

BMBC: Bilateral Motion Estimation with Bilateral Cost Volume for Video Interpolation

Junheum Park¹[0000–0002–9094–128X], Keunsoo Ko¹[0000–0003–0203–4530],
Chul Lee²[0000–0001–9329–7365], and Chang-Su Kim¹[0000–0002–4276–1831]

¹ School of Electrical Engineering, Korea University, Seoul, Korea

² Department of Multimedia Engineering, Dongguk University, Seoul, Korea
{jhpark,ksko}@mcl.korea.ac.kr, chullee@dongguk.edu,
changsukim@korea.ac.kr

Abstract. Video interpolation increases the temporal resolution of a video sequence by synthesizing intermediate frames between two consecutive frames. We propose a novel deep-learning-based video interpolation algorithm based on bilateral motion estimation. First, we develop the bilateral motion network with the bilateral cost volume to estimate bilateral motions accurately. Then, we approximate bi-directional motions to predict a different kind of bilateral motions. We then warp the two input frames using the estimated bilateral motions. Next, we develop the dynamic filter generation network to yield dynamic blending filters. Finally, we combine the warped frames using the dynamic blending filters to generate intermediate frames. Experimental results show that the proposed algorithm outperforms the state-of-the-art video interpolation algorithms on several benchmark datasets. The source codes and pre-trained models are available at <https://github.com/JunHeum/BMBC>.

Keywords: Video interpolation, bilateral motion, bilateral cost volume

1 Introduction

A low temporal resolution causes aliasing, yields abrupt motion artifacts, and degrades the video quality. In other words, the temporal resolution is an important factor affecting video quality. To enhance temporal resolutions, many video interpolation algorithms [2–4, 12, 14, 16–18, 20–22] have been proposed, which synthesize intermediate frames between two actual frames. These algorithms are widely used in applications, including visual quality enhancement [32], video compression [7], slow-motion video generation [14], and view synthesis [6]. However, video interpolation is challenging due to diverse factors, such as large and nonlinear motions, occlusions, and variations in lighting conditions. Especially, to generate a high-quality intermediate frame, it is important to estimate motions or optical flow vectors accurately.

Recently, with the advance of deep-learning-based optical flow methods [5, 10, 25, 30], flow-based video interpolation algorithms [2, 3, 14] have been developed, yielding reliable interpolation results. Niklaus *et al.* [20] generated intermediate frames based on the forward warping. However, the forward warping may

cause interpolation artifacts because of the hole and overlapped pixel problems. To overcome this, other approaches leverage the backward warping. To use the backward warping, intermediate motions should be obtained. Various video interpolation algorithms [2–4, 14, 16, 20, 32] based on the bilateral motion estimation approximate these intermediate motions from optical flows between two input frames. However, this approximation may degrade video interpolation results.

In this work, we propose a novel video interpolation network, which consists of the bilateral motion network and the dynamic filter generation network. First, we predict six bilateral motions: two from the bilateral motion network and the other four through optical flow approximation. In the bilateral motion network, we develop the bilateral cost volume to facilitate the matching process. Second, we extract context maps to exploit rich contextual information. We then warp the two input frames and the corresponding context maps using the six bilateral motions, resulting in six pairs of warped frame and context map. Next, these pairs are used to generate dynamic blending filters. Finally, the six warped frames are superposed by the blending filters to generate an intermediate frame. Experimental results demonstrate that the proposed algorithm outperforms the state-of-the-art video interpolation algorithms [2, 3, 18, 22, 32] meaningfully on various benchmark datasets.

This work has the following major contributions:

- We develop a novel deep-learning-based video interpolation algorithm based on the bilateral motion estimation.
- We propose the bilateral motion network with the bilateral cost volume to estimate intermediate motions accurately.
- The proposed algorithm performs better than the state-of-the-art algorithms on various benchmark datasets.

