Recurrent Video Deblurring with Blur-Invariant Motion Estimation and Pixel Volumes

HYEONGSEOK SON, POSTECH
JUNYONG LEE, POSTECH
JONGHYEOP LEE, POSTECH
SUNGYUN CHO, POSTECH
SEUNGYONG LEE, POSTECH

For the success of video deblurring, it is essential to utilize information from neighboring frames. Most state-of-the-art video deblurring methods adopt motion compensation between video frames to aggregate information from multiple frames that can help deblur a target frame. However, the motion compensation methods adopted by previous deblurring methods are not blur-invariant, and consequently, their accuracy is limited for blurry frames with different blur amounts. To alleviate this problem, we propose two novel approaches to deblur videos by effectively aggregating information from multiple video frames. First, we present blur-invariant motion estimation learning to improve motion estimation accuracy between blurry frames. Second, for motion compensation, instead of aligning frames by warping with estimated motions, we use a pixel volume that contains candidate sharp pixels to resolve motion estimation errors. We combine these two processes to propose an effective recurrent video deblurring network that fully exploits deblurred previous frames. Experiments show that our method achieves the state-of-the-art performance both quantitatively and qualitatively compared to recent methods that use deep learning.

Additional Key Words and Phrases: video deblurring, pixel volume, blur-invariant motion estimation, deep learning, recurrent network

ACM Reference Format:

1 INTRODUCTION

Videos captured by hand-held devices are vulnerable to motion blur due to camera shakes and object motion. Video deblurring increases the visual quality of a video, and can also improve the accuracy of other video processing tasks, such as object recognition [Kupyn et al. 2018], tracking [Seibold et al. 2017], and 3D reconstruction [Lee and Lee 2013]. Numerous video deblurring methods have been proposed, including deconvolution-based [Kim and Lee 2015; Ren et al. 2017], aggregation-based [Cho et al. 2012; Delbracio and Sapiro 2015b], and deep learning-based approaches [Kim et al. 2017, 2018; Nah et al. 2019; Su et al. 2017; Wang et al. 2019]. Video deblurring methods typically rely on information about motion between neighboring frames, because it can be used for roughly estimating the blur and aligning frames to provide different captures of the same scene. Deconvolution-based methods [Kim and Lee 2015; Ren et al. 2017] exploit motion between neighboring frames to estimate spatially-varying blur kernels, which are then used to deconvolve blurry frames and restore sharp frames. Aggregation-based methods [Cho et al. 2012; Delbracio and Sapiro 2015b] use motion information to directly aggregate sharp pixels from neighboring frames. Recent deep learning-based methods [Kim...
et al. 2017, 2018; Nah et al. 2019; Su et al. 2017; Wang et al. 2019] train deep convolutional neural networks (CNNs) to directly produce deblurred frames from multiple input frames. While some deep learning-based methods do not explicitly assume aligned input frames, Su et al. [2017] show that rough alignment can increase deblurring quality.

However, accurate estimation of motion is challenging in the presence of blur, and incorrectly estimated motion may cause structural deformations during the motion compensation process, and thereby eventually degrade the deblurring quality. To resolve this problem, we introduce two novel approaches: blur-invariant motion estimation learning and pixel volume-based motion compensation.

First, to estimate motion accurately, we adopt LiteFlowNet [Hui et al. 2018] and train it with a blur-invariant loss so that the trained network can estimate blur-invariant optical flow between frames. We refer to our deblurring network as a blur-invariant motion estimation network (BIMNet). Second, for effective motion compensation, instead of aligning a frame by warping, we construct a pixel volume that consists of multiple matching candidates for each pixel. Compared to a warped frame, our pixel volume provides additional information for a deblurring network to robustly restore a sharp frame, even under slight motion estimation errors.

By leveraging pixel volume-based motion compensation with blur-invariant motion estimation, we propose an effective video deblurring framework that is based on a recurrent CNN structure. To produce a deblurred result of the current frame, our framework takes four video frames as input: the previous, current, and next blurry input frames, and the deblurred result of the previous frame. In this way, we can exploit restored information of the previous frame as well as other information in neighboring input frames. Our framework then estimates motion between the current and previous frames in a blur-invariant way. By using BIMNet, then uses the estimated motion to construct a pixel volume from the deblurred previous frame. Finally, our deblurring network restores a sharp image for the current frame using the pixel volume with input blurry frames.

We refer to our deblurring network as a pixel volume-based deblurring network (PVDNet). Due to BIMNet and pixel volume, PVDNet can produce visually pleasing deblurring results (Fig. 1). Experimental results show that our framework achieves the state-of-the-art performance both quantitatively and qualitatively.

Our main contributions are summarized as follows.

- We present blur-invariant learning of a motion estimation network for blurry video frames, which is essential for accurate alignment of neighboring frames.
- We propose a novel pixel volume of matching candidates for motion compensation, which provides additional information for robust reconstruction of sharp frames.
- We propose an effective recurrent CNN framework based on BIMNet and pixel volume for video deblurring.

2 RELATED WORK

2.1 Video deblurring

**Deconvolution-based approaches.** Typical single image deblurring approaches [Cho and Lee 2009; Hirsch et al. 2011; Kim et al. 2013; Pan et al. 2016; Whyte et al. 2012; Xu and Jia 2010, 2012] estimate blur kernels and restore latent sharp images by applying deconvolution with the estimated kernels. However, estimating spatially-varying blur, which is common in the real world, is a severely ill-posed problem. Therefore, many approaches assume specific blur models and focus on limited cases such as camera shakes [Hirsch et al. 2011; Whyte et al. 2012], moving objects [Kim et al. 2013; Pan et al. 2016] and depth variation [Xu and Jia 2012].

Video deblurring based on deconvolution also requires estimation of spatially-varying blur kernels. Early approaches [Bar et al. 2007; Wulff and Black 2014] used segmentation maps to reduce the ill-posedness and to effectively process blur caused by moving objects. Later methods [Kim and Lee 2015; Ren et al. 2017] proposed video deblurring frameworks based on optical flow to model inter-frame motion. These methods alternatingly perform optical flow estimation and image deblurring. However, they still assume relatively simple blur models, so can fail when applied to real-world videos that include complex blur.

**Multi-frame aggregation-based approaches.** A few methods [Cho et al. 2012; Delbracio and Sapiro 2015b; Matsushita et al. 2006] assume that input frames are differently blurred, and have different partial information about latent sharp frames. Then, by aggregating such partial information on the current frame, deblurring can be done without deconvolution. These methods are usually performed in combination with frame alignment. Matsushita et al. [2006] align video frames by using a homography, then blend them to remove blur. Cho et al. [2012] introduced patch-based local search to increase the accuracy of pixel-wise correspondence between video frames. The patch-based local search yields visually pleasing results by directly blending sharp patches from neighboring frames. To achieve video deblurring, Delbracio and Sapiro [2015b] combined a multi-frame deblurring method for burst shot [Delbracio and Sapiro 2015a] with optical flow-based frame alignment. As these methods do not use deconvolution, they could be computationally more efficient and produce less artifacts. However, they cannot restore sharp frames when all input frames are blurry.

**Deep learning-based approaches.** Several approaches use neural networks that are trained to automatically aggregate information from neighboring frames and reconstruct deblurred frames. Su et al. [2017] proposed a CNN that receives multiple frames concatenated together as input. They also showed rough motion compensation between input frames helps to deblur difficult scenes with large motions. Kim et al. [2017] and Nah et al. [2019] presented recurrent CNNs for video deblurring, and showed that utilizing previous results can improve deblurring quality. However, these methods do not work well for videos with large motion, as they do not use motion compensation. Recently, a spatio-temporal transformer network [Kim et al. 2018] was proposed to improve motion compensation performance for video restoration tasks, such as video super-resolution and video deblurring. The network estimates optical flow from multiple neighboring frames together to effectively handle occlusions. More recently, Wang et al. [2019] proposed an alignment module based on deformable convolutions for video restoration tasks that allows more effective utilization of information from other video frames.
3 OUR VIDEO DEBLURRING FRAMEWORK

Our video deblurring framework consists of three modules: a blur-invariant motion estimation network (BIMNet), a pixel volume generator, and a pixel volume-based deblurring network (PVDNet) (see Fig. 2). We first train BIMNet; after it has converged, we combine the two networks with the pixel volume generator. We then fix the parameters of BIMNet and train PVDNet by training the entire network.

