A NOVEL NEURAL NETWORK BASED MOTION ESTIMATION AND COMPENSATION BASED VIDEO ENHANCEMENT

B. Shravan Kumar[1], Dr. V. Usha Shree[2]
[1]Research Scholar JNTUH Hyderabad, India
[2]Professor and Principal, JBREC, R.R Dist, India

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.

INTRODUCTION

Video frame interpolation aims to synthesize nonexistent 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

Motion compensation (MC) is very useful in video filtering to remove noise and enhance signal. It is useful since it allows the filter or coder to process through the video on a path of near-maximum correlation based on following motion trajectories across the frames making up the image sequence or video. Motion compensation is also employed in all distribution-quality video coding formats, since it is able to achieve the smallest prediction error, which is then easier to code. Motion can be characterized in terms of either a velocity vector ***v*** or displacement vector ***d*** and is used to warp a *reference frame* onto a *target frame*. Motion estimation is used to obtain these displacements, one for each pixel in the target frame.

Several methods of motion estimation are commonly used:

- Block matching
- Hierarchical block matching
- Pel-recursive motion estimation
- Direct optical flow methods
- Mesh-matching methods

Optical flow is the apparent displacement vector field ***d*** D *.d*1,*d*2/ we get from setting (i.e., forcing) equality in the so-called *constraint equation*

$$x(n_1,n_2,n) = x(n_1 - d_1, n_2 - d_2, n-1).$$

All five approaches start from this basic equation, which is really just an idealization. Departures from the ideal are caused by the covering and uncovering of objects in the viewed scene, lighting variation both in time and across the objects in the scene, movement toward or away from the camera, as well as rotation about an axis (i.e., 3-D motion). Often the constraint equation is only solved approximately in the least-squares sense. Also, the displacement is not expected to be an integer as assumed in (11.2–1), often necessitating some type of interpolation to be used.

Motion cannot be determined on a pixel-by-pixel basis since there are two components for motion per pixel, and hence twice the number of unknowns as equations. A common approach then is to assume the motion is constant over a small region called the *aperture*. If the aperture is too large, then we will miss detailed motion and only get an average measure of the movement of objects in our scene. If the aperture is too small, the motion estimate may be poor to very wrong.

If the motion of the uniform dark region is parallel to its edge, then this motion cannot be detected. Since this situation would typically only hold for small regions in natural images, the aperture effect leads us to choose a not-too-small aperture size. Thus, finding the right aperture size is an important problem that depends on the video content.

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 Lp-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 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,

(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 WORKS

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.

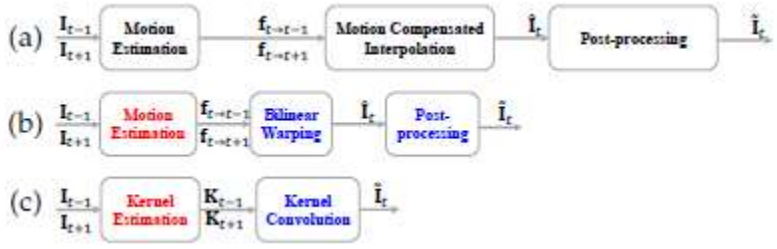

Fig. 2. **Frameworks of (a) the conventional MEMC-based approaches, (b) the flow-based and (c) the kernel-based models.** The **black**, **red**, and **blue** text boxes correspond to the conventional modules, network modules, and network layers respectively.

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 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 neighbour 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 pixelwise 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.

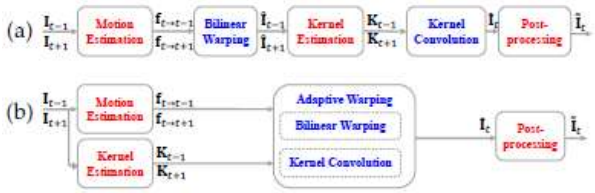

Fig. 3. **Frameworks of (a) the sequential MEMC-Net model and (b) our proposed MEMC-Net model.**

For an input frame of H x W pixels, the AdaConv model needs to estimate H x W x R x 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(R2) 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].

