernels for each pixel, as shown in Fig. 5(b). Two pairs of intermediate flow and the kernel coefficients, $\{\mathbf{f}_{t \rightarrow t-1}, \mathbf{K}_{t-1}\}$ and $\{\mathbf{f}_{t \rightarrow t+1}, \mathbf{K}_{t+1}\}$ are then fed into the proposed adaptive warping layer to warp the input frames by Eq. (1) and generate two warped frames $\hat{\mathbf{I}}_{t-1}$ and $\hat{\mathbf{I}}_{t+1}$.

Mask estimation. Due to the depth variation and relative motion of objects, there are occluded pixels between the two reference frames. To select valid pixels from the two warped reference frames, we learn a mask estimation network to predict the occlusion masks. The mask estimation network has the same U-Net architecture as our kernel
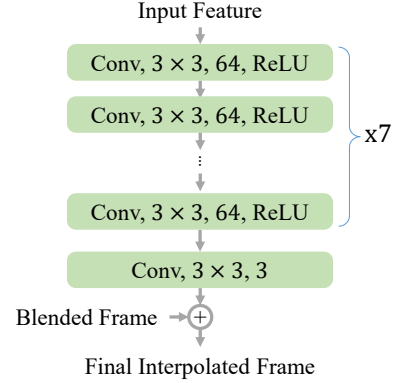

Fig. 8. **Proposed post-processing network.**

estimation network, but the last convolutional layer outputs a 2-channel feature map as the occlusion masks $\mathbf{M}_{t-1}$ and $\mathbf{M}_{t+1}$. The blended frame is generated by:

$$\hat{\mathbf{I}}_t = \mathbf{M}_{t-1} \otimes \hat{\mathbf{I}}_{t-1} + \mathbf{M}_{t+1} \otimes \hat{\mathbf{I}}_{t+1}, \tag{8}$$

where $\otimes$ denotes the channel-wise multiplication operation.

Context extraction. We also use the contextual information [27] in the post-processing module to better deal with occlusion. We extract the *conv1* features of the input reference frames from a pre-trained ResNet18 [44] as the contextual maps. The contextual maps are then warped by the optical flow and the interpolation kernels via the adaptive warping layer. The warped contextual maps, denoted as $\hat{\mathbf{C}}_{t-1}$ and $\hat{\mathbf{C}}_{t+1}$, are fed as inputs to the following post-processing .

Post-processing. Since the blended image $\hat{\mathbf{I}}_t$ usually contains artifacts caused by inaccurate flow estimation or masks, we introduce a post-processing network to improve the visual quality. The post-processing module takes as input the blended frame $\hat{\mathbf{I}}_t$, estimated flows $\mathbf{f}_{t \rightarrow t+1}$ and $\mathbf{f}_{t \rightarrow t-1}$, coefficient maps of the interpolation kernels $\mathbf{K}_{t-1}$ and $\mathbf{K}_{t+1}$, occlusion masks $\mathbf{M}_{t-1}$ and $\mathbf{M}_{t+1}$, and the warped context features $\hat{\mathbf{C}}_{t-1}$ and $\hat{\mathbf{C}}_{t+1}$. Our post-processing network contains 8 convolutional layers as shown in Fig. 8. Except for the last one, each convolutional layer has a filter size of 3×3 with 64 output channels and is followed by a Rectified Linear Unit (ReLU). The last convolutional layer outputs a 3-channel RGB image. As the output and input of this module are highly similar (i.e., both are the interpolated frame at t), we enforce the network to output the *residual* between the blended frame $\hat{\mathbf{I}}_t$ and the ground-truth frame. Therefore, the post-processing module learns to enhance the details and remove the artifacts in the blended frame. We present an example in Fig. 9 to demonstrate the effect of the post-processing module. The blurry edges and lines are sharpened by our method. The proposed model generates $\tilde{\mathbf{I}}_t$ as the final interpolated frame.

MEMC-Net. We provide two variants of the proposed model. The first one does not use the contextual information, where we refer the model as *MEMC-Net*. The second one includes the context information as the inputs to the post-processing module, where we refer the model as *MEMC-Net**.
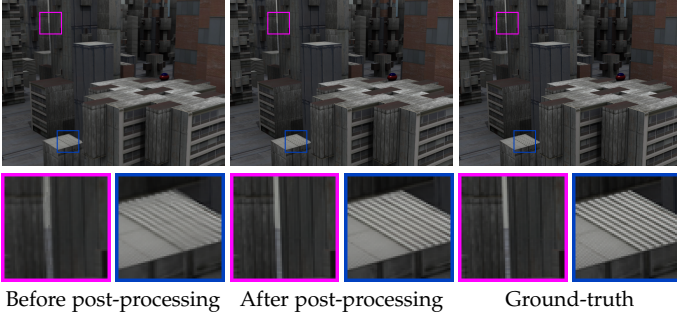
Before post-processing   After post-processing   Ground-truth

Fig. 9. **Effectiveness of the proposed post-processing network.**

(a) Frame enhancement network   (b) Residual block

Fig. 10. **Network architecture for frame enhancement.**

Fig. 11. **Network architecture for video frame super-resolution.**

5 IMPLEMENTATION DETAILS

In this section, we discuss the implementation details including the loss function, datasets, and hyper-parameter settings of the proposed MEMC-Net and MEMC-Net*.

Loss Function. We use a robust loss function between the restored frames $\tilde{\mathbf{I}}_t$, $\hat{\mathbf{I}}_t$ and the corresponding ground truth frame $\mathbf{I}_t^{GT}$. We also regularize the sum of two masks to be 1.0. The combined loss function is given by:

$$\mathcal{L} = \sum_{\mathbf{x}} \Phi\left(\tilde{\mathbf{I}}_t - \mathbf{I}_t^{GT}\right) + \alpha \sum_{\mathbf{x}} \Phi\left(\hat{\mathbf{I}}_t - \mathbf{I}_t^{GT}\right)$$
$$+ \beta \sum_{\mathbf{x}} \Phi\left(\mathrm{M}_{t-1} + \mathrm{M}_{t+1} - 1.0\right) \tag{9}$$

where $\Phi(x) = \sqrt{x^2 + \epsilon^2}$ is the Charbonnier penalty function [45] with ϵ to be $1e-6$. We empirically set α and β to be $1e-3$ and $2e-3$ respectively.

Datasets. We use the training set of the Vimeo90K dataset [9] to learn the proposed frame interpolation model. There are 51,312 triplets, and each image is of 448×256 pixels. During the training process, we use the data augmentation with random horizontal and vertical flipping as well as reversing the temporal order of input sequences.

Hyper-parameter settings. We initialize the network parameters with the method of He et al. [46]. We set the initial learning rate of the kernel prediction, mask estimation and post-processing networks to be 0.001 while using a smaller learning rate of 0.00001 for fine-tuning the flow estimation network. We decrease the learning rates by a factor of 0.2 if the validation loss does not decrease during 5 epochs. We use a batch size of 4 and use the Adam [47] optimizer with β_1 of 0.9 and β_2 of 0.999 for training our model. In addition, we use a weight decay of $1e-6$. The entire network is trained for 100 epochs. Except for the last output layer, the convolutional layers of the FlowNetS network are activated by the leaky ReLU [48], while those of the other three networks are activated by the ReLU [49] layer. We use the batch normalization [50] layer in the kernel prediction and mask estimation networks.