The framework takes three consecutive input blurry frames $I^b_{t-1}$, $I^b_t$, and $I^b_{t+1}$, and the deblurred result of the previous frame $I^\text{est}_{t-1}$ as input where $I^b_t$ is the t-th input blurry frame. Our BIMNet first estimates optical flow $W_{t-1}$ from $I^b_{t-1}$ to $I^b_t$ (Sec. 3.1). The pixel volume generator then takes $W_{t-1}$ and $I^\text{est}_{t-1}$, and constructs a pixel volume by using pixel values of $I^\text{est}_{t-1}$ according to $W_{t-1}$ (Sec. 3.2). Finally, our PVDNet takes the pixel volume $V_{t-1}$ and the input blurry frames $I^b_{t-1}$, $I^b_t$, and $I^b_{t+1}$, and produces a deblurring result $I^\text{est}_t$ for the current frame $I^b_t$ (Sec. 3.3).

We feed a pixel volume $V_{t-1}$ obtained from the restored frame $I^\text{est}_{t-1}$ into PVDNet, then the use of $I^b_t$ as input of PVDNet may seem unnecessary. However, $I^b_{t-1}$ is occasionally sharper than $I^\text{est}_{t-1}$ especially when camera motion is not significant, and PVDNet can take advantage of $I^b_{t-1}$ in such cases. In addition, we apply motion compensation not to $I^b_{t-1}$ and $I^b_{t+1}$ but only to $I^\text{est}_{t-1}$. This choice may seem counterintuitive, but preliminary experiments demonstrated that it maximizes the deblurring performance of PVDNet (Appendix B.2).

3.1 Blur-invariant motion estimation learning

Our BIMNet adopts the network architecture of LiteFlowNet [Hui et al. 2018], which takes two images and generates optical flow between them. FlowNet [Dosovitskiy et al. 2015] and its follow-up works, FlowNet2 [Ilg et al. 2017] and LiteFlowNet [Hui et al. 2018], have a Siamese network structure for the encoder, and we found that the structure enables effective blur-invariant feature learning (Appendix A.1). We adopt LiteFlowNet for the network structure of BIMNet, because LiteFlowNet shows comparable accuracy to FlowNet2 with far fewer parameters (Sec. 4.2). Originally, LiteFlowNet was trained using blur-free datasets that provide ground truth optical flow maps, such as Scene Flow [Mayer et al. 2016], Sintel [Butler et al. 2012], KITTI [Geiger et al. 2012], and Middlebury [Scharstein et al. 2014]. However, no available dataset has ground truth optical flow maps for blurry images. Thus, we train our BIMNet in a self-supervised way by using a blurred video dataset [Su et al. 2017] that contains pairs of sharp and blurred videos.

A successfully trained BIMNet should be able to produce accurate optical flow from a pair of video frames regardless of the amounts of blur that the frames include. To train the network to acquire this property, we generate four pairs of frames from each pair of consecutive frames in sharp and blurred videos: $(I^s_{t-1}, I^b_t)$, $(I^s_{t-1}, I^b_{t-1})$, $(I^b_{t-1}, I^s_t)$, and $(I^b_{t-1}, I^b_t)$, where $I^s_t$ and $I^b_t$ are the t-th frames in sharp and blurred videos, respectively. For each image pair $(I^s_{t-1}, I^b_t)$, where $\alpha, \beta \in \{s, b\}$, our BIMNet estimates optical flow $W^\alpha\beta_t$ from $I^s_t$ to $I^\beta_{t-1}$.

Then, we train BIMNet with a blur-invariant loss $L^\alpha\beta_{\text{BIM}}$ for every pair $(\alpha, \beta)$:

$$L^\alpha\beta_{\text{BIM}} = \text{MSE}(\text{Warp}(I^\beta_{t-1}, W^\alpha\beta_t), I^\alpha_t),$$

where $\text{MSE}$ computes the mean squared error between two images and $\text{Warp}(I^\beta_{t-1}, W^\alpha\beta_t)$ is a warping of $I^\beta_{t-1}$ by $W^\alpha\beta_t$. Warp can be easily implemented using the sampling layer of the spatial transformer network [Jaderberg et al. 2015]. The key idea of $L^\alpha\beta_{\text{BIM}}$ is that it induces BIMNet to learn optical flow between sharp frames, $I^s_t$ and $I^s_{t-1}$, as the ground truth regardless of the blur types of input images, $I^\beta_{t-1}$ and $I^\beta_t$.

A straightforward idea for blur-invariant learning would be to estimate optical flow maps between sharp video frames and use them as ground truth labels for training optical flow estimation between blurry frames. However, in preliminary experiments, this approach did not improve accuracy as anticipated. One possible reason for the failure is that optical flow estimated from sharp video frames had the low quality. Specifically, there may be a domain gap between our training dataset and the optical flow datasets that have been used by recent learning-based optical flow methods, such as, FlowNet [Dosovitskiy et al. 2015], FlowNet2 [Ilg et al. 2017], and LiteFlowNet [Hui et al. 2018]. Thus, optical flow maps estimated from sharp frames may have errors that can hinder using them as ground-truth labels. In contrast, our proposed approach can effectively train BIMNet without the need for accurate ground-truth labels.
3.2 Pixel volume for motion compensation

Previous video deblurring methods [Kim et al. 2018; Su et al. 2017] first align video frames by warping them before deblurring. However, during the warping process, structural artifacts may occur due to inaccurately estimated motion, which can eventually degrade the final deblurring results. To relieve this problem, Liao et al. [2015] proposed the draft-ensemble approach that computes multiple optical flow maps. However, this approach is computationally inefficient because it requires multiple optical flow estimation. To tackle the problem effectively and efficiently, we propose a novel pixel volume that does not require multiple optical flow estimation but can still provide multiple candidates for matching pixels between images.

For a pixel \((x, y)\) of \(I_b^1\), we consider a spatial window of size \(k \times k\) centered at \((x, y)\), where we fix \(k = 5\). For each pixel \((x + \Delta x, y + \Delta y)\) in the window, where \(\Delta x, \Delta y \in [-\lfloor k/2 \rfloor, \ldots, \lfloor k/2 \rfloor]\), we have a matched pixel \((x', y')\) in \(I_{est}^1\) determined by optical flow \(W_{est}^1\). Then, its neighboring pixel \((x' - \Delta x, y' - \Delta y)\) in \(I_{est}^1\) is a matching candidate for pixel \((x, y)\). Consequently, we have \(k^2\) matching candidates in \(I_{est}^1\) for \((x, y)\). We collect such matching candidates for all pixels in \(I_b^1\) and stack them to obtain a pixel volume \(V_{1-1}\) of size \(W \times H \times k^2\), where \(W\) and \(H\) are the width and height of a video frame, respectively. The resulting pixel volume is in the form of a pile of \(k^2\) images, where each image consists of pixels sampled from \(I_{est}^1\), and the image for \((\Delta x, \Delta y) = (0, 0)\) is the warping of \(I_b^1\) by \(W_{est}^1\).

Fig. 3 illustrates the construction of a pixel volume.

The occurrence of multiple matching candidates in a pixel volume provide additional information that can reduce the deblurring artifacts caused by inaccurate motion estimation. Suppose that the pixel matching for \((x, y)\) determined by \(W_{est}^1\) is incorrect due to small errors in motion estimation. Even in that case, if one of the \(k^2\) neighbors of \((x, y)\) has a correct match, the pixel pile of \(V_{1-1}\) at \((x, y)\) would contain a correct match for \((x, y)\). Moreover, a pixel volume provides an additional cue for motion compensation based on the majority. When the neighbors of a pixel have correct matches, the pixel volume can provide multiple duplicates of the correct match in the matching candidates for the pixel. Then, the majority cue can increase the reliability of motion compensation in deblurring the current frame \(I_b^1\). In Sec. 4.4, we provide detailed statistics of candidate pixels in a pixel volume.

In short, our pixel volume approach leads to the performance improvement of video deblurring by utilizing the multiple candidates in a pixel volume in two aspects: 1) in most cases, the majority cue for the correct match would help as the statistics in Sec. 4.4 shows, and 2) in other cases, PVDNet would exploit multiple candidates to estimate the correct match referring to nearby pixels that have majority cues. This is a clear advantage over warping based alignment, because both majority cue and exploitation of multiple values cannot be provided by a single warped image that has only one candidate for each pixel.