2 Related Work

2.1 Deep-learning-based video interpolation

The objective of video interpolation is to enhance a low temporal resolution by synthesizing intermediate frames between two actual frames. With the great success of CNNs in various image processing and computer vision tasks, many deep-learning-based video interpolation techniques have been developed. Long *et al.* [18] developed a CNN, which takes a pair of frames as input and then directly generates an intermediate frame. However, their algorithm yields severe blurring since it does not use a motion model. In [19], PhaseNet was proposed using the phase-based motion representation. Although it yields robust results to lightning changes or motion blur, it may fail to faithfully reconstruct detailed texture. In [21, 22], Niklaus *et al.* proposed kernel-based methods that estimate an adaptive convolutional kernel for each pixel. The kernel-based methods produce reasonable results, but they cannot handle motions larger than a kernel’s size.

To exploit motion information explicitly, flow-based algorithms have been developed. Niklaus and Liu [20] generated an intermediate frame from two con-

secutive frames using the forward warping. However, the forward warping suffers from holes and overlapped pixels. Therefore, most flow-based algorithms are based on backward warping. In order to use backward warping, intermediate motions (*i.e.* motion vectors of intermediate frames) should be estimated. Jiang *et al.* [14] estimated optical flows and performed bilateral motion approximation to predict intermediate motions from the optical flows. Bao *et al.* [3] approximated intermediate motions based on the flow projection. However, large errors may occur when two flows are projected onto the same pixel. In [2], Bao *et al.* proposed an advanced projection method using the depth information. However, the resultant intermediate motions are sensitive to the depth estimation performance. To summarize, although the backward warping yields reasonable video interpolation results, its performance degrades severely when intermediate motions are unreliable or erroneous. To solve this problem, we propose the bilateral motion network to estimate intermediate motions directly.

2.2 Cost volume

A cost volume records similarity scores between two data. For example, in pixel-wise matching between two images, the similarity is computed between each pixel pair: one in a reference image and the other in a target image. Then, for each reference pixel, the target pixel with the highest similarity score becomes the matched pixel. The cost volume facilitates this matching process. Thus, the optical flow estimation techniques in [5, 10, 30, 31] are implemented using cost volumes. In [5, 10, 31], a cost volume is computed using various features of two video frames, and optical flow is estimated using the similarity information in the cost volume through a CNN. Sun *et al.* [30] proposed a partial cost volume to significantly reduce the memory requirement while improving the motion estimation accuracy based on a reduced search region. In this work, we develop a novel cost volume, called bilateral cost volume, which is different from the conventional volumes in that its reference is an intermediate frame to be interpolated, instead of one of the two input frames.

3 Proposed Algorithm

Fig. 1 is an overview of the proposed algorithm that takes successive frames I_0 and I_1 as input and synthesizes an intermediate frame I_t at $t \in (0, 1)$ as output. First, we estimate two ‘bilateral’ motions $V_{t \rightarrow 0}$ and $V_{0 \rightarrow t}$ between the input frames. Second, we estimate ‘bi-directional’ motions $V_{0 \rightarrow 1}$ and $V_{1 \rightarrow 0}$ between I_0 and I_1 and then use these motions to approximate four further bilateral motions. Third, the pixel-wise context maps C_0 and C_1 are extracted from I_0 and I_1 . Then, the input frames and corresponding context maps are warped using the six bilateral motions. Note that, since the warped frames become multiple candidates of the intermediate frame, we refer to each warped frame as an intermediate candidate. The dynamic filter network then takes the input frames, and the intermediate candidates with the corresponding warped context maps

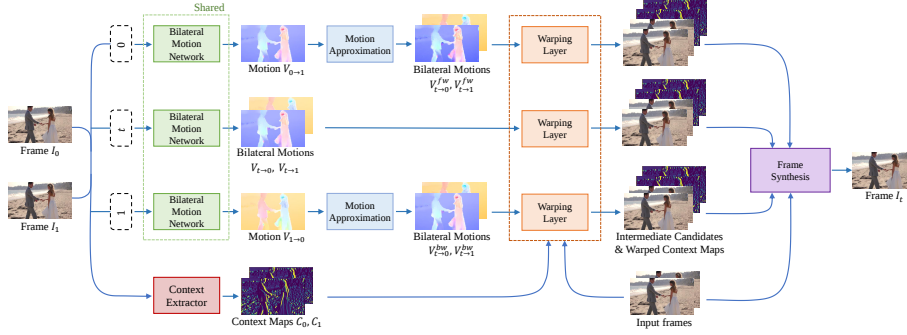


Fig. 1. An overview of the proposed video interpolation algorithm.

to generate the dynamic filters for aggregating the intermediate candidates. Finally, the intermediate frame I_t is synthesized by applying the blending filters to the intermediate candidates.