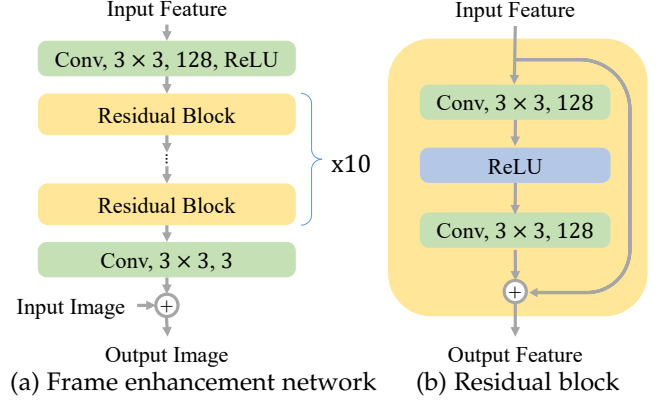The source code, trained model, and video frame interpolation results generated by all the evaluated methods are available on our project website: https://sites.google.com/view/wenbobao/memc-net.
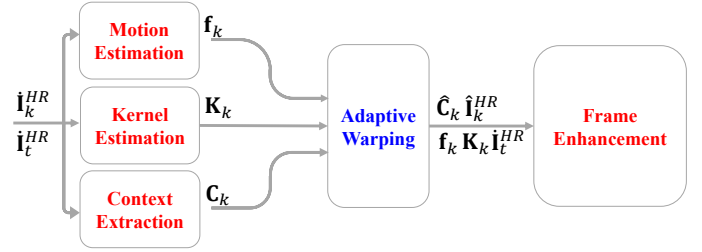
6 VIDEO FRAME ENHANCEMENT

In addition to video frame interpolation, we show that the proposed framework can be generalized to several video frame enhancement tasks, including video super-resolution, video denoising, and video deblocking. In these tasks, multiple consecutive frames are used to extract useful texture cues to reduce the distortions like low-resolution, noise, blockiness, etc. Existing learning-based methods [9], [51] often align the consecutive frames based on the estimated optical flow and bilinear interpolation. However, as discussed in Section 3.1 and 4, the bilinear interpolation process may result in blurred pixels. In contrast, the proposed adaptive warping layer is able to compensate for more accurate frame pixels.

Here we discuss how the proposed method can be extended for the video super-resolution problem. Given $2L+1$ consecutive low-resolution frames $\{\mathbf{I}_k^{LR}\}_{k=t-L}^{t+L}$, our goal is to recover a high-resolution frame $\mathbf{I}_t^{HR}$ at the time step t. We first use bicubic interpolation to up-sample all the low-resolution frames to the target resolution, denoted by $\{\dot{\mathbf{I}}_k^{HR}\}_{k=t-L}^{t+L}$. For each pair of $\dot{\mathbf{I}}_k^{HR}(k \neq t)$ and $\dot{\mathbf{I}}_t^{HR}$, we estimate the optical flow $\mathbf{f}_k(k \neq t)$ and compensation kernel $\mathbf{K}_k(k \neq t)$ via our flow estimation and kernel estimation networks. Then, we use the proposed adaptive warping layer to warp all the neighboring frames to align with $\dot{\mathbf{I}}_t^{HR}$ at the time step t, denoted by $\hat{\mathbf{I}}_k^{HR}(k \neq t)$. Alongside the frame pixels, we also extract and warp the context information from a pre-trained ResNet18 [44] model. Finally, all the generated motions $\mathbf{f}_k$, kernel $\mathbf{K}_k$, context $\hat{\mathbf{C}}_k$, warped frame $\hat{\mathbf{I}}_k^{HR}$ as well as up-sampled blurry frame $\dot{\mathbf{I}}_t^{HR}$ are fed into a frame enhancement network. Our frame enhancement network has a similar architecture to the single-image super-

resolution method, EDSR [52].

The frame enhancement network is deeper than the post-processing network for video frame interpolation. Since the input frames are heavily degraded by low resolution, noise, or blockiness, the frame enhancement network thus requires more complex architecture to restore high-quality results. In the frame enhancement network shown in Fig. 10(a), we first use one convolutional layer with a ReLU activation. Then, we adopt 10 residual blocks, where each residual block contains two convolutional layers, one ReLU layer, and a skip connection as shown in Fig. 10(b). Finally, the last convolutional layer generates the residuals between the input and output images. All the convolutional layers have 128 channels and a 3×3 kernel size. The final output frame of this network is denoted by $\tilde{\mathbf{I}}_t$. We name the entire video super-resolution network as MEMC-Net_SR as shown in Fig. 11.

The model difference between the MEMC-Net_SR and MEMC-Net* is twofold. First, the MEMC-Net_SR does not require the flow projection layer as we can directly estimate flow for the target frame. Second, since each pixel of the target frame has a valid flow vector, we discard the mask estimation module in MEMC-Net_SR. We use the same network architecture as MEMC-Net_SR for video denoising and deblocking. The extended model for denoising and deblocking are referred to as MEMC-Net_DN and MEMC-Net_DB, respectively.

For each of the three video enhancement tasks, we train our network on the corresponding training set from the Vimeo90K dataset [9], namely Vimeo90K-SR, Vimeo90K-DN, and Vimeo90K-DB. Each of the training sets consists of 91,701 7-frame sequences with an image resolution of 448×256 pixels. Note that the input images of the Vimeo90K-SR set are first downsampled to a resolution of 224×128 and then upsampled to 448×256 with the bicubic interpolation. In each training iteration, a batch contains one sequence with 7 consecutive frames. The learning rate is initialized to 0.0005 for the first 10 epochs and then is dropped to 0.0001 for the following 5 epochs. Similar to the video frame interpolation task, we initialized the parameters with the method of He et al. [46] and use the Adamax optimizer [47] to update the parameters.

7 EXPERIMENTAL RESULTS

In this section, we first analyze and discuss the contributions of each sub-module, processing speed, and model parameters. We then present the experimental results on video frame interpolation and the other three video frame enhancement tasks.

7.1 Analysis and Discussion

We first describe the evaluated datasets and conduct experiments to analyze the contribution of each component in the proposed model, especially on flow and kernel estimation.

7.1.1 Datasets

We evaluate the proposed frame interpolation approach on a wide variety of video datasets.

Middlebury. The Middlebury dataset [29] is widely used for evaluation of optical flow estimation, stereo image matching, and frame interpolation methods. There are 12 sequences in the OTHER set and 8 sequences in the EVALUATION set with a resolution of 640×480 pixels. We use the evaluation protocol to compute the *Interpolation Error* (IE) and *Normalized Interpolation Error* (NIE).

UCF101. The UCF101 dataset [30] contains a large variety of human actions. We use 379 triplets from the UCF101 test set, where the image resolution is 256×256 of pixels.

Vimeo90K. Xue et al. [9] develop a high-quality dataset with videos from Vimeo (https://vimeo.com). There are 3,782 triplets for evaluation with the image resolution of 448×256 pixels.

HD videos. In this work, we collect 7 HD (High Definition) videos from the Xiph website (https://media.xiph.org/video/derf/), and interpolate the first 50 even frames for each of the videos. We also evaluate on four short video clips from the Sintel dataset [53], where each image is of 1280×544 pixels.

7.1.2 Ablation Studies

Flow-based analysis. We first construct a baseline model by using the optical flow estimation network and bilinear interpolation for warping images. Similar to the ToFlow [9] method, the baseline model does not contain the kernel estimation, mask estimation, and post-processing networks. Table 2 shows the performance of this baseline model is similar to that by the ToFlow method. We then include the mask estimation and post-processing (abbreviated by post-proc. in Table 2) networks, where both modules clearly contribute to the performance on all the test datasets. By replacing the fixed bilinear kernel module with the proposed spatially-adaptive kernel estimation network, a significant performance gain can be achieved. Our final model with all the components achieves the state-of-the-art performance on all three benchmark datasets. The FlowNetS contains 38.6M parameters, which take 57.4% of the parameters in our MEMC-Net model. We conduct an experiment to use a fixed flow estimation network. The performance of this variant drops a lot on all the datasets, as shown in the last two rows of Table 2. Without fine-tuning the flow estimation network, the projected flow is a simple approximation of the flow from the intermediate frame to the reference frame, which is not able to synthesize high-quality intermediate frame.