Fig. 4 compares motion compensation using warping and a pixel volume. The warped result of \(I_{est}^1\) may contain deformation artifacts caused by motion estimation errors (Fig. 4c top). We also consider the majority-based warped result of \(I_{est}^1\) (Fig. 4c middle). In majority-based warping, for each pixel, we choose the most frequent pixel value (the mode) among the \(k^2\) candidates in a pixel volume. The majority-based warped frame contains similar artifacts to the simple warping result. In regions where the motion estimation results are inaccurate, the estimated flows are not consistent among neighboring pixels. Then, the most frequent values may not always be the best for motion compensation. As a result, the deblurring results obtained using such warped frames still contain a small amount of blur (Fig. 4d top and middle). In contrast, even in regions where motion estimation results are inaccurate, probably one or a few
of the candidates in the pixel volume are correct. To demonstrate this phenomenon, we reconstructed an ideally warped image (Fig. 4c bottom) by choosing the most similar pixel to the ground truth sharp frame \( I_s^t \) among the \( k^2 \) candidates for each pixel. The ideally warped image contains successfully reconstructed structures with less distortion, which shows the benefit of using a pixel volume. Consequently, the deblurring result obtained using the pixel volume (Fig. 4a) shows successfully restored sharp structures (Fig. 4d bottom). In Sec. B.4, we provide a quantitative analysis on the effect of a pixel volume.

**Differences from relevant approaches.** Our pixel volume construction scheme shares the same basis with the propagation step of the PatchMatch algorithm [Barnes et al. 2009]: both methods exploit the spatial coherency of neighbor matches. However, our pixel volume scheme is distinct from PatchMatch in the following points. The PatchMatch propagation is designed to accelerate correspondence search, and therefore does not construct an explicit volume. Moreover, it is not straightforward to adopt PatchMatch to deep learning models. In contrast, our pixel volume scheme is specifically designed to improve the robustness of motion compensation and easily integrated into deep learning networks.

There have been other approaches relevant to our pixel volume scheme. Cho et al. [2012] also considered multiple candidates for video deblurring by using a window-based patch search. However, their method does not exploit the spatial coherency of neighbor matches, and is not designed for neural networks. Conventional stereo matching algorithms also construct a cost volume, but such cost volumes contain not pixel values but matching costs, and are therefore not directly applicable to video deblurring. Cost volumes do not exploit the spatial coherency of neighbor matches either.

### 3.3 Pixel volume based video deblurring network

**PVDNet** employs the U-Net [Ronneberger et al. 2015] architecture that has shown to be effective for deblurring [Nah et al. 2017; Su et al. 2017; Tao et al. 2018]. We use symmetric skip-connections between the encoder and decoder to reconstruct a result image from encoded features while preserving image structures (Fig. 5). We add 12 residual blocks between the encoder and decoder to effectively refine blurry feature maps in low-resolution. We apply ReLU after all convolution and deconvolution layers except the layers that are connected by skip-connections; in these layers, the ReLU is applied after the summation of skip-connected features. We refer the readers to the appendix for a detailed architecture of **PVDNet**.

For early fusion of input images and a pixel volume, we add initial feature transform layers to the network. As aforementioned, our **PVDNet** takes four inputs (the previous, current, and next input frames, and pixel volume of the deblurred previous frame). Instead of directly concatenating the inputs, we transform and concatenate them using feature transform layers. We divide the inputs into two groups: the concatenated consecutive frames, and the pixel volume. We feed each group into the feature transform layers and fuse their features by concatenation. We use a grayscale pixel volume to reduce training and computational complexity. Our preliminary test shows that the model using a grayscale pixel volume has comparable performance to the model using a color pixel volume (Appendix B.2). The network has a long skip-connection from the input frame \( I_b^t \) to the network output to make the network predict only residuals while preserving overall contents of the input frame.

To train **PVDNet**, we use mean absolute difference between deblurred frames and their corresponding ground truth sharp frames as the loss function. Although the network does not have any specific constraints or loss functions for temporal coherence, it generally produces temporally smooth results as our recurrent network utilizes the deblurred previous frame to process the current frame.

### 4 EXPERIMENTS

#### 4.1 Implementation and training details

To train both **BIMNet** and **PVDNet**, we used a dataset of synthetically blurred videos [Su et al. 2017], which was generated by capturing sharp videos at high frame rate, and averaging adjacent frames. The dataset consists of 71 pairs of a blurry video and its corresponding sharp video, which provide 6,708 pairs of 1280×720 blurred and
sharp frames in total. The videos in the dataset have blur caused by camera shake and object motion. We used 61 videos in the dataset as our training set, and 10 videos as our test set as Su et al. [2017] did. The training and test sets consist of 5,708 and 1,000 pairs of images, respectively.

Our training process consists of two steps. We first train BIMNet, and then PVDNet while fixing BIMNet. To train BIMNet, we fine-tuned LiteFlowNet [Hui et al. 2018] over the pre-trained weights provided by the authors. For fine-tuning, we randomly selected two consecutive frames from the training video set. We then randomly cropped the same areas of size 256×256 from the selected frames, and fed them to the network. We used normalized pixel values from -1 to 1, and set the batch size to eight. We used Adam optimizer [Kingma and Ba 2015] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We initially set the learning rate to 0.0001, and decayed it with a decaying rate 0.1 for every 100,000 iterations. The network converged after ~200,000 iterations. The training took about two days on a PC with an Nvidia GeForce TiTAN-Xp (12GB).

To train PVDNet, we follow a conventional training strategy such as [Kim et al. 2017] to generate mini-batches for recurrent video processing. Specifically, we randomly sampled sequences of 13 consecutive frames from the training set. We sampled eight sequences, i.e., we used batch size of eight. Then, we randomly cropped the same areas of size 256×256 from the frames in each sequence. For each sequence, we iterate from the first to last cropped frame. For each iteration, we set the current cropped frame as the target blurry frame $I^b_t$ and compute the gradient using the result of the previous frame $I^e_{t-1}$, as well as the current target blurry frame and its neighboring blurry frames. Then, we update the network parameters by using backpropagation. The backpropagation is performed only for the current iteration, i.e., the gradient does not propagate to the network at the previous state through $I^e_{t-1}$. After iterating to the last frames, we repeat the process from the random sampling of video frame sequences. For the initial frame of each sequence, we set $I^e_{t-1} = I^b_t$ and $I^b_{t-1} = I^b_t$. We used Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as was done for BIMNet. We trained PVDNet for 400,000 iterations with a learning rate of 0.0001. Then, we further trained the network for 200,000 iterations with the learning rate of 0.000025. The training of PVDNet took about four days on the aforementioned environment.

We do not train BIMNet and PVDNet jointly, because we use the explicit blur-invariant loss for motion estimation learning. BIMNet had been already trained with highly accurate supervision based on ground truth sharp images, so end-to-end training of BIMNet and PVDNet together did not improve the accuracy or quality of video deblurring. Similarly, Gast and Roth [2019] reported that fixed pre-trained optical flow models [Dosovitskiy et al. 2015; Sun et al. 2018] can be used to improve video deblurring accuracy quite well without sophisticated joint training. It is also known that end-to-end training schemes do not always improve the model performances in various image processing tasks, such as object detection [He et al. 2017; Ren et al. 2015], image super-resolution [Ge et al. 2018], and image inpainting [Ren et al. 2019].

4.2 Motion estimation performance

We first verify the effectiveness of our blur-invariant motion estimation learning by comparing our BIMNet with state-of-the-art optical flow estimation methods: TVL1 [Sánchez et al. 2013], FlowNet2 [Ilg et al. 2017], and LiteFlowNet [Hui et al. 2018]. TVL1 is a traditional optimization-based approach. FlowNet2 improves FlowNet in terms of accuracy at the expense of an increased model size and slower computation. LiteFlowNet is a light-weight variant of FlowNet with a much reduced model size and faster computation. We considered two versions of BIMNet: BIMNet$_{FN2}$, which uses FlowNet2 network architecture; and BIMNet$_{LFN}$, which uses LiteFlowNet network architecture. BIMNet$_{LFN}$ is the model adopted in our final framework. Our blur-invariant motion estimation learning uses blurry video frames and the blur-invariant loss in the training process. To verify the effect of each component, we consider four versions of FlowNet2 and LiteFlowNet: FlowNet2$_{SS}$, LiteFlowNet$_{SS}$, FlowNet2$_{++}$, and LiteFlowNet$_{++}$. FlowNet2$_{SS}$ and LiteFlowNet$_{SS}$ are fine-tuned versions of FlowNet2 and LiteFlowNet using sharp video frames in our training set, respectively. FlowNet2$_{++}$ and LiteFlowNet$_{++}$ are fine-tuned using all combinations of sharp and blurry video frames in our training set as done for BIMNet$_{FN2}$ and BIMNet$_{LFN}$. To fine-tune these variants, we did not use a blur-invariant loss in Eq. (1). Instead, we used the loss function defined as:

\[
L^αβ = MSE(Warp(I^α_{t−1}, W^αβ), I^β_t),
\]

which is blur-variant, where $α, β ∈ \{s, b\}$.