3.1 Bilateral motion estimation

Given the two input frames I_0 and I_1 , the goal is to predict the intermediate frame I_t using motion information. However, it is impossible to directly estimate the intermediate motion between the intermediate frame I_t and one of the input frames I_0 or I_1 because there is no image information of I_t . To address this issue, we assume linear motion between successive frames. Specifically, we attempt to estimate the backward and forward motion vectors $V_{t \rightarrow 0}(\mathbf{x})$ and $V_{t \rightarrow 1}(\mathbf{x})$ at $\mathbf{x}$, respectively, where $\mathbf{x}$ is a pixel location in I_t . Based on the linear assumption, we have $V_{t \rightarrow 0}(\mathbf{x}) = -\frac{t}{1-t} \times V_{t \rightarrow 1}(\mathbf{x})$.

We develop a CNN to estimate bilateral motions $V_{t \rightarrow 0}$ and $V_{t \rightarrow 1}$ using I_0 and I_1 . To this end, we adopt an optical flow network, PWC-Net [30], and extend it for the bilateral motion estimation. Fig. 2 shows the key components of the modified PWC-Net. Let us describe each component subsequently.

Warping layer: The original PWC-Net uses the previous frame I_0 as a reference and the following frame I_1 as a target. On the other hand, the bilateral motion estimation uses the intermediate frame I_t as a reference, and the input frames I_0 and I_1 as the target. Thus, whereas the original PWC-Net warps the feature c_1^l of I_1 toward the feature c_0^l of I_0 , we warp both features c_0^l and c_1^l toward the intermediate frame, leading to $c_{0 \rightarrow t}^l$ and $c_{1 \rightarrow t}^l$, respectively. We employ the spatial transformer networks [11] to achieve the warping. Specifically, a target feature map c_{tgt} is warped into a reference feature map c_{ref} using a motion vector field by

$$c_{ref}^w(\mathbf{x}) = c_{tgt}(\mathbf{x} + V_{ref \rightarrow tgt}(\mathbf{x})) \quad (1)$$

where $V_{ref \rightarrow tgt}$ is the motion vector field from the reference to the target.

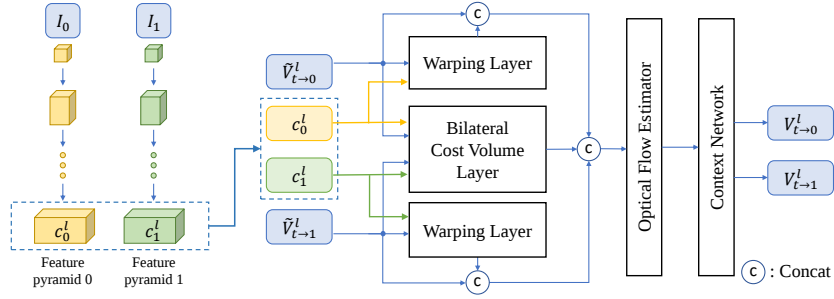


Fig. 2. The architecture of the bilateral motion network: The feature maps c_0^l and c_1^l of the previous and following frames I_0 and I_1 at the l th level and the up-sampled motion fields $\tilde{V}_{t \rightarrow 0}^l$ and $\tilde{V}_{t \rightarrow 1}^l$ estimated at the $(l-1)$ th level are fed into the CNN to generate the motion fields $V_{t \rightarrow 0}^l$ and $V_{t \rightarrow 1}^l$ at the l th level.