Kernel-based analysis. We conduct experiments to analyze the contribution of the learned interpolation kernels in the proposed method. We train a baseline model by removing the optical flow estimation network and only learn 4×4 spatially-adaptive kernels for interpolation. This baseline model is similar to the SepConv method [12] but with a much smaller kernel. In Table 3, it is interesting to see that this baseline model already outperforms the SepConv method on the UCF101 dataset. It is sufficient to use a 4×4 interpolation kernel for all videos as the image resolution is low and object motion is small. However, it does not perform well on the Vimeo90K and Middlebury datasets, which contain much larger motion. By introducing the flow

TABLE 2. **Analysis on flow-based methods.** The red numbers indicate the best performance.

Methods	Sub-networks				UCF101 [30]		Vimeo90K [9]		Middlebury [29]	#Param.
	flow	kernel (size)	mask	post-proc.	PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	
DVF [36]	Enc-Dec	bilinear (2)	√	×	34.12	0.9631	31.54	0.9462	7.75	1,604,547
ToFlow+Mask [9]	SPyNet	bilinear (2)	√	√	34.58	0.9667	33.73	0.9682	2.51	1,074,635
MEMC-Net	FlowNetS	bilinear (2)	×	×	34.65	0.9664	32.73	0.9606	2.81	38,676,496
	FlowNetS	bilinear (2)	×	√	34.70	0.9667	33.25	0.9646	2.50	38,922,323
	FlowNetS	bilinear (2)	√	×	34.69	0.9667	32.94	0.9626	2.72	52,842,818
	FlowNetS	bilinear (2)	√	√	34.76	0.9671	33.40	0.9661	2.47	53,088,645
	FlowNetS	learned (4)	×	√	34.77	0.9669	33.29	0.9663	2.42	53,092,995
	FlowNetS	learned (4)	√	×	34.88	0.9669	33.51	0.9670	2.37	67,013,490
	FlowNetS (fixed)	learned (4)	√	√	33.15	0.9632	32.05	0.9580	3.26	28,583,973
	FlowNetS	learned (4)	√	√	**34.95**	**0.9679**	**34.02**	**0.9704**	**2.24**	67,260,469

TABLE 3. **Analysis on kernel-based methods.**

Methods	Sub-networks				UCF101 [30]		Vimeo90K [9]		Middlebury [29]	#Param.
	flow	kernel (size)	mask	post-proc.	PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	
SepConv-L_f [12]	×	learned (51)	×	×	34.69	0.9655	33.45	0.9674	2.44	21,675,452
SepConv-L_1 [12]	×	learned (51)	×	×	34.78	0.9669	33.79	0.9702	2.27	21,675,452
MEMC-Net	×	learned (4)	×	×	34.89	**0.9682**	32.73	0.9581	2.74	14,710,415
	×	learned (4)	×	√	34.97	**0.9682**	33.31	0.9633	2.57	14,720,783
	FlowNetS	learned (4)	√	√	**34.95**	0.9679	**34.02**	**0.9704**	**2.24**	67,260,469

TABLE 4. **Evaluation on models with fewer model parameters.** M.B. stands for Middlebury.

Methods	UCF101 [30]		Vimeo90K [9]		M.B. [29]	#Param.
	PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	
MEMC-Net_s	34.83	0.9676	33.97	0.9721	2.44	**7.2M**
MEMC-Net	34.95	0.9679	34.02	0.9704	2.24	67.2M
MEMC-Net*	**35.01**	**0.9683**	**34.40**	**0.9742**	**2.10**	70.3M

TABLE 5. **Runtime of frame interpolation methods (seconds).**

Methods	640 × 480p	1280 × 720p	1920 × 1080p
AdaConv [28]	2.80	—	—
ToFlow [9]	0.43	1.01	1.90
SepConv [12]	0.20	0.50	0.90
MEMC-Net_s	0.13	0.33	0.67
MEMC-Net	**0.06**	**0.20**	**0.41**
MEMC-Net*	0.12	0.36	0.64

TABLE 6. **Runtime of the proposed models (seconds).** We evaluate these models on 640 × 480 videos.

Networks	flow	kernel	mask	context	post-proc.	Total
MEMC-Net_s	0.103	0.005	—	—	0.020	0.13
MEMC-Net	0.024	0.008	0.008	—	0.020	0.06
MEMC-Net*	0.024	0.008	0.008	0.001	0.080	0.12

estimation network, the proposed method performs better than the evaluated models on the Vimeo90K and Middlebury datasets. The results demonstrate the importance of integrating optical flow and learned interpolation kernels to deal with large motions in frame interpolation.

Model parameters. Since modern mobile devices typically have limited memory size, we present a smaller model with fewer parameters but maintaining the same MEMC framework. We first replace the FlowNetS [21] with the SPyNet [11] model, which reduces about 97% parameters in the flow estimation network. We then simplify the kernel prediction network by removing one convolutional layer before each max-pooling and un-pooling layer and discard the occlusion estimation network. This reduced model, denoted by MEMC-Net_s, has only 7,204,367 trainable parameters, which is 89.3% smaller than our full network. We compare the performance of the full and small models in Table 4. As the SPyNet performs better than the FlowNetS [21] on small motion [11], the performance of MEMC-Net_s is slightly better than our full model on the Vimeo90K dataset. However, MEMC-Net_s does not perform as well on the Middlebury dataset as it contains large displacement.

Execution speed. We evaluate the runtime of the proposed algorithm on an NVIDIA Titan X (Pascal) GPU. We compare the execution time of the proposed method and the state-of-the-art algorithms on 640 × 480p, 1280 × 720p and 1920 × 1080p videos in Table 5. Our MEMC-Net model can process 1920 × 1080p videos with the runtime of 0.41 second per frame. Moreover, when using four GPU cards to process a 1920 × 1080p videos in parallel by splitting input frames into 270 × 240 non-overlapped patches, our method is able to process 30 frames per second. We note that the small model MEMC-Net_s does not necessarily have better runtime performance as the SPyNet applies several convolutional layers on the input resolution (which results in larger feature maps and a higher computational cost). On the other hand, the operations of the FlowNetS model are mostly applied on the 1/4 or smaller resolution space. The SPyNet uses 97% fewer parameters but has 1.35 times more FLOPs than the FlowNetS. In Table 6, we show the runtime of each component in the proposed models. The small model can be used for memory-constrained devices while the full model is preferable for applications that require prompt response time. The proposed MEMC-Net* can be used for cases where the interpolation quality is of most importance.

Our MEMC-Net_s consists of only 7.2M parameters but performs better than a larger SepConv model which contains 21.7M parameters. Furthermore, the amount of parameters is not the only factor to be considered. Although the ToFlow+Mask model uses fewer parameters, it runs slower than the proposed and SepConv methods. We present the evaluation results of these algorithms in Table 7.

TABLE 7. **Quantitative evaluation on UCF101, Vimeo90K, and Middlebury datasets.** The abbreviations *oth.* and *eval.* represent the OTHER and EVALUATION sets in the Middlebury dataset. The runtime is evaluated on the 640×480 sequences.