To measure the performance of each method, we measured the warping accuracy of motion estimation results. Specifically, we first estimated the optical flow from $I^b_t$ to $I^e_{t-1}$. We then obtained $I^s_t$ by warping $I^e_{t-1}$ with the computed optical flow, and measured the difference between the warping result $I^s_t$ and its ground truth sharp frame $I^s_t$ in terms of peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [Wang et al. 2004]. Following Su et al. [2017], to handle the ambiguity in the pixel location caused by blur, we measure PSNRs and SSIMs using an approach [Köhler et al. 2012] that first aligns two images using global translation.

To analyze the performance of different methods on blurry videos more clearly, we also measured the performance on the sharpest 10% and the blurriest 10% video frames in each video in the test set. To choose the sharpest and blurriest 10% frames, we measured the bluriness of all test frames as PSNRs between blurry frames and their corresponding sharp ground truth frames. We assume that a test frame with a low PSNR is likely to have large blur, because blur is the primary factor of structural differences from its ground truth.

Table 1 shows a summary of the quantitative comparison using our test set. For the entire test set, BIMNet$_{FN2}$ and BIMNet$_{LFN}$ achieve the best and second-best PSNR and comparable SSIM. The result shows that BIMNet$_{FN2}$ significantly outperforms the other methods on the severely blurred frames by a large margin, while it performs comparably on the relatively sharp frames. BIMNet$_{LFN}$ was slightly inferior to BIMNet$_{FN2}$, but still outperforms all the other methods. Moreover, BIMNet$_{LFN}$ has only 5.4M parameters, whereas BIMNet$_{FN2}$ has 150M parameters. Thus, we adopt BIMNet$_{LFN}$ for our framework.
Table 1. Quantitative comparison between BIMNet and other optical flow estimation methods on blurry video frames. We computed average errors between $I^b_{t-1}$ and $Warp(I^s_{t-1}, W^{bb})$ in PSNR and SSIM, where $W^{bb}$ is the optical flow estimated from $I^b_{t-1}$ to $I^b_t$.

<table>
<thead>
<tr>
<th>Model</th>
<th>Entire test set</th>
<th>Sharpest 10%</th>
<th>Blurred 10%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR (dB)</td>
<td>SSIM</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>TVL1 [Sánchez et al. 2015]</td>
<td>28.20</td>
<td>0.890</td>
<td>30.73</td>
</tr>
<tr>
<td>FlowNet2 [Ilg et al. 2017]</td>
<td>27.85</td>
<td>0.903</td>
<td>29.75</td>
</tr>
<tr>
<td>LiteFlowNet [Hui et al. 2018]</td>
<td>27.64</td>
<td>0.895</td>
<td>29.99</td>
</tr>
<tr>
<td>FlowNet2SS</td>
<td>28.21</td>
<td><strong>0.909</strong></td>
<td>30.57</td>
</tr>
<tr>
<td>LiteFlowNetSS</td>
<td>27.91</td>
<td>0.893</td>
<td>30.12</td>
</tr>
<tr>
<td>FlowNet2ss</td>
<td>27.89</td>
<td>0.868</td>
<td>30.20</td>
</tr>
<tr>
<td>LiteFlowNetss</td>
<td>27.23</td>
<td>0.811</td>
<td>30.06</td>
</tr>
<tr>
<td>BIMNetF2N2</td>
<td><strong>28.79</strong></td>
<td>0.907</td>
<td>30.61</td>
</tr>
<tr>
<td>BIMNetLFN</td>
<td>28.43</td>
<td>0.900</td>
<td>30.29</td>
</tr>
</tbody>
</table>

Fig. 6. Qualitative comparison between BIMNet and other optical flow estimation methods. (a) and (f) are input blurry frames $I^b_{t-1}$ and $I^b_t$, respectively. From (b) to (i), we show the warped images $I^s_{t-1} = Warp(I^s_{t-1}, W^{bb})$, where the optical flow maps $W^{bb}$ have been estimated from $I^b_{t-1}$ and $I^b_t$ by different methods.

To identify the source of the improvement of BIMNet, we compared FlowNet2, FlowNet2SS, FlowNet2ss, and BIMNetF2N2. Compared to FlowNet2, FlowNet2SS shows improvement for relatively sharp frames, but yields inferior results for blurry frames as a result of overfitting to sharp video frames. FlowNet2ss performs worse than FlowNet2SS for both sharp and blurry frames; this result shows that simply including blurry frames in training does not improve the accuracy of optical flow estimation on blurry frames. Compared to these, BIMNetF2N2 that is trained using our blur-invariant loss shows improvements in all cases. A similar trend can also be found from LiteFlowNetSS, LiteFlowNetss, and BIMNetLFN. This result verifies that our blur-invariant loss is the source of the improvement in motion estimation accuracy in the presence of blur.

Fig. 6 shows a qualitative comparison of different optical flow methods. To visually compare the quality of different optical flow methods, we estimated optical flow maps between two consecutive blurred video frames, $I^b_{t-1}$ and $I^b_t$, using different optical flow methods, and warped the sharp video frame $I^s_{t-1}$ corresponding to $I^b_{t-1}$ using the estimated optical flow maps. As the figure shows, the results of the other methods contain severely distorted structures due to errors in their optical flow maps. In contrast, the results of BIMNetF2N and BIMNetLFN show much less distortions.

4.3 Ablation study

To show the effectiveness of the BIMNet and pixel volume in our video deblurring framework, we performed an ablation study. We tested five models in this study. The baseline is PVDNet that receives consecutive blurry frames as well as a deblurred previous frame without any motion compensation. Then, one by one, we add motion compensation (MC) using LiteFlowNetSS, blur-invariant motion compensation (BIC) using BIMNetLFN, and pixel volume (PV) into the baseline model. Another straightforward idea to increase the performance is to jointly train motion estimation and deblurring. To verify the effectiveness of our BICM against such a joint approach,
we also include a jointly trained model (MC$_{c2e}$ + PV) in which the motion estimation module is first initialized by LiteFlowNet$_{SS}$ and then jointly trained with PVDNet by using the deblurring loss in an end-to-end manner.

Our model that uses BIMC and PV shows the best PSNR and SSIM (Table 2). For the sharpest 10% of the frames of each video in the test set, all the models, even the baseline model without motion compensation, show high performances. In contrast, for the blurriest 10% of the frames, each of BIMC and PV achieves drastic improvement over the models with and without MC; these results indicate that BIMC and PV are effective for handling large blur.

The table also shows that joint training of motion estimation and PVDNet (MC$_{c2e}$ + PV) does not increase but decreases the deblurring performance compared to MC + PV. This result occurs because the deblurring loss does not provide good guidance for accurate motion estimation, and the loss adversarially affects its training.

Fig. 7 shows a qualitative comparison. The models without BIMC cannot compensate for large motions, and therefore did not well restore the big red characters in the blue box. The models without PV cannot compensate for small distortions caused by motion estimation errors, and did not properly recover the small black characters in the green box. Our full model with BIMC and PV could deblur successfully in both cases, and produced the best results.

4.4 Statistics of candidate pixels in pixel volume
From each video in the test set of [Su et al. 2017], we sampled ten frames, and gathered 100 frames. We then constructed PVs for those frames, and investigated the distribution of pixel values in the volumes. In the PVs, 95.1% of the pixels had majority values in the candidates, where a majority value means more than a half of the $k^2$ candidates for a pixel have the same value. We then investigated the accuracy of those majority values. As our PVs are constructed from deblurred previous frames, they may not have perfect matches with ground truth pixel values. Thus, to measure the accuracy, we counted the number of majority values that are close enough to the ground truth pixel values, i.e., the pixel value differences are within $\pm 0.255$ scale. The accuracy of those majority values was 89.5% when $c = 0$ (perfect match) and increased to 92% for $c = 2$.

While most of the pixels in the pixel volume have correct majority values, the pixel volume is still helpful even for deblurring regions that have no majority values or wrong ones. To validate it, we measured the accuracy of deblurring results on those regions using two models: one that uses PV, and the other that uses naïve warping. The accuracy was measured in the same way as above with $c = 2$.

In regions where the majority values are correct, the model that uses PV achieved accuracy of 97.42%, and the model that uses naïve warping achieved 96.91%. For regions in which majority values are wrong, the model that uses PV (57.35%) shows a significantly higher accuracy than the model that uses naïve warping (50.49%). Likewise, for the region that has no majority values, the model that uses PV (89.97%) also shows higher accuracy than the model that uses naïve warping (87.77%). This result confirms that our PV helps our deblurring network obtain the correct values even when the majority pixels in the PV are wrong.

4.5 Motion compensation vs. larger model
Our method adopts an additional motion estimation network, BIMNet, so the accuracy gain could be simply a result of increasing the total model size. To investigate on this possibility, we prepared variants of our deblurring framework with different model sizes either with or without motion compensation. Specifically, we prepared small, medium, and large versions of PVDNet. The medium model is the model described in Sec. 3.3. The small model has half the number of filters in every convolution layer, and half the number of residual blocks of the medium model. The large model has twice the number of filters in every convolution layer. Then, we prepared six variants of our framework either with or without our motion compensation scheme that combines BIMNet and PV, and measured the models’ deblurring accuracies.