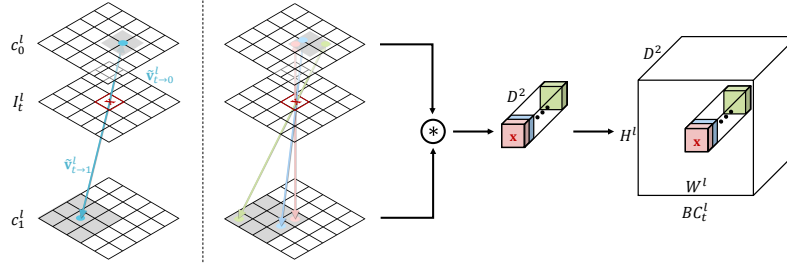


Fig. 3. Illustration of the bilateral cost volume layer for a specific time t .

Bilateral cost volume layer: A cost volume has been used to store the matching costs associating with a pixel in a reference frame with its corresponding pixels in a single target frame [5, 9, 30, 31]. However, in the bilateral motion estimation, because the reference frame does not exist and should be predicted from two target frames, the conventional cost volume cannot be used. Thus, we develop a new cost volume for the bilateral motion estimation, which we refer to as the bilateral cost volume.

Fig. 3 illustrates the proposed bilateral cost volume generation that takes the features c_0^l and c_1^l of the two input frames and the up-sampled bilateral motion fields $\tilde{V}_{t \rightarrow 0}^l$ and $\tilde{V}_{t \rightarrow 1}^l$ estimated at the $(l-1)$ th level. Let $\mathbf{x}$ denote a pixel location in the intermediate frame I_t^l . Then, we define the matching cost as the bilateral correlation between features c_0^l and c_1^l , indexed by the bilateral motion vector that passes through $\mathbf{x}$, given by

$$BC_t^l(\mathbf{x}, \mathbf{d}) = c_0^l(\mathbf{x} + \tilde{V}_{t \rightarrow 0}^l(\mathbf{x}) - 2t \times \mathbf{d})^T c_1^l(\mathbf{x} + \tilde{V}_{t \rightarrow 1}^l(\mathbf{x}) + 2(1-t) \times \mathbf{d}) \quad (2)$$

where $\mathbf{d}$ denotes the displacement vector within the search window $\mathcal{D} = [-d, d] \times [-d, d]$. Note that we compute only $|\mathcal{D}| = D^2$ bilateral correlations to construct a partial cost volume, where $D = 2d + 1$. In the L -level pyramid architecture,

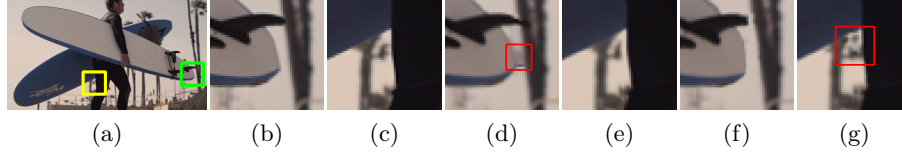


Fig. 4. Comparison of interpolation results obtained by the bilinear motion estimation and the motion approximation: (a) ground-truth intermediate frame; (b), (c) enlarged parts for the green and yellow squares in (a); (d), (e) interpolation results using the bilateral motion estimation; (f), (g) those using the motion approximation. The red squares in (d) and (g) contain visual artifacts caused by motion inaccuracies.

a one-pixel motion at the coarsest level corresponds to 2^{L-1} pixels at the finest resolution. Thus, the search range D of the bilateral cost volume can be set to a small value to reduce the memory usage. The dimension of the bilateral cost volume at the l th level is $D^2 \times H^l \times W^l$, where H^l and W^l denote the height and width of the l th level features, respectively. Also, the up-sampled bilateral motions $\tilde{V}_{t \rightarrow 0}^l$ and $\tilde{V}_{t \rightarrow 1}^l$ are set to zero at the coarsest level.