Methods	#Parameters (million)	Runtime (seconds)	UCF101 [30]		Vimeo90K [9]		Middlebury [29]	
			PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	IE (*eval.*)
ToFlow+Mask [9]	**1.07**	0.43	34.58	0.9667	33.73	0.9682	2.51	—
SepConv-L_1 [12]	21.6	0.20	34.78	0.9669	33.79	0.9702	2.27	5.61
MEMC-Net_s	<u>7.20</u>	0.13	34.83	0.9676	33.97	<u>0.9721</u>	2.44	—
MEMC-Net	67.2	**0.06**	<u>34.95</u>	<u>0.9679</u>	<u>34.02</u>	0.9704	<u>2.24</u>	<u>5.35</u>
MEMC-Net*	70.3	<u>0.12</u>	**35.01**	**0.9683**	**34.40**	**0.9742**	**2.10**	**5.00**

TABLE 8. **Quantitative comparisons with the sequential model.** M.B. stands for Middlebury.

Methods	UCF101 [30]		Vimeo90K [9]		M.B. [29]	#Param.
	PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	
Sequential	34.34	0.9652	32.94	0.9639	2.47	67.2M
MEMC-Net	**34.95**	**0.9679**	**34.02**	**0.9704**	**2.24**	67.2M

7.2 Video Frame Interpolation

We first provide the comparison with the sequential model and then present quantitative and qualitative evaluations with the state-of-the-art approaches.

7.2.1 Comparisons with the Sequential Model

We train the sequential model and present the quantitative results in Table 8. Compared to the proposed approach, the performance of the sequential model is 0.61dB and 1.08dB lower on the UCF101 and Vimeo90K datasets, respectively. We attribute the performance difference to the flow warping errors on the motion boundary (which has occlusion and dis-occlusion). The proposed model avoid estimating kernels from warped images and leads to better performance.

7.2.2 Comparisons with the State-of-the-arts

We evaluate the proposed MEMC-Net and MEMC-Net* against the kernel-based method (SepConv [12]), flow-based algorithms (DVF [36], ToFlow [9], and CtxSyn [27]), and a direct interpolation approach (MIND [7]). The ToFlow method [9] generates two results with and without learning occlusion masks. Two pre-trained models of the SepConv approach [12] are available: the SepConv-L_1 model is optimized with a L_1 loss function while the SepConv-L_f model uses both the L_1 loss and the perceptual loss [56] for generating more realistic results. As no pre-trained model of the MIND method [7] is available, we train their network on the Vimeo90K training set for evaluations. In addition to the above learning-based frame interpolation methods, we also use existing optical flow algorithms (SPyNet [11] and EpicFlow [10]) to directly interpolate frames.

We show the interpolation results of the Middlebury EVALUATION set in Table 9. These results are also publicly available on the Middlebury benchmark website (http://vision.middlebury.edu/flow/eval/results/results-i1.php). For the sequences with smaller motions (i.e., the maximum flow magnitude is less than 10 pixels) or fine textures, such as the *Mequon*, *Teddy* and *Schefflera*, the CtxSyn method [27] obtains the best results in terms of both IE and NIE metrics. In contrast, our MEMC-Net and MEMC-Net* perform well on the videos with complicated motion,

e.g., the *Backyard* sequence with dancing feet and the *Basketball* video with moving arms and fingers. Notably, the SuperSlomo method [38] generates the best results for the synthetic *Urban* sequence. On average, the proposed models perform favorably against the state-of-the-art approaches on the Middlebury dataset. In Table 10, we present the average IE and NIE values with standard variances on the Middlebury dataset. Also in Fig. 13, we present error bars on the IE and NIE metrics to show the statistical comparison between different methods. The MEMC-Net* obtains lower interpolation error at smaller variance on different scenarios.

Table 11 shows that the proposed methods perform favorably against the state-of-the-art approaches on the UCF101 [30], Vimeo90K [9], and Middlebury [29] datasets. The numbers in **red** depict the best performance, while the numbers in <u>blue</u> depict the second-best performance. The diverse scenarios in these video datasets demonstrate that our model generalizes well to different types of motion. On the other hand, the MIND model [7] trained on the same Vimeo90k training set does not perform well on the UCF101 and Middlebury datasets.

In Table 12, we present the evaluation results on the HD videos, which typically contain much larger motion. Our approach consistently performs well against the state-of-the-art methods on different resolutions. The performance gap between the proposed MEMC-Net and SepConv [12] becomes larger especially on 1080p videos, which demonstrates that it is not feasible to handle large motion with the fixed kernel size (e.g., 51×51). Our MEMC-Net* with context information and residual blocks performs favorably against the existing methods with significant improvement up to 0.9dB (e.g., *Alley2* and *ParkScene*).

7.2.3 Qualitative Results

We present sample interpolation results from the evaluated datasets in Fig. 12, 14, 15, and 16. On the first row of Fig. 12, the EpicFlow [10] and SPyNet [11] methods do not reconstruct the straight lamppost due to inaccurate optical flow estimation. Both the SepConv-L_1 and SepConv-L_f [12] models cannot interpolate the lamppost well as the motion is larger than the size of the interpolation kernels. In contrast, our method reconstructs the lamppost well. On the second row of Fig. 12, the proposed method interpolates the falling ball with a clear shape with fewer artifacts on the leg of the girl.

In Fig. 15, the ToFlow [9] and SepConv [12] methods generate ghost effect around the hand. Due to large and non-rigid motion, flow-based methods are less effective in estimating accurate optical flow for interpolation, while kernel-based approaches are not able to infer motion beyond

TABLE 9. **Quantitative results on the Middlebury** EVALUATION **set.** The **red** numbers indicate that corresponding method takes the 1st place among all the evaluated algorithms.

Methods	Mequon		Schefflera		Urban		Teddy		Backyard		Basketball		Dumptruck		Evergreen		Average	
	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE	IE	NIE
EpicFlow [10]	3.17	0.62	3.79	0.70	4.28	1.06	6.37	1.09	11.2	1.18	6.23	1.10	8.11	1.00	8.76	1.04	6.49	0.97
MDP-Flow2 [54]	2.89	0.59	3.47	0.62	3.66	1.24	5.20	0.94	10.2	0.98	6.13	1.09	7.36	0.70	7.75	0.78	5.83	0.87
DeepFlow2 [55]	2.98	0.62	3.88	0.74	3.62	0.86	5.39	0.99	11.0	1.04	5.91	1.02	7.14	0.63	7.80	0.96	5.97	0.86
SepConv-L_1 [12]	2.52	0.54	3.56	0.67	4.17	1.07	5.41	1.03	10.2	0.99	5.47	0.96	6.88	0.68	6.63	0.70	5.61	0.83
SuperSlomo [38]	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	5.31	0.78
CtxSyn [27]	**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	5.28	0.82
MEMC-Net	2.83	0.64	3.84	0.73	4.16	0.84	5.75	0.99	8.57	0.93	4.99	0.96	**5.86**	**0.60**	6.83	0.69	5.35	0.80
MEMC-Net*	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**	**5.00**	**0.74**
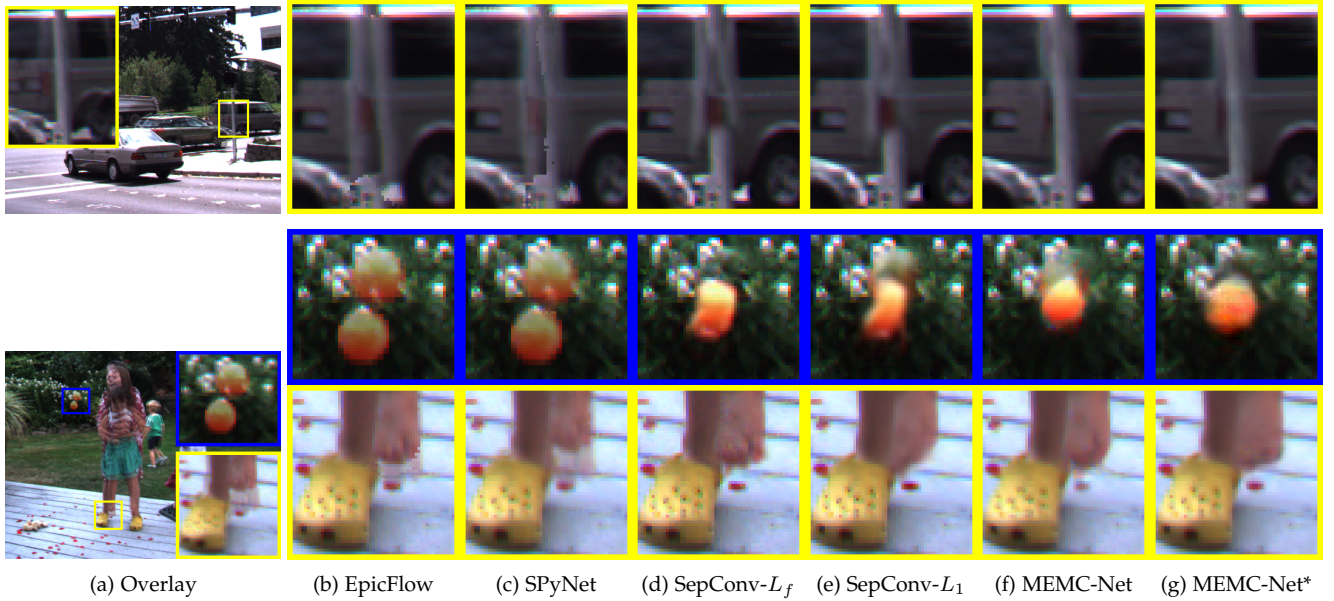
Fig. 12. **Visual comparisons on Middlebury [29].** The sequences are from the EVALUATION set.