3 MOTION ESTIMATION AND MOTION COMPENSATION DRIVEN NEURAL NETWORK

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 postprocessing network is also added to the sequential model to reduce the possible pixel outliers. Fig. 3(a) illustrates this sequential model. However, according to our experiments, the warped frames (^It□1 and^It+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.

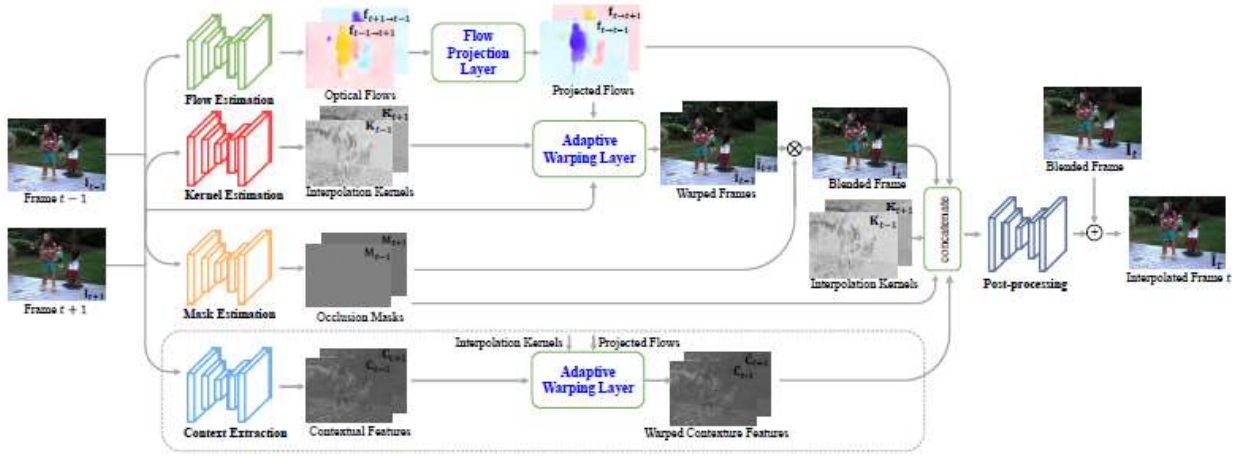

Fig. 4. **Network architecture of the proposed MEMC-Net and MEMC-Net.** The context extraction module and its generated contextual features and warped contextual features are for MEMC-Net.

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 postprocessing networks. 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 I(x) : Z^2 ! R^3 denote the RGB image where x ∈ [1;H] x [1;W], f (x) := (u(x); v(x)) represent the optical flow field and k^l(x) = [k^lr (x)]HxW (r ∈ [-R + 1;R]^2) indicate the interpolation kernel where R is the kernel size. The adaptive warping layer synthesizes an output image by:

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

where the weight kr = k^lr k^dr is determined by both the learned interpolation kernel k^lr and bilinear coefficient k^dr. 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] x [1;H], the kernel estimation network generates a 16-channel feature vector. We then map the feature vector into a 4 x 4 square matrix as the 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:

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

where θ(u) = u - uc denotes the fractional part of a float point number, and the subscript u, v of the 2-D vector 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., P11, P12, P21, and P22, 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 f is computed by (using the horizontal component u for example):

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

Where,

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

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^lr is:

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

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 ft!t□1(x) be the motion vector field of frame It to It□1. Similarly, ft□1!t+1(y) represents the motion vector field of frame It□1 to It+1. Note that we use y to index the 2-D coordinate at time step t □ 1, as distinguished to x at t. Our flow projection layer is designed to transform an estimated flow ft□1!t+1(y) to ft!t□1(x). Here we assume that the local motion between consecutive frames is linear and invert the flow between It□1 and It+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 x of time step t by S(x) := fy : round _ y +ft□1!t+1(y)=2_= x; 8 y 2 [1;H]_[1;W]g and denote the 4-directional nearest available flow vectors of a hole

by N(x) := fx0 : jS(x0)j > 0g. The forward pass of the proposed projection layer is defined by:

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

The backward pass computes the derivative with respect to the input optical flow ft□1!t+1(y):

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

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 fX is approximated by fX = (fA + fB + fC + fD)=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.