Most conventional video interpolation algorithms generate a single intermediate frame at the middle of two input frames, *i.e.* $t = 0.5$. Thus, they cannot yield output videos with arbitrary frame rates. A few recent algorithms [2, 14] attempt to interpolate intermediate frames at arbitrary time instances $t \in (0, 1)$. However, because their approaches are based on the approximation, as the time instance gets far from either of the input frames, the quality of the interpolated frame gets worse. On the other hand, the proposed algorithm takes into account the time instance $t \in [0, 1]$ during the computation of the bilateral cost volume in (2). Also, after we train the bilateral motion network with the bilateral cost volume, we can use the shared weights to estimate the bilateral motions at an arbitrary $t \in [0, 1]$. In the extreme cases $t = 0$ or $t = 1$, the bilateral cost volume becomes identical to the conventional cost volume in [5, 10, 24, 30, 31], which is used to estimate the bi-directional motions $V_{0 \rightarrow 1}$ and $V_{1 \rightarrow 0}$ between input frames.

3.2 Motion approximation

Although the proposed bilateral motion network effectively estimates motion fields $V_{t \rightarrow 0}$ and $V_{t \rightarrow 1}$ from the intermediate frame at t to the previous and following frames, it may fail to find accurate motions, especially at occluded regions. For example, Fig. 4(d) and (e) show that the interpolated regions, reconstructed by the bilateral motion estimation, contain visual artifacts. To address this issue and improve the quality of an interpolated frame, in addition to the bilateral motion estimation, we develop an approximation scheme to predict a different kind of bilateral motions $V_{t \rightarrow 0}$ and $V_{t \rightarrow 1}$ using the bi-directional motions $V_{0 \rightarrow 1}$ and $V_{1 \rightarrow 0}$ between the two input frames.

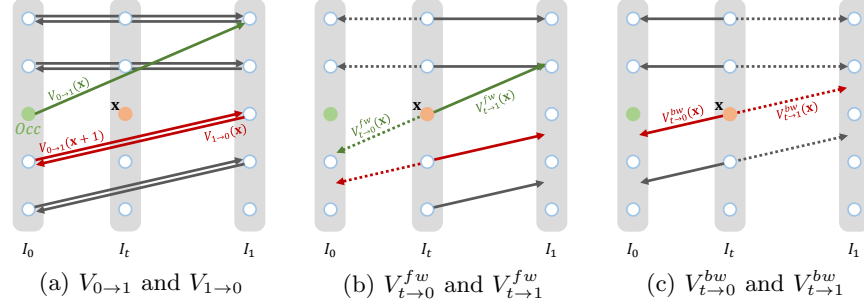


Fig. 5. Motion approximation: bi-directional motions in (a) are used to approximate forward bilateral motions in (b) and backward bilateral motions in (c).

Fig. 5 illustrates this motion approximation, in which each column represents a frame at a time instance and a dot corresponds to a pixel in the frame. In particular, in Fig. 5(a), an occluded pixel in I_0 is depicted by a green dot. To complement the inaccuracy of the bilateral motion at pixel $\mathbf{x}$ in I_t , we use two bi-directional motions $V_{0 \rightarrow 1}(\mathbf{x})$ and $V_{1 \rightarrow 0}(\mathbf{x})$, which are depicted by green and red lines, respectively. We approximate two forward bilateral motions $V_{t \rightarrow 1}^{fw}$ and $V_{t \rightarrow 0}^{fw}$ in Fig. 5(b) using $V_{0 \rightarrow 1}$. Specifically, for pixel $\mathbf{x}$ in I_t , depicted by an orange dot, we approximate a motion vector $V_{t \rightarrow 1}^{fw}(\mathbf{x})$ by scaling $V_{0 \rightarrow 1}(\mathbf{x})$ with a factor $(1-t)$, assuming that the motion vector field is locally smooth. Since the bilateral motion estimation is based on the assumption that a motion trajectory between consecutive frames is linear, two approximate motions $V_{t \rightarrow 1}^{fw}(\mathbf{x})$ and $V_{t \rightarrow 0}^{fw}(\mathbf{x})$ should be symmetric with respect to $\mathbf{x}$ in I_t . Thus, we obtain an additional approximate vector $V_{t \rightarrow 0}^{fw}(\mathbf{x})$ by reversing the direction of the vector $V_{t \rightarrow 1}^{fw}(\mathbf{x})$. In other words, we approximate the forward bilateral motions by

$$V_{t \rightarrow 1}^{fw}(\mathbf{x}) = (1-t) \times V_{0 \rightarrow 1}(\mathbf{x}), \quad (3)$$

$$V_{t \rightarrow 0}^{fw}(\mathbf{x}) = (-t) \times V_{0 \rightarrow 1}(\mathbf{x}). \quad (4)$$