Table 3 shows that the small and medium models with our motion compensation scheme outperformed the large model without motion compensation despite their significantly smaller model sizes.

<table>
<thead>
<tr>
<th>Model</th>
<th>Entire test set</th>
<th>Sharpest 10%</th>
<th>Blurriest 10%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR SSIM</td>
<td>PSNR SSIM</td>
<td>PSNR SSIM</td>
</tr>
<tr>
<td>input</td>
<td>27.40 0.819</td>
<td>33.02 0.936</td>
<td>22.77 0.665</td>
</tr>
<tr>
<td>baseline</td>
<td>30.66 0.891</td>
<td>34.76 0.954</td>
<td>26.15 0.780</td>
</tr>
<tr>
<td>MC</td>
<td>31.23 0.901</td>
<td>35.10 0.957</td>
<td>26.69 0.798</td>
</tr>
<tr>
<td>BIMC</td>
<td>31.46 0.912</td>
<td>35.15 0.958</td>
<td>27.28 0.820</td>
</tr>
<tr>
<td>MC + PV</td>
<td>31.40 0.910</td>
<td>35.13 0.958</td>
<td>27.19 0.817</td>
</tr>
<tr>
<td>MC$_{c2e}$ + PV</td>
<td>31.32 0.909</td>
<td>34.95 0.956</td>
<td>27.21 0.818</td>
</tr>
<tr>
<td>BIMC + PV</td>
<td><strong>31.63 0.915</strong></td>
<td><strong>35.18 0.958</strong></td>
<td><strong>27.46 0.828</strong></td>
</tr>
</tbody>
</table>
Table 3. Performance comparison of PVDNets with different model sizes either with or without MC. The unit of the number of parameters is million (M). Values in the parentheses are the numbers of parameters of PVDNet and BIMNet.

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR (dB)</th>
<th>SSIM</th>
<th>Parameters (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>30.03</td>
<td>0.880</td>
<td>0.7</td>
</tr>
<tr>
<td>Medium</td>
<td>30.66</td>
<td>0.891</td>
<td>5.1</td>
</tr>
<tr>
<td>Large</td>
<td>30.85</td>
<td>0.898</td>
<td>19.6</td>
</tr>
<tr>
<td>Small + MC</td>
<td>30.95</td>
<td>0.904</td>
<td>6.1 (0.7 + 5.4)</td>
</tr>
<tr>
<td>Medium + MC</td>
<td>31.63</td>
<td>0.915</td>
<td>10.5 (5.1 + 5.4)</td>
</tr>
<tr>
<td>Large + MC</td>
<td>31.75</td>
<td>0.918</td>
<td>25.0 (19.6 + 5.4)</td>
</tr>
</tbody>
</table>

Fig. 8 also shows that the small and medium models with motion compensation could recover small-scale structures and textures better than the large model without motion compensation. Both quantitative and qualitative comparisons clearly show the superiority of our motion compensation scheme over simply increasing the size of the model.

4.6 Comparison with other deblurring methods
We compared our method with six recent state-of-the-art deep learning-based deblurring methods, including two single image-based methods [Nah et al. 2017; Tao et al. 2018], and four video deblurring methods [Kim et al. 2017; Nah et al. 2019; Su et al. 2017; Wang et al. 2019]. For the single image deblurring methods of Nah et al. [2017] and Tao et al. [2018], we applied them to each input frame independently. The method of Su et al. [2017] takes consecutive blurry video frames as input, which can be either motion compensated using optical flow or not. In our comparison, we included both versions with and without motion compensation, which are denoted by DVD-MC and DVD-noMC, respectively. For motion compensation for DVD-MC, we used the method of Sánchez et al. [2013] as done in [Su et al. 2017]. The method of Wang et al. [2019], which we refer to as EDVR, also takes consecutive blurry video frames as input. The methods of Kim et al. [2017] and Nah et al. [2019], which we refer to as OVD and IFI-RNN, respectively, adopt recurrent network structures that take an intermediate feature map of the previous frame without motion compensation.

For fair comparison in terms of model size, we also included a variant of IFI-RNN with an increased model size, which we refer to as IFI-RNN-L, and a smaller version of our framework (the small PVDNet in Sec. 4.4), which we refer to as Ours-small. IFI-RNN-L has twice the number of residual blocks and also twice the number of channels in every convolution layer. Finally, for fair comparison with EDVR, we also include another variant of our framework (Ours w/ CA), which is trained using cosine annealing (CA) [Loshchilov and Hutter 2017], a sophisticated learning rate scheduling strategy adopted to EDVR.

For evaluation, we used three test sets in the comparison. Specifically, in addition to Su et al. [2017]’s dataset, we used another popular video deblurring dataset [Nah et al. 2017], synthetically generated in a similar way to Su et al. [2017]’s dataset. We also used real-world videos [Su et al. 2017] to check the generalization ability of our model.

Comparison using Su et al. [2017]’s dataset. For fair comparison, we trained all the models except for OVD [Kim et al. 2017] and DVD [Su et al. 2017] using the dataset of Su et al. [2017]. For this training, we used the source code provided by the authors and their own training strategies. The training code of OVD is not available, so we used its pre-trained model provided by the authors. We also used the model of DVD provided by the authors; it had already been trained with Su et al.’s dataset. For quantitative evaluation, we excluded the first frames of test videos, because most video deblurring methods cannot successfully handle them.

In the quantitative comparison (Table 4), our model achieved higher PSNR and SSIM than all the other methods, including single image and video deblurring methods. Compared to both DVD-noMC and DVD-MC, even Ours-small performed much better despite its significantly smaller size, because DVD cannot utilize previous deblurred frames due to the lack of a recurrent structure. Compared to OVD, Ours-small achieved PSNRs more than 1 dB higher. Ours-small has more parameters than OVD, but most are used for motion estimation; the deblurring network, PVDNet, has fewer parameters than OVD. This comparison clearly shows the benefit of our motion compensation scheme. Similarly, compared to IFI-RNN, Ours-small as well as Ours outperformed it both in PSNR and SSIM, in spite of the smaller deblurring network. Compared to IFI-RNN-L and EDVR, which have larger numbers of parameters, Ours achieved higher PSNR and SSIM even though the total number of parameters is smaller. Finally, Ours trained with CA outperformed EDVR by a margin of 0.49 dB, proving the effectiveness of our framework.

The advantage of our approach is emphasized in the qualitative comparison (Fig. 9). As the input video frame (Fig. 9a) is severely blurred, the single image deblurring method of Tao et al. [2018] fails to recover fine-scale structures and produces blurry results. DVD-MC recovers fine-scale structures better, but produces distorted structures due to motion compensation errors. IFI-RNN, IFI-RNN-L, and EDVR also fail to restore fine-scale structures due to large motions in the input frames. In contrast, thanks to our motion compensation scheme, our method effectively restores fine details without distortions even in the presence of large inter-frame motion and blur.

Comparison using Nah et al. [2017]’s dataset. For fair comparison, all the methods except for OVD [Kim et al. 2017] were trained from
Table 4. Quantitative comparison of our video deblurring framework with recent state-of-the-art methods on the dataset of [Su et al. 2017]. Values in the parentheses are the numbers of parameters or computations of PVDNet and BIMNet. For DVD-MC, the numbers of parameters and computations for motion estimation are not included, as DVD-MC does not use a deep learning-based model for motion estimation. CA denotes the cosine annealing scheduling method for learning rate [Loshchilov and Hutter 2017], which EDVR adopts for its training.