(a) Overlay   (b) EpicFlow   (c) SPyNet   (d) SepConv-L_f (e) SepConv-L_1 (f) MEMC-Net   (g) MEMC-Net*

TABLE 10. **Average IE and NIE values with standard variances on Middlebury benchmark.**

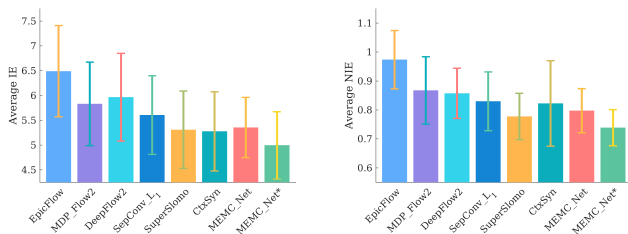| Method	IE	NIE
EpicFlow [10]	6.48 ± 2.75	0.97 ± 0.20
MDP_Flow2 [54]	5.83 ± 2.52	0.86 ± 0.23
DeepFlow2 [55]	5.96 ± 2.65	0.85 ± 0.17
SepConv_L1 [12]	5.60 ± 2.37	0.83 ± 0.20
SuperSlomo [38]	5.31 ± 2.34	0.77 ± 0.16
CtxSyn [27]	5.27 ± 2.39	0.82 ± 0.29
MEMC-Net	5.35 ± 1.81	0.79 ± 0.15
MEMC-Net*	**4.99 ± 2.02**	**0.73 ± 0.12**

Fig. 13. **Error bars on the Middlebury sequences.**

the size of local kernels. In contrast, our model reconstructs the hand with fewer visual artifacts. As shown in Fig. 16, the SepConv [12] method is not able to interpolate the scene

well due to large motion. In contrast, the proposed method interpolates frames well with visually pleasing results when optical flow is not accurately estimated.

7.3 Video Frame Enhancement

We use the Vimeo90K dataset [9] to evaluate the proposed method on the video denoising, video super-resolution, and video deblocking tasks. There are 7,824 sequences in the Vimeo90k test set, and each contains 7 consecutive frames. The qualitative results for super-resolution, denoising and deblocking tasks are presented in Table 13, Table 14 and Table 15, respectively.

Super-Resolution. We evaluate the proposed method on the widely used video super-resolution dataset developed by Liu et al. [57], which is denoted by BayesSR in Table 13. The low-resolution image distortion for both the Vimeo90K and BayesSR datasets are generated by down-sampling the original high-resolution frames at the scaling ratio of 4 (use the MATLAB function *imresize* with the *bicubic* mode). And the evaluated algorithms are to up-sample the middle frame of a sequence in Vimeo90K dataset or each frame of a video in BayesSR dataset by a factor of 4. The DeepSR [51] and ToFlow [9] methods are CNN-based approaches for video super-resolution. In addition, we also compare with

TABLE 11. **Quantitative evaluation on UCF101, Vimeo90K, and Middlebury datasets.** The abbreviations *oth.* and *eval.* represent the OTHER and EVALUATION sets in the Middlebury dataset. The numbers in red depict the best performance, while the numbers in blue depict the second-best performance.

Methods	UCF101 [30]		Vimeo90K [9]		Middlebury [29]	
	PSNR	SSIM	PSNR	SSIM	IE (*oth.*)	IE (*eval.*)
SPyNet [11]	33.67	0.9633	31.95	0.9601	2.49	—
EpicFlow [10]	33.71	0.9635	32.02	0.9622	2.47	6.48
MIND [7]	33.93	0.9661	33.50	0.9429	3.35	—
DVF [36]	34.12	0.9631	31.54	0.9462	7.75	—
ToFlow [9]	34.54	0.9666	33.53	0.9668	—	—
ToFlow+Mask [9]	34.58	0.9667	33.73	0.9682	2.51	—
SepConv-L_f [12]	34.69	0.9655	33.45	0.9674	2.44	—
SepConv-L_1 [12]	34.78	0.9669	33.79	0.9702	2.27	5.61
MEMC-Net_s	34.83	0.9676	33.97	0.9721	2.43	—
MEMC-Net	34.95	0.9679	34.02	0.9704	2.24	5.35
MEMC-Net*	**35.01**	**0.9683**	**34.40**	**0.9742**	**2.10**	**5.00**

TABLE 12. **Quantitative evaluation on HD videos.**

Video	Resolution	ToFlow+Mask [9]		SepConv-L_f [12]		SepConv-L_1 [12]		MEMC-Net		MEMC-Net*	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Alley2	544p	26.30	0.7997	28.26	0.8462	28.52	0.8646	29.57	0.8845	**29.60**	**0.8920**
Market5	544p	18.21	0.7324	20.59	0.7878	20.57	0.8012	21.16	0.8074	**21.64**	**0.8105**
Temple1	544p	25.20	0.9174	26.42	0.9295	26.69	0.9370	**27.25**	0.9354	27.18	**0.9390**
Temple2	544p	19.90	0.8246	21.74	0.8471	21.93	0.8533	22.72	0.8628	**22.94**	0.8532
Parkrun	720p	27.77	0.8841	28.69	0.9083	29.03	**0.9158**	29.07	0.9125	**29.15**	0.9145
Shields	720p	34.10	0.8884	34.55	0.9093	34.91	0.9188	35.21	0.9206	**35.49**	**0.9251**
Stockholm	720p	33.53	0.8534	33.99	0.8669	34.27	0.8826	34.64	0.8894	**34.89**	**0.8931**
Kimono	1080p	33.34	0.9107	34.07	0.9168	34.31	0.9287	34.93	0.9341	**34.99**	**0.9363**
ParkScene	1080p	33.49	0.9233	35.27	0.9374	35.51	0.9451	36.20	0.9491	**36.64**	**0.9521**
Sunflower	1080p	33.75	0.9476	34.88	0.9539	35.02	0.9605	35.42	0.9616	**35.59**	**0.9638**
Bluesky	1080p	37.53	0.9673	38.32	0.9730	38.83	0.9775	39.28	0.9791	**39.55**	**0.9801**
Average		29.37	0.8772	30.61	0.8978	30.87	0.9077	31.40	0.9124	**31.60**	**0.9145**