Similarly, we approximate the backward bilateral motions by

$$V_{t \rightarrow 0}^{bw}(\mathbf{x}) = t \times V_{1 \rightarrow 0}(\mathbf{x}), \quad (5)$$

$$V_{t \rightarrow 1}^{bw}(\mathbf{x}) = -(1-t) \times V_{1 \rightarrow 0}(\mathbf{x}), \quad (6)$$

as illustrated in Fig. 5(c). Note that Jiang *et al.* [14] also used these equations (3)~(6), but derived only two motion candidates: $V_{t \rightarrow 1}(\mathbf{x})$ by combining (3) and (6) and $V_{t \rightarrow 0}(\mathbf{x})$ by combining (4) and (5). Thus, if an approximated motion in (3)~(6) is unreliable, the combined one is also degraded. In contrast, we use all four candidates in (3)~(6) directly to choose reliable motions in Section 3.3.

Fig. 4 shows that, whereas the bilateral motion estimation provides visual artifacts in (d), the motion approximation provides results without noticeable artifacts in (f). On the other hand, the bilateral motion estimation is more

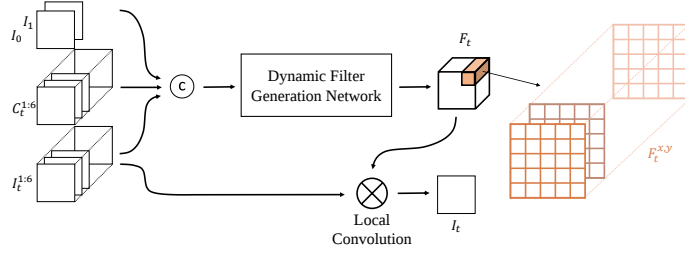


Fig. 6. Frame synthesis using dynamic local blending filters.

effective than the motion approximation in the cases of (e) and (g). Thus, the two schemes are complementary to each other.

3.3 Frame synthesis

We interpolate an intermediate frame by combining the six intermediate candidates, which are warped by the warping layers in Fig. 1. If we consider only color information, rich contextual information in the input frames may be lost during the synthesis [16, 20, 27], degrading the interpolation performance. Hence, as in [2, 3, 20], we further exploit contextual information in the input frames, called context maps. Specifically, we extract the output of the conv1 layer of ResNet-18 [8] as a context map, which is done by the context extractor in Fig. 1.

By warping the two input frames and the corresponding context maps, we obtain six pairs of a warped frame and its context map: two pairs are reconstructed using the bilateral motion estimation, and four pairs using the motion approximation. Fig. 1 shows these six pairs. Since these six warped pairs have different characteristics, they are used as complementary candidates of the intermediate frame. Recent video interpolation algorithms employ synthesis neural networks, which take warped frames as input and yield final interpolation results or residuals to refine pixel-wise blended results [2, 3, 20]. However, these synthesis networks may cause artifacts if motions are inaccurately estimated. To alleviate these artifacts, instead, we develop a dynamic filter network [13] that takes the aforementioned six pairs of candidates as input and outputs local blending filters, which are then used to process the warped frames to yield the intermediate frame. These local blending filters compensate for motion inaccuracies, by considering spatiotemporal neighboring pixels in the stack of warped frames. The frame synthesis layer performs this synthesis in Fig. 1.