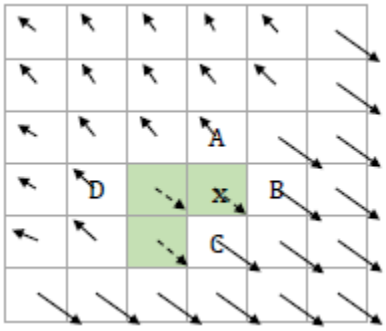

Fig. 6. **Outside-in strategy for filling the flow holes.** The green regions indicate a hole, where the flow vectors are approximated by the average of 4-directional available flow vectors from the non-hole regions.

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 It-1 and It+1, we first estimate the forward flow ft-1!t+1 and backward flow ft+1!t-1 by passing It-1 and It+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 ft-1!t+1 and backward flow ft+1!t-1 into ft!t□1 and ft!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 maxpooling 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 R2 coefficient maps, denoted by Kt□1 and Kt+1. We then reshape the coefficient maps to R_R convolutional kernels for each pixel, as shown in Fig. 5(b). Two pairs of intermediate flow and the kernel coefficients, fft!1, Kt1g and fft!t+1, Kt+1g are then fed into the proposed adaptive warping layer to warp the input frames by Eq. (1) and generate two warped frames^It□1 and^It+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 estimation network, but the last convolutional layer outputs a 2-channel feature map as the occlusion masks Mt-1 and Mt+1. The blended frame is generated by:

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

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 ^Ct-1 and ^Ct+1, are fed as inputs to the following postprocessing

.

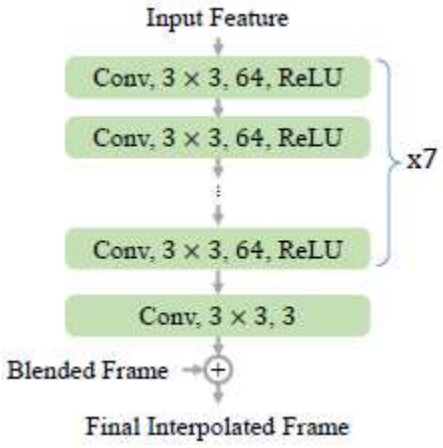

Fig. **Proposed post-processing network**

Post-processing. Since the blended image ^It 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 ^It, estimated flows ft!t+1 and ft!t-1, coefficient maps of the interpolation kernels Kt□1 and Kt+1, occlusion masksMt-1 andMt+1, and the warped context features ^Ct-1 and ^Ct+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 ^It 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 ~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 MEMCNet*.

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 ~It, ^It and the corresponding ground truth frame IGT t . We also regularize the sum of two masks to be 1.0. The combined loss function is given by:

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

Where, $\Phi(x) = \sqrt{x^2 + \epsilon^2}$

is the Charbonnier penalty function [45] with ∈ to be 1e-6. We empirically set _ and _ 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 β1 of 0:9 and β2 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.

6 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 favourably 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.

[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.

[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.

[4] G. De Haan, P. W. Biezen, H. Huijgen, and O. A. Ojo, "Truemotion 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.

[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.

[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.

[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.

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

[9] T. Xue, B. Chen, J.Wu, D.Wei, andW. T. Freeman, "Video enhancement with task-oriented flow," arXiv preprint arXiv:1711.09078, 2017.

[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.

[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.

[12] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive separable convolution," in IEEE International Conference on Computer Vision, 2017.

[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.

[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.

[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.

[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.

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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

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

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

[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.

[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.

[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

[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.

[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.

[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.

[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.

[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.

[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.

[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.

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

[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.

[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.

[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.

[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.

[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.

[45] P. Charbonnier, L. Blanc-F´eraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic

regularization algorithms for computed imaging," in IEEE International Conference on Image Processing, 1994.

[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.

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

[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.

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

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

[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.

[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.

[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.

[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.

[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.

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

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

[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.

[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.