(a) Overlay (b) MIND [7] (c) ToFlow [9] (d) EpicFlow [10] (e) SPyNet [11]

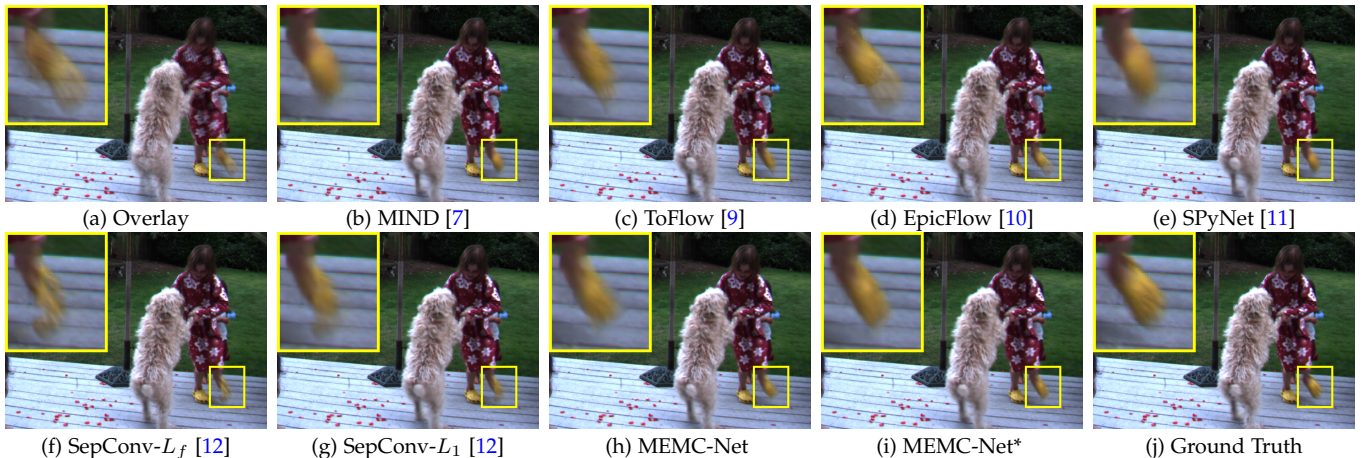(f) SepConv-L_f [12] (g) SepConv-L_1 [12] (h) MEMC-Net (i) MEMC-Net* (j) Ground Truth

Fig. 14. **Visual comparisons on Middlebury [29].** The sequences are from the OTHER set.

the BayesSR [57] method. Since the single-image super-resolution (SISR) is also a well-studied task, we include the state-of-the-art SISR method, EDSR [52], for evaluations.

We present the quantitative results on video super-resolution in Table 13. Our method performs favorably against the state-of-the-art approaches on both benchmark datasets. Compared to the state-of-the-art SISR method [52], MEMC-Net_SR has fewer residual blocks and a smaller number of filters but obtains higher PSNRs on both the Vimeo90K and BayesSR datasets. Compared to existing video super-resolution approaches [9], [51], [57], our method

is more favorable, especially on the BayesSR dataset. In Fig. 17, we present the video super-resolution results. In the first row, the EDSR [52] does not restore the correct shape of the number "31" on the calendar. The results by the ToFlow [9] and BayesSR [57] methods contain artifacts and blurry pixels. In contrast, the proposed MEMC-Net_SR model is able to restore sharper video frames.

Denoising. We evaluate our method with the ToFlow [9] and V-BM4D [58] algorithms. In addition, we train a single frame denoising model as the baseline. The model architecture is the same as the EDSR network except that the
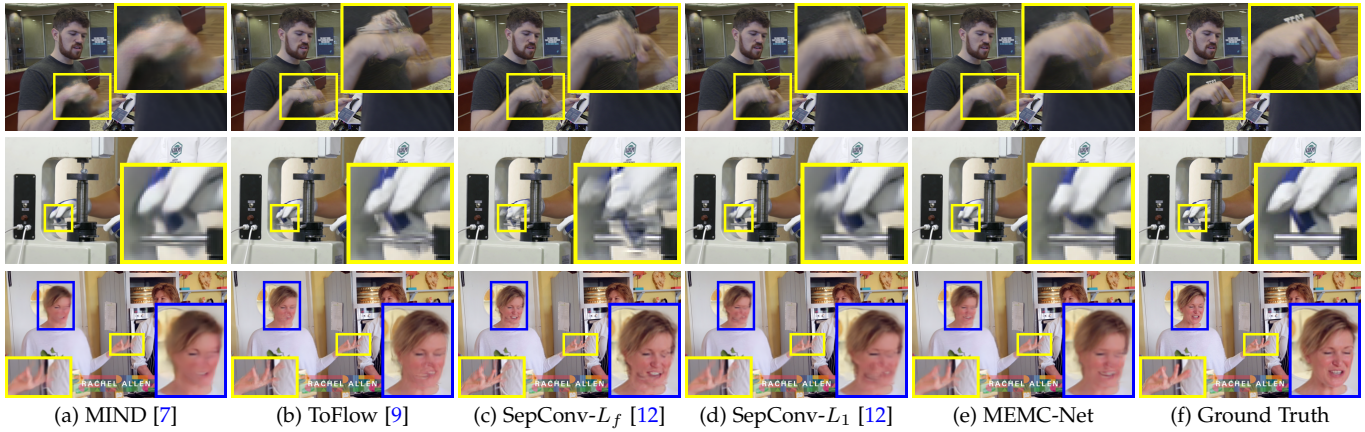
(a) MIND [7] (b) ToFlow [9] (c) SepConv-L_f [12] (d) SepConv-L_1 [12] (e) MEMC-Net (f) Ground Truth

Fig. 15. **Visual comparisons on the Vimeo90K dataset [9].**

(a) ToFlow [9] (b) SepConv-L_f [12] (c) SepConv-L_1 [12] (d) MEMC-Net (e) MEMC-Net* (f) Ground Truth

Fig. 16. **Visual comparisons on HD videos.**

(a) Bicubic (b) EDSR [52] (c) ToFlow [9] (d) BayesSR [57] (e) MEMC-Net_SR (f) Ground Truth
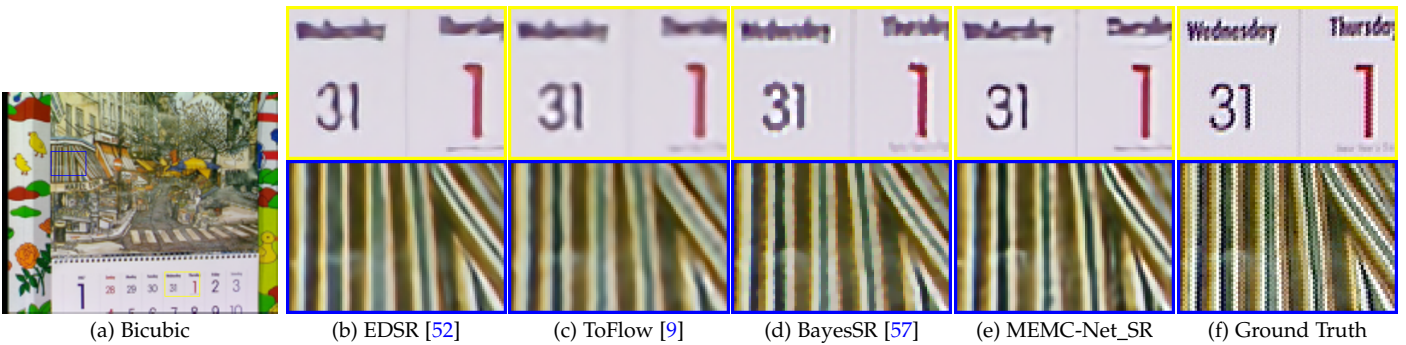
Fig. 17. **Visual comparisons of video super-resolution methods.**