<table>
<thead>
<tr>
<th></th>
<th>PSNR (dB)</th>
<th>SSIM</th>
<th>Parameters (M)</th>
<th>Computations (BMACs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Nah et al. 2017]</td>
<td>29.85</td>
<td>0.880</td>
<td>75.7</td>
<td>2364</td>
</tr>
<tr>
<td>[Tao et al. 2018]</td>
<td>30.53</td>
<td>0.894</td>
<td>3.8</td>
<td>1175</td>
</tr>
<tr>
<td>DVD-noMC [Su et al. 2017]</td>
<td>30.29</td>
<td>0.888</td>
<td>15.3</td>
<td>557</td>
</tr>
<tr>
<td>DVD-MC [Su et al. 2017]</td>
<td>30.47</td>
<td>0.898</td>
<td>15.3</td>
<td>557</td>
</tr>
<tr>
<td>OVD [Kim et al. 2017]</td>
<td>30.08</td>
<td>0.873</td>
<td>0.9</td>
<td>206</td>
</tr>
<tr>
<td>IFI-RNN [Nah et al. 2019]</td>
<td>31.04</td>
<td>0.903</td>
<td>1.6</td>
<td>200</td>
</tr>
<tr>
<td>IFI-RNN-L [Nah et al. 2019]</td>
<td>31.67</td>
<td>0.916</td>
<td>12.2</td>
<td>1,425</td>
</tr>
<tr>
<td>EDVR [Wang et al. 2019]</td>
<td>31.82</td>
<td>0.916</td>
<td>23.6</td>
<td>2,739</td>
</tr>
<tr>
<td>Ours-small</td>
<td>31.25</td>
<td>0.908</td>
<td>6.1 (0.7 + 5.4)</td>
<td>424 (105 + 319)</td>
</tr>
<tr>
<td>Ours</td>
<td>31.95</td>
<td>0.920</td>
<td>10.5 (5.1 + 5.4)</td>
<td>936 (617 + 319)</td>
</tr>
<tr>
<td>Ours w/ CA</td>
<td><strong>32.31</strong></td>
<td><strong>0.926</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 9. Qualitative comparison on synthetically blurred video frames of Su et al. [2017]’s dataset.

In a qualitative comparison, our method using cosine annealing (Ours w/ CA) achieved better or comparable performance to the other state-of-the-art methods including IFI-RNN-L [Nah et al. 2019] and EDVR [Wang et al. 2019] despite its smaller model size. In this comparison, we also included our method with a similar model size to that of EDVR, to show the effectiveness of our explicit motion compensation approach. Specifically, to increase the model size to match that of EDVR, we used a larger PVDNet that has 24 residual blocks with 192 channels. Our larger model (Ours-large w/ CA) clearly outperforms EDVR both in PSNR and SSIM.

In a qualitative comparison, the input blurred frame (Fig. 10a) has severe blur caused by large camera motion. Consequently, the results of all the other methods have remaining blur. On the other hand, our result (Fig. 10g) shows clearly restored details with no remaining blur.

Comparison using real-world videos. Finally, we qualitatively compare the performance of our approach to those of the state-of-the-art methods on real-world data (Fig. 11). The datasets [Nah et al. 2017; Su et al. 2017] that were used for training and evaluation are synthetically generated using high-speed cameras, so it may not fully represent the characteristics of real blurry videos. Thus, evaluation
using real-world data is essential for predicting the generalization ability of a method. For evaluation, we used the real blurry video test set in Su et al. [2017].

In this comparison, the input video frame (Fig. 11a) is severely blurred by camera motions, so the characters in the frame are illegible. The single image deblurring method of Tao et al. [2018] (Fig. 11b) failed to restore the characters properly and introduced noisy artifacts. The results of the video deblurring methods, DVD-MC (Fig. 11c) and IFI-RNN (Fig. 11e) have remaining blurs. Compared to the results of the other methods, the results of EDVR (Fig. 11d) and IFI-RNN-L (Fig. 11f) show relatively better restored details because of their large model sizes, but small characters remain illegible either. In contrast, both Ours-small (Fig. 11g) and Ours (Fig. 11h) more clearly restore characters with less artifacts despite their smaller models than EDVR and IFI-RNN-L.

In another qualitative comparison (Fig. 12), the input video frames suffer from large blur. DVD-MC restored small-scale structures relatively well, but its results still have remaining blur and noisy artifacts. IFI-RNN-L completely failed to restore sharp frames, possibly due to large motions between input frames that are difficult to handle without explicit motion compensation. On the contrary, our results clearly restored sharp details. Additional examples can be found in the supplementary material.

4.7 Computational cost analysis

To process a single 1280×720 video frame, our framework takes about 0.11s with an NVIDIA Titan Xp GPU, and our framework can process images of arbitrary size, because both BIMNet and PVDNet consist of convolutional layers. Specifically, BIMNet, PVDNet, and pixel volume construction take 0.04s, 0.04s, and 0.03s per frame, respectively. The pixel volume construction step takes relatively long computation time. However, its code is un-optimized PyTorch code that uses for-loops, which are known to be slow in PyTorch. We believe that the code can be further optimized using CUDA programming. The computational overhead caused by a pixel volume in PVDNet is smaller than that of a single convolution layer in our framework. In the aspect of memory usage, the size of a pixel volume is $W \times H \times k^2$, and $k$ is set to 5 in our experiments. In contrast, the first convolution layer of PVDNet uses 64 channels so its size is $W \times H \times 64$. Thus, a PV requires only ~39% memory usage of the first convolution layer. In the aspect of computation time, once a pixel volume is constructed, it is processed by convolution operations like other layers. Thus, the overhead computation time caused by a
pixel volume is also smaller than that of a single convolution layer in our framework.  

We compared the computational costs of our method and others. We used the number of Multiply-accumulate (MAC) operations required to process a single 1280×720 video frame as the measurement.
Our method (936 BMACs) has a much smaller computational cost than IFI-RNN-L (1425 BMACs) and EDVR (2739 BMACs) despite achieving higher deblurring accuracy (Table 4).

5 CONCLUSION

In this paper, we presented two novel approaches for motion compensation to effectively deliver previously deblurred information. We proposed simple but effective blur-invariant learning for accurate estimation of optical flow between blurred video frames. In addition, we constructed a pixel volume that contains matching candidates for robustly handling motion estimation errors, and fed it to our deblurring network instead of directly feeding a motion compensated warped frame. Finally, we proposed an effective video deblurring framework based on the proposed motion compensation approaches.

Our framework achieved the state-of-the-art deblurring performance, as demonstrated by extensive experiments on various challenging videos with large blur. Combination of blur-invariant motion estimation learning and a pixel volume can improve motion compensation without significant computational overhead or changes in network design. We expect that these two modules would be applicable to other video processing tasks, such as video super-resolution that requires sub-pixel-level motion compensation.

Similarly to previous deblurring methods [Kim et al. 2017, 2018; Nah et al. 2017; Su et al. 2017; Tao et al. 2018; Wang et al. 2019], in the synthetic datasets [Nah et al. 2017; Su et al. 2017], we use temporal center frames as ground truth sharp frames for training both BIMNet and PVDDNet. However, complex blur such as circular blur may introduce ambiguity, because more than one camera motions with different temporal centers can produce the same shape of blur; this ambiguity may confuse the networks and degrade their performances. This problem would not be severe in practice given the success of recent deblurring methods that use temporal center frames for ground truths, but analyzing its effect and resolving it can be a good future direction.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by the Ministry of Science and ICT, Korea, through IITP grants (SW Star Lab, IITP-2015-0-00174: Artificial Intelligence Graduate School Program (POSTECH), IITP-2019-0-01906) and NRF grants (NRF-2018R1A5A1060031; NRF-2020R1C1C1014863).

APPENDIX

A BIMNET

A.1 Effect of Siamese structure on blur-invariant feature learning

As described in Sec. 3.1, we adopt LiteFlowNet [Hui et al. 2018] for our BIMNet as LiteFlowNet has a Siamese network architecture that is effective to learn blur-invariant features. In this section, we conduct an experiment to verify the effectiveness of the Siamese architecture on blur-invariant feature learning.

To this end, we compare two different networks for optical flow estimation: FlowNetS and FlowNetC [Dosovitskiy et al. 2015]. FlowNetS and FlowNetC share almost the same network architecture except that FlowNetC has a Siamese network at the beginning of the network, while FlowNetS has a plain CNN architecture. Using the Siamese network architecture, FlowNetC first extracts a feature for each pixel and then match the extracted features to estimate optical flow. While both FlowNetS and FlowNetC can run as stand-alone networks for optical flow estimation, LiteFlowNet combines them together for obtaining high-quality optical flow.

To investigate the effect of the Siamese network architecture, we evaluated the performance of FlowNetS and FlowNetC on blur-invariant feature learning. Specifically, we fine-tuned both pre-trained networks with our blur-invariant loss $L_{\text{BIM}}$. The networks were originally pre-trained with datasets of [Butler et al. 2012; Geiger et al. 2012; Scharstein et al. 2014] and we used the dataset of Su et al. [2017] for fine-tuning them. Table 6 compares FlowNetS and FlowNetC with their fine-tuned versions. While $L_{\text{BIM}}$ improves the accuracy of both networks for blurry videos, the improvement of FlowNetC is much larger than that of FlowNetS, showing that the Siamese network architecture of FlowNetC learns to extract blur-invariant features more effectively than a plain CNN architecture.