(a) Noisy (b) EDSR_DN [52] (c) ToFlow [9] (d) V-BM4D [58] (e) MEMC-Net_DN (f) Ground Truth
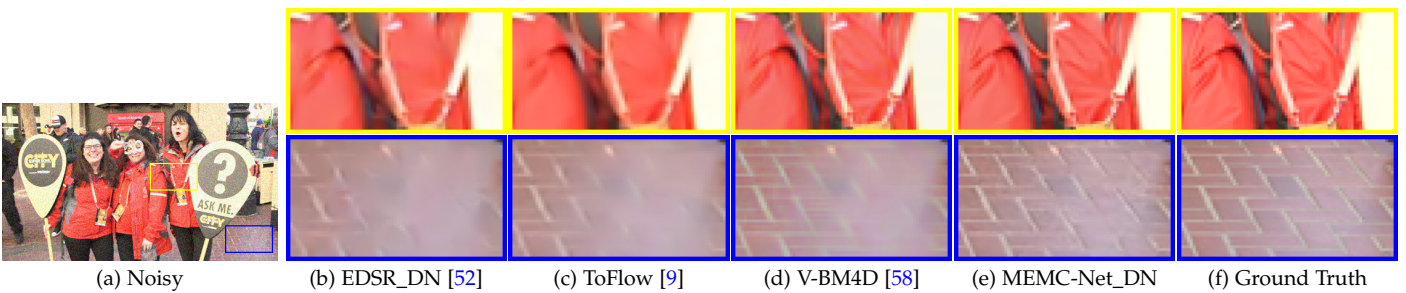
Fig. 18. **Visual comparisons of video denoising methods.**

(a) Blocky (b) EDSR_DB [52] (c) ToFlow [9] (d) V-BM4D [58] (e) MEMC-Net_DB (f) Ground Truth
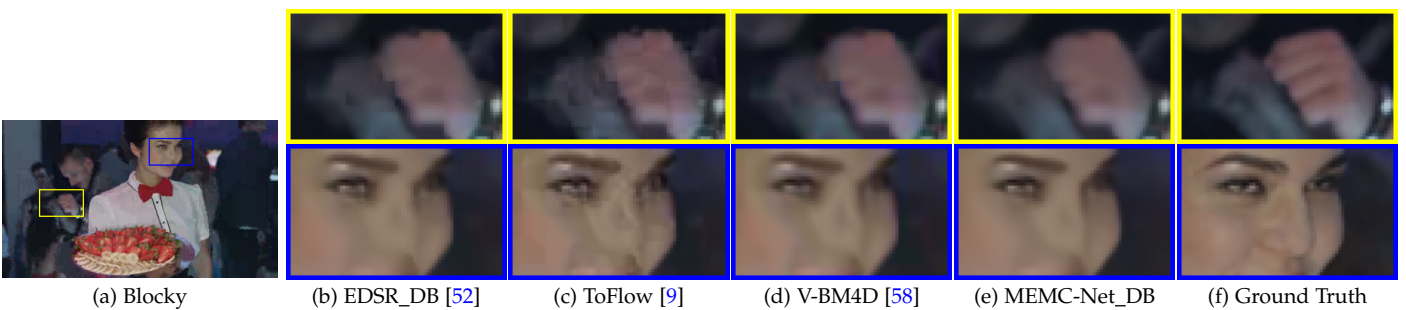
Fig. 19. **Visual comparisons of video deblocking methods.**

input images are with noise instead of low-resolution, and referred to as EDSR_DN. We evaluate on the Vimeo90k test set as well as the dataset developed by Maggioni et al. [58]. For the denoising experiments, we add Gaussian noise with

TABLE 13. **Quantitative evaluation for video super-resolution.**

Frame #Num.	Methods	Vimeo90K [9]		BayesSR [57]	
		PSNR	SSIM	PSNR	SSIM
1	Bicubic	29.79	0.9036	22.17	0.7391
	EDSR [52]	33.08	0.9411	23.93	0.8113
7	DeepSR [51]	25.55	0.8498	21.85	0.7535
	BayesSR [57]	24.64	0.8205	21.95	0.7369
	ToFlow [9]	33.08	0.9417	23.54	0.8070
	MEMC-Net_SR	**33.47**	**0.9470**	**24.37**	**0.8380**

TABLE 14. **Quantitative evaluation for video denoising.**

Frame #Num.	Methods	Vimeo90K [9]		V-BM4D [58]	
		PSNR	SSIM	PSNR	SSIM
1	Noisy	22.63	0.5007	22.28	0.4715
	EDSR_DN [52]	35.11	0.9513	32.02	0.8828
7	ToFlow [9]	32.66	0.9198	30.19	0.8699
	V-BM4D [58]	34.39	0.9217	32.27	0.8913
	MEMC-Net_DN	**36.35**	**0.9642**	**34.22**	**0.9310**

TABLE 15. **Quantitative evaluation for video deblocking.**

Frame #Num.	Methods	Vimeo90K [9]		V-BM4D [58]	
		PSNR	SSIM	PSNR	SSIM
1	Blocky	31.99	0.9179	29.38	0.8302
	EDSR_DB [52]	32.87	0.9319	29.66	0.8362
7	ToFlow [9]	32.57	0.9292	29.59	0.8390
	V-BM4D [58]	32.74	0.9293	29.94	0.8435
	MEMC-Net_DB	**33.37**	**0.9388**	**30.14**	**0.8498**

$\sigma = 20$ to synthesize noisy input frames.

The quantitative results for video denoising are presented in Table 14. Our method performs well on both datasets. The PSNR gains of MEMC-Net_DN over the second best method are 1.24dB and 1.95dB on the Vimeo90K and V-BM4D datasets, respectively. In Fig. 18, the fine textures on the clothes and street are not well restored by the EDSR_DN, ToFlow, and V-BM4D methods. In contrast, our MEMC-Net_DN preserves these textures well.

Deblocking. For the video deblocking task, we use the same videos as in the denoising task. The images encoded by the widely used H.264 [59] standard may generate blockiness due to the block-based approach. We use the FFmpeg software to encode the images in the Vimeo90K and V-BM4D datasets with *libx264* and the quality parameter *qp* of 37, and disable the in-loop deblocking of the codec. We compare the proposed algorithm with the EDSR_DB [52], ToFlow [9] and V-BM4D [58] methods.

The quantitative evaluation results for video deblocking are presented in Table 15. Overall, the proposed model performs favorably against all the evaluated algorithms. In Fig. 19, the blocky regions around the hand and eye are sufficiently reduced by both MEMC-Net_DB and V-BM4D [58] methods. The ToFlow [9] and EDSR_DB schemes, however, do not reduce the blocky pixels well.