A.2 Performance of BIMNet on different types of image pairs $(I_{t-1}^s, I_t^b)$

Sec. 4.2 verified the effectiveness of our blur-invariant motion estimation learning by comparing BIMNet with state-of-the-art optical flow estimation methods, by measuring PSNR and SSIM between $\text{Warp}(I_{t-1}^s, W^{bb})$ and $I_t^b$. In this section, we provide an additional evaluation of the performance of BIMNet on images with different types of blur. Specifically, we evaluated the performance of BIMNet for the other three cases: BS (Warp$(I_{t-1}^s, W^{bb})$, $I_t^b$), SB (Warp$(I_{t-1}^s, W^{bb})$, $I_t^s$), and SS (Warp$(I_{t-1}^s, W^{ss})$, $I_t^s$).

Table 7 shows the evaluation results of BS, SB, and SS. In the cases of BS and SB, BIMNet$_{FN2}$ and BIMNet$_{LEN}$ achieved the best and second-best performances, respectively, in both PSNR and SSIM, clearly outperforming all the other methods. In the case of SS, while FlowNet2S, which is trained for sharp images, performs the best in PSNR, BIMNet$_{FN2}$ and BIMNet$_{LEN}$ perform comparably in PSNR.
Table 7. Quantitative comparison between BIMNet and other optical flow estimation methods on different types of image pairs \((I_{t-1}^b, I_t^b)\). We computed average errors between \(I_t^b\) and Warp\((I_{t-1}^b, W^{(a|b)})\) in PSNR and SSIM, where \(W^{(a|b)}\) is the optical flow estimated from \(I_{t-1}^a\) to \(I_t^b\). BS, SB, and SS denote \((a, b)\): (b, s), (s, b), and (s, s), respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>BS PSNR (dB)</th>
<th>SSIM</th>
<th>SB PSNR (dB)</th>
<th>SSIM</th>
<th>SS PSNR (dB)</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVL1 [Sánchez et al. 2013]</td>
<td>27.54 0.869</td>
<td></td>
<td>28.08 0.875</td>
<td></td>
<td>30.78 0.904</td>
<td></td>
</tr>
<tr>
<td>FlowNet2 [Ilg et al. 2017]</td>
<td>28.22 0.903</td>
<td></td>
<td>28.04 0.898</td>
<td></td>
<td>30.03 0.928</td>
<td></td>
</tr>
<tr>
<td>LiteFlowNet [Hui et al. 2018]</td>
<td>27.47 0.882</td>
<td></td>
<td>26.51 0.858</td>
<td></td>
<td>29.49 0.923</td>
<td></td>
</tr>
<tr>
<td>FlowNet2SS</td>
<td>27.41 0.869</td>
<td></td>
<td>27.76 0.864</td>
<td></td>
<td>30.95 0.910</td>
<td></td>
</tr>
<tr>
<td>LiteFlowNet1s</td>
<td>27.90 0.885</td>
<td></td>
<td>27.57 0.874</td>
<td></td>
<td>30.64 0.927</td>
<td></td>
</tr>
<tr>
<td>LiteFlowNetS</td>
<td>27.86 0.824</td>
<td></td>
<td>27.53 0.837</td>
<td></td>
<td>30.64 0.868</td>
<td></td>
</tr>
<tr>
<td>LiteFlowNet3s</td>
<td>27.02 0.811</td>
<td></td>
<td>26.83 0.789</td>
<td></td>
<td>29.76 0.887</td>
<td></td>
</tr>
<tr>
<td>BIMNet(_b_s)</td>
<td><strong>28.98 0.909</strong></td>
<td></td>
<td><strong>29.29 0.910</strong></td>
<td></td>
<td><strong>30.68 0.928</strong></td>
<td></td>
</tr>
<tr>
<td>BIMNet(_s_b)</td>
<td>28.85 0.901</td>
<td></td>
<td>28.86 0.901</td>
<td></td>
<td>30.64 0.928</td>
<td></td>
</tr>
</tbody>
</table>

and better than the other methods in SSIM, showing that both of them perform reasonably well even for sharp images.

B PVDNET

B.1 Network architecture

A detailed architecture of PVDNet is given in Table 8.

B.2 Preliminary tests for PVDNet

As described in Sec. 3, our framework takes neighboring frames \(I_{t-1}^b\) and \(I_t^b\) without motion compensation. We also construct a pixel volume of \(I_{t-1}^b\) using its grayscale pixel values. While this combination may seem less optimal, we found that it actually achieves the highest quality in our preliminary tests. In this section, we summarize the results of our preliminary tests.

In the tests, we considered the following options for PVDNet:

- warping vs. pixel volume for motion compensation,
- grayscale pixel volume vs. color pixel volume,
- applying motion compensation only to \(I_{t-1}^{est}\) vs. to all images of \(I_{t-1}^{est}\), \(I_{t-1}^b\), and \(I_t^b\), and
- using only \(I_{t-1}^{est}\) vs. all images of \(I_{t-1}^{est}\), \(I_{t-1}^b\), and \(I_t^b\) for the neighboring frame information. Here, motion compensation is applied only to \(I_{t-1}^{est}\).

To compare the different options, we prepared various variants of PVDNet.

Table 9 shows the variants of PVDNet and their performances in PSNR and SSIM. In the table, ‘all’ in the first column (‘MC target’) means that all three images, \(I_{t-1}^{est}\), \(I_{t-1}^b\), and \(I_t^b\), are motion-compensated. On the other hand, ‘\(I_{t-1}^{est}\) only’ means that only \(I_{t-1}^{est}\) is motion-compensated while \(I_{t-1}^b\) and \(I_t^b\) are used as they are. In the second column (‘MC type’), ‘gray PV’ means that a pixel volume is constructed using grayscale pixel values, while ‘color PV’ means that it is constructed using RGB values. ‘warp’ means that simple warping is used instead of a pixel volume for motion compensation. In the third column (‘use \(I_{t-1}^b\) and \(I_t^b\)’), ‘no’ means that a PVDNet does not take \(I_{t-1}^b\) and \(I_t^b\) as input, and uses only \(I_{t-1}^{est}\). Our final model corresponds to \(I_{t-1}^{est}\), ‘gray PV’, ‘yes’ in the table.

From Table 9, we can make the following observations: 1) using a pixel volume consistently outperforms using warping, showing the effectiveness of a pixel volume, 2) using a grayscale pixel volume achieves comparable performance to using a color pixel volume.
while it requires much less memory and computation, 3) interestingly, applying motion compensation only to \( I_{t-1}^{est} \) leads to higher quality results than applying motion compensation to all images, and 4) using the neighboring input blurry frames \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \) helps increase the performance, compared to using \( I_{t-1}^{est} \) only. Based on these observations, we chose \( \{ I_{t-1}^{est}, \text{gray PV}, \text{yes} \} \) as our final model.

It may seem counter-intuitive that applying motion compensation to all images decreases the deblurring performance. A possible reason is that the neighboring input blurry frames \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \) would help deblurring only in limited cases. Specifically, if there is a large motion between \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \), it is likely that \( I_{t-1}^{b} \) is blurry, and motion-compensating the blurry neighboring frame would not help deblurring much. In other words, \( I_{t-1}^{b} \) helps deblurring only when there is slight motion, and in that case, keeping the original relatively sharp pixels could be more beneficial than distorting the image for small motion compensation. The same arguments hold for \( I_{t+1}^{b} \). Our PVDNet is trained to learn to selectively use the information in motion-compensated \( I_{t-1}^{est} \) and input neighboring frames, \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \), and consequently leads to higher performance.

### B.3 Non-recurrent PVDNet

While our PVDNet adopts a recurrent framework, non-recurrent approaches like DVD [Su et al. 2017] have also been widely adopted so far. Thus, we study the performance of non-recurrent variants of our framework and the effect of our motion compensation on them. In this study, we prepared four different non-recurrent variants of our framework: baseline\(_{nr} \), baseline\(_{nr} + MC \), baseline\(_{nr} + BIMC \), and baseline\(_{nr} + BIMC + PV \), similarly to the experiment in Table 2. For baseline\(_{nr} \), we modified the architecture of PVDNet so that it receives three consecutive input blurry frames without a previous deblurred frame. For all the variants, motion compensation is applied to the previous and next input blurry frames instead of a deblurred previous frame. Specifically, baseline\(_{nr} \) takes input blurry frames without any motion compensation, while baseline\(_{nr} + MC \) and baseline\(_{nr} + BIMC \) take warped input blurry frames whose motions are estimated by LitFlowNetSS and BIMNet, respectively.

#### Table 9. Preliminary test results of various variants of PVDNet. The first column ('MC target') denotes the target frames among \( I_{t-1}^{est}, I_{t-1}^{b} \), and \( I_{t+1}^{b} \), for which motion compensation is applied. The second column ('MC type') denotes the type of motion compensation applied for the frames in 'MC target'. The third column ('use \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \)') denotes whether PVDNet takes \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \) as input or not. We computed the average errors between deblurred results \( I_{t}^{est} \) and ground truths \( I_{t} \) in PSNR and SSIM.

<table>
<thead>
<tr>
<th>MC target</th>
<th>MC type</th>
<th>use ( I_{t-1}^{b} ) and ( I_{t+1}^{b} )</th>
<th>PSNR (dB)</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td>gray PV</td>
<td>yes</td>
<td>31.30</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td>color PV</td>
<td>yes</td>
<td>31.63</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td>warp</td>
<td>yes</td>
<td>31.11</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>no</td>
<td>31.10</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>yes</td>
<td>31.11</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>yes</td>
<td>31.11</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>no</td>
<td>31.10</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>yes</td>
<td>31.11</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>yes</td>
<td>31.11</td>
</tr>
<tr>
<td>all</td>
<td>( I_{t-1}^{est} ) only</td>
<td></td>
<td>no</td>
<td>31.10</td>
</tr>
</tbody>
</table>

baseline\(_{nr} + BIMC + PV \), we modified the network architecture to take two pixel volumes of the previous and next input blurry frames, and the current target frame. Specifically, the network has two input branches similarly to the network in Fig. 2. One branch takes the current target blurry frame \( I_{t}^{b} \) instead of three blurry frames. On the other hand, the other branch takes two pixel volumes of \( I_{t-1}^{b} \) and \( I_{t+1}^{b} \) and concatenates them before the feature transform layer.

Table 10 shows the performance evaluation result of the non-recurrent models. Both baseline models of the non-recurrent and recurrent networks show similar accuracy (30.65 dB and 30.66 dB) as shown in Table 10 and Table 2. Although BIMC and a pixel volume improve the deblurring quality the most, all kinds of motion compensation in Table 10 do not lead to significant improvement of the deblurring quality. Su et al. [2017] also reported a similar result where motion compensation does not improve deblurring quality much. We guess that this is because of the higher training complexity of non-recurrent deblurring. In non-recurrent deblurring, there would be no sharp information that can help deblur the current target frame, in contrast to recurrent deblurring that uses a deblurred previous frame. This can induce the network trained to less utilize information from neighboring frames.

#### Table 10. Effectiveness of blur invariant motion estimation and pixel volumes with the non-recurrent PVDNet.

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR (dB)</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline(_{nr} )</td>
<td>30.65</td>
<td>0.889</td>
</tr>
<tr>
<td>baseline(_{nr} + MC )</td>
<td>30.79</td>
<td>0.896</td>
</tr>
<tr>
<td>baseline(_{nr} + BIMC )</td>
<td>30.84</td>
<td>0.899</td>
</tr>
<tr>
<td>baseline(_{nr} + BIMC + PV )</td>
<td>30.87</td>
<td>0.900</td>
</tr>
</tbody>
</table>

#### Table 11. Deblurring performance of our PVDNet according to different motion compensation approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple warping</td>
<td>31.46</td>
</tr>
<tr>
<td>Majority-based warping</td>
<td>31.47</td>
</tr>
<tr>
<td>Naïve PV</td>
<td>31.48</td>
</tr>
<tr>
<td>Our PV</td>
<td>31.63</td>
</tr>
</tbody>
</table>

In this section, we compare our pixel volume approach to two additional motion compensation approaches to analyze the effect of a pixel volume. As described in Sec. 3.2, our pixel volume approach leads to the performance improvement in video deblurring by utilizing multiple candidates with majority cues in a pixel volume. One natural question would be whether simply using the majority values in the pixel volume or multiple candidates without majority cues also leads the performance improvement. Therefore, we experimented with two other motion compensation approaches.

First, instead of our PV, we simply fed the deblurring result of the previous sharp frame warped by the majority-based warping to PVDNet. In majority-based warping, we built a PV based on the estimated flow, then chose the most frequent pixel value (the mode) among the \( k^2 \) candidates for each pixel. Second, instead of constructing our PV, we constructed a naïve PV in a way that it can provide...
multiple candidates without spatial coherency, corresponding to simply increase the number of candidates. Specifically, for each spatial position in a PV, we do not exploit the flow vectors of the neighbor pixels, but we simply collect pixels in the \( k \times k \) window centered at the estimated position by BIMNet. This approach is similar to the neighbor search of [Cho et al. 2012] and the cost volume construction of conventional stereo matching algorithms. For the experiment and comparison, we conducted the same procedure used in Sec. 4.3.

Table 11 summarizes the performance of different approaches. The performance improvement of majority-based warping over simple warping is marginal, as the warped images from simple warping and majority-based warping would be similar to each other, as shown in Fig. 4c. The performance improvement of the naïve PV over simple warping is also marginal. Although the naïve PV contains multiple candidates, those multiple candidates do not have majority values due to the way of construction. In contrast, the multiple candidates in our PV are obtained by exploiting flow information of neighboring pixels and provide the majority cue.

### B.5 Patch size of pixel volume

In this experiment, we analyze the deblurring performance change by a different patch size of a pixel volume. We tested five patch sizes (1, 3, 5, 7, 9) and found that 5×5 performs the best (Table 12). Ideally, it can be expected that a larger patch size can help the network to handle more spacious flow errors, but a large patch size rather decreases the performance in practice. This may be because a pixel volume generated by large patches contains too many candidates to consider, which exceeds the capability of PVDNet.

<table>
<thead>
<tr>
<th>Patch size</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB)</td>
<td>31.42</td>
<td>31.54</td>
<td>31.63</td>
<td>31.55</td>
<td>31.11</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.912</td>
<td>0.912</td>
<td>0.915</td>
<td>0.913</td>
<td>0.905</td>
</tr>
</tbody>
</table>

### B.6 Pixel volume vs flow aggregation using conv layers

Our framework explicitly constructs a pixel volume for motion compensation. One natural question would be whether it would be possible to learn to aggregate flow information of neighboring pixels using additional convolution layers instead of explicit construction of a pixel volume. To verify this, we may consider the following two options:

1. Additional convolution layers at the end of BIMNet to refine the flow result
2. Additional convolution layers at the beginning of PVDNet to aggregate flow results

Specifically, the first option refines the flow result through additional convolution layers. This option is equivalent to adopt a larger network architecture for BIMNet. To verify whether this option brings a significant improvement, we trained a larger network with additional five convolution layers for BIMNet. We then measured the deblurring performance, but found no noticeable improvement.

The second option is not trivial. To deblur the current frame, video deblurring requires pixel value information of the previous frame, not its flow information [Cho et al. 2012; Kim et al. 2017, 2018; Su et al. 2017; Wang et al. 2019]. Thus, even if we somehow aggregate flow information using additional convolution layers, we still need to warp the previous deblurred frame. Moreover, aggregating and refining flow information is not trivial as well. Instead, our pixel-volume approach provides an effective way that allows the network to aggregate pixel value information from multiple candidate pixels. In addition, a pixel volume takes smaller memory than a single convolution layer (Sec. 4.7).

### B.7 Deblurring of first few frames

Fig. 13 shows the deblurring results of the first few frames of a video. The top row shows input consecutive blurry frames, while the bottom row shows their corresponding results. When our method starts to deblur the video from the first frame, the first frame has no deblurred previous frame, and its deblurring result still has a small amount of remaining blur, e.g., the details of the wheel are slightly blurry. Nevertheless, from the second frame, our method can utilize the deblurred result of the previous frame, so it produces a clearer result with sharper details.

Fig. 14 visualizes the average PSNRs of the deblurring results at different time steps. The average PSNRs are measured from the test set of Su et al.’s dataset [Su et al. 2017]. The average PSNR of the first frames is low as they have no deblurring results of the
average PSNR quickly raises and gets stabilized. This proves the practicality and effectiveness of our approach despite its recurrent structure that requires deblurring results of previous frames.

REFERENCES


Li Xu and Jiaya Jia. 2010. Two-Phase Kernel Estimation for Robust Motion Deblurring. In Proc. ICCP.

Li Xu and Jiaya Jia. 2012. Depth-Aware Motion Deblurring. In Proc. ICCP.

Li Xu and Jiaya Jia. 2013. Cost-Segmentation Guided Motion Deblurring. In Proc. CVPR.

Li Xu and Jiaya Jia. 2018. Scale-recurrent Convolutional Neural Network for Dynamic Scene Deblurring. In Proc. CVPR.


Li Xu and Jiaya Jia. 2010. Two-Phase Kernel Estimation for Robust Motion Deblurring. In Proc. CVPR.


Li Xu and Jiaya Jia. 2010. Two-Phase Kernel Estimation for Robust Motion Deblurring. In Proc. CVPR.