8 CONCLUSIONS

In this work, we propose the motion estimation and motion compensation driven neural network for learning video frame interpolation and enhancement. Our model exploits the merits of the MEMC framework to handle large motion as well as the data-driven learning-based methods to extract effective features. Two network layers, namely the adaptive warping layer and flow projection layers, are proposed to tightly integrate all the sub-networks to make our model end-to-end trainable. The generalized motion compensated alignment of the proposed MEMC framework enables it to be extended to various video enhancement tasks such as video super-resolution, denoising, and deblocking. Quantitative and qualitative evaluations on the various benchmark datasets show that the proposed methods perform favorably against the state-of-the-art algorithms in video interpolation and enhancement.
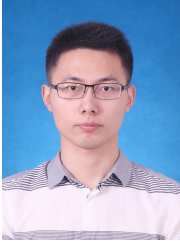
REFERENCES

[1] R. Castagno, P. Haavisto, and G. Ramponi, "A method for motion adaptive frame rate up-conversion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 436–446, 1996. 1

[2] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[3] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Modeling and optimization of high frame rate video transmission over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2713–2726, 2016. 1

[4] G. De Haan, P. W. Biezen, H. Huijgen, and O. A. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 5, pp. 368–379, 1993. 1, 2

[5] W. Bao, X. Zhang, L. Chen, L. Ding, and Z. Gao, "High-order model and dynamic filtering for frame rate up-conversion," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3813–3826, 2018. 1

[6] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1988–2001, 2015. 1

[7] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *European Conference on Computer Vision*, 2016. 1, 2, 3, 10, 12, 13

[8] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *International Conference on Learning Representations*, 2016. 1

[9] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *arXiv preprint arXiv:1711.09078*, 2017. 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14

[10] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 10, 11, 12

[11] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 9, 10, 12

[12] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in *IEEE International Conference on Computer Vision*, 2017. 2, 3, 4, 8, 9, 10, 11, 12, 13

[13] J. Konrad and E. Dubois, "Bayesian estimation of motion vector fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 9, pp. 910–927, 1992. 1

[14] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 693–699, 1994. 1, 2

[15] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000. 1

[16] X. Gao, C. Duanmu, and C. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 501–504, 2000. 1

[17] C. Wang, L. Zhang, Y. He, and Y.-P. Tan, "Frame rate up-conversion using trilateral filtering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 6, pp. 886–893, 2010. 1, 2

[18] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European Conference on Computer Vision*, 2004. 1

[19] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2443–2450. 1

[20] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1289–1297. 1

[21] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE International Conference on Computer Vision*, 2015. 1, 3, 6, 9

[22] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 4, pp. 407–416, 2007. 1

[23] M. Biswas and V. Namboodiri, "On handling of occlusion for frame rate up-conversion using video in-painting," in *IEEE International Conference on Image Processing*. IEEE, 2010, pp. 785–788. 1, 2

[24] W. H. Lee, K. Choi, and J. B. Ra, "Frame rate up conversion based on variational image fusion," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 399–412, 2014. 1, 2

[25] U. S. Kim and M. H. Sunwoo, "New frame rate up-conversion algorithms with low computational complexity," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 3, pp. 384–393, 2014. 1, 2

[26] J. Zhai, K. Yu, J. Li, and S. Li, "A low complexity motion compensated frame interpolation method," in *IEEE International Symposium on Circuits and System*. IEEE, 2005, pp. 4927–4930. 1

[27] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 6, 10, 11

[28] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3, 4, 9

[29] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011. 2, 5, 8, 9, 10, 11, 12

[30] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," in *CRCV-TR-12-01*, 2012. 2, 8, 9, 10, 12

[31] K. M. Nam, J.-S. Kim, R.-H. Park, and Y. S. Shim, "A fast hierarchical motion vector estimation algorithm using mean pyramid," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 4, pp. 344–351, 1995. 2

[32] H. R. Kaviani and S. Shirani, "Frame rate upconversion using optical flow and patch-based reconstruction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1581–1594, 2016. 2

[33] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, "Phase-based frame interpolation for video," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 3

[34] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[35] W.-S. Lai, J.-B. Huang, and M.-H. Yang, "Semi-supervised learning for optical flow with generative adversarial networks," in *Neural Information Processing Systems*, 2017. 3

[36] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *IEEE International Conference on Computer Vision*, 2017. 3, 4, 9, 10, 12

[37] J. van Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero, "Frame interpolation with multi-scale deep loss functions and generative adversarial networks," *arXiv preprint arXiv:1711.06045*, 2017. 3

[38] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3, 10, 11

[39] Y. Zhang, D. Zhao, X. Ji, R. Wang, and X. Chen, "A spatio-temporal autoregressive frame rate up conversion scheme," in *IEEE International Conference on Image Processing*, 2007. 3

[40] Y. Zhang, D. Zhao, X. Ji, R. Wang, and W. Gao, "A spatio-temporal auto regressive model for frame rate upconversion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 9, pp. 1289–1301, 2009. 3

[41] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2010. 4

[42] D. Wang, A. Vincent, P. Blanchfield, and R. Klepko, "Motion-compensated frame rate up-conversionpart ii: New algorithms for frame interpolation," *IEEE Transactions on Broadcasting*, vol. 56, no. 2, pp. 142–149, 2010. 4

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015. 6

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6, 7

[45] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *IEEE International Conference on Image Processing*, 1994. 7

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision*, 2015. 7, 8

[47] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015. 7, 8

[48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, 2013. 7

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010. 7

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015. 7

[51] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, "Video super-resolution via deep draft-ensemble learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 7, 11, 12, 14

[52] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 8, 12, 13, 14

[53] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision*, 2012. 8

[54] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1744–1757, 2012. 11

[55] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *IEEE International Conference on Computer Vision*, 2013. 11

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 10

[57] C. Liu and D. Sun, "A bayesian approach to adaptive video super resolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 11, 12, 13, 14

[58] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-d

nonlocal spatiotemporal transforms," *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 3952–3966, 2012. 12, 13, 14

[59] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003. 14

Ming-Hsuan Yang is a professor in Electrical Engineering and Computer Science at University of California, Merced. He received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 2000. Yang serves as a program co-chair of IEEE International Conference on Computer Vision (ICCV) in 2019, program co-chair of Asian Conference on Computer Vision (ACCV) in 2014, and general co-chair of ACCV 2016. Yang served as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence from 2007 to 2011, and is an associate editor of the International Journal of Computer Vision, Image and Vision Computing and Journal of Artificial Intelligence Research. He received the NSF CAREER award in 2012, the Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He is a Fellow of the IEEE and a Senior Member of the ACM.

Wenbo Bao is a Ph.D. candidate of Electrical Engineering with the Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China. He received the B.S. degree in Electronic Information Engineering from Huazhong University of Science and Technology, Hubei, China, in 2014. His research interests include computer vision, machine learning, and video processing.

Wei-Sheng Lai is a Ph.D. candidate of Electrical Engineering and Computer Science at the University of California, Merced, CA, USA. He received the B.S. and M.S. degree in Electrical Engineering from the National Taiwan University, Taipei, Taiwan, in 2012 and 2014, respectively. His research interests include computer vision, computational photography, and deep learning.

Xiaoyun Zhang is an associate professor in Electrical Engineering at Shanghai Jiao Tong University. She received the B.S. and M.S. degrees in applied mathematics from Xian Jiaotong University in 1998 and 2001, respectively, and the Ph.D. degree in pattern recognition from Shanghai Jiao Tong University, China, in 2004. Her Ph.D. thesis has been nominated as National 100 Best Ph.D. Theses of China. Her research interests include computer vision and pattern recognition, image and video processing, digital TV system.

Zhiyong Gao is a professor in Electrical Engineering with Shanghai Jiao Tong University. He received the B.S. and M.S. degrees in electrical engineering from the Changsha Institute of Technology, Changsha, China, in 1981 and 1984, respectively, and the Ph.D. degree from Tsinghua University, Beijing, China, in 1989. From 1994 to 2010, he took several senior technical positions in England, including a Principal Engineer with Snell and Wilcox, Petersfield, U.K., a Video Architect with 3DLabs, Egham, U.K., a Consultant Engineer with Sony European Semiconductor Design Center, Basingstoke, U.K., and a Digital Video Architect with Imagination Technologies, Kings Langley, U.K. His research interests include video processing, video coding, digital TV, and broadcasting.