Dynamic local blending filters: Fig. 6 shows the proposed synthesis network using dynamic local blending filters. The coefficients of the filters are learned from the images and contextual information through a dynamic blending filter network [13]. We employ the residual dense network [34] as the backbone for the filter generation. In Fig. 6, the generation network takes the input frames I_0 and I_1 and the intermediate candidates $\{I_t^{1:6}\}$ with the corresponding context maps

$\{C_t^{1:6}\}$ as input. Then, for each pixel $\mathbf{x} = (x, y)$, we generate six blending filters to fuse the six intermediate candidates, given by

$$F_t^{x,y} \in \mathbb{R}^{5 \times 5 \times 6}. \quad (7)$$

For each $\mathbf{x}$, the sum of all coefficients in the six filters are normalized to 1.

Then, the intermediate frame is synthesized via the dynamic local convolution. More specifically, the intermediate frame is obtained by filtering the intermediate candidates, given by

$$I_t(x, y) = \sum_{c=1}^6 \sum_{i=-2}^2 \sum_{j=-2}^2 F_t^{x,y}(i, j, c) I_t^c(x + i, y + j). \quad (8)$$

3.4 Training

The proposed algorithm includes two neural networks: the bilateral motion network and the dynamic filter generation network. We found that separate training of these two networks is more efficient than the end-to-end training in training time and memory space. Thus, we first train the bilateral motion network. Then, after fixing it, we train the dynamic filter generation network.

Bilateral motion network: To train the proposed bilateral motion network, we define the bilateral loss $\mathcal{L}_b$ as

$$\mathcal{L}_b = \mathcal{L}_p + \mathcal{L}_s \quad (9)$$

where $\mathcal{L}_p$ and $\mathcal{L}_s$ are the photometric loss [26, 33] and the smoothness loss [17].

For the photometric loss, we compute the sum of differences between a ground-truth frame I_t^l and two warped frames $I_{0 \rightarrow t}^l$ and $I_{1 \rightarrow t}^l$ using the bilateral motion fields $V_{t \rightarrow 0}^l$ and $V_{t \rightarrow 1}^l$, respectively, at all pyramid levels,

$$\mathcal{L}_p = \sum_{l=1}^L \alpha_l \left[\sum_{\mathbf{x}} \rho(I_{0 \rightarrow t}^l(\mathbf{x}) - I_t^l(\mathbf{x})) + \rho(I_{1 \rightarrow t}^l(\mathbf{x}) - I_t^l(\mathbf{x})) \right] \quad (10)$$

where $\rho(x) = \sqrt{x^2 + \epsilon^2}$ is the Charbonnier function [23]. The parameters α_l and ϵ are set to 0.01×2^l and 10^{-6} , respectively. Also, we compute the smoothness loss to constrain neighboring pixels to have similar motions, given by

$$\mathcal{L}_s = \|\nabla V_{t \rightarrow 0}\|_1 + \|\nabla V_{t \rightarrow 1}\|_1. \quad (11)$$

We use the Adam optimizer [15] with a learning rate of $\eta = 10^{-4}$ and shrink it via $\eta \leftarrow 0.5\eta$ at every 0.5M iterations. We use a batch size of 4 for 2.5M iterations and augment the training dataset by randomly cropping 256×256 patches with random flipping and rotations.

Dynamic filter generation network: We define the dynamic filter loss $\mathcal{L}_d$ as the Charbonnier loss between I_t and its synthesized version $\hat{I}_t$, given by

$$\mathcal{L}_d = \sum_{\mathbf{x}} \rho(\hat{I}_t(\mathbf{x}) - I_t(\mathbf{x})). \quad (12)$$

Similarly to the bilateral motion network, we use the Adam optimizer with $\eta = 10^{-4}$ and shrink it via $\eta \leftarrow 0.5\eta$ at 0.5M, 0.75M, and 1M iterations. We use a batch size of 4 for 1.25M iterations. Also, we use the same augmentation technique as that for the bilateral motion network.

Datasets: We use the Vimeo90K dataset [32] to train the proposed networks. The training set in Vimeo90K is composed of 51,312 triplets with a resolution of 448×256 . We train the bilateral motion network with $t = 0.5$ at the first 1M iterations and then with $t \in \{0, 0.5, 1\}$ for fine-tuning. Next, we train the dynamic filter generation network with $t = 0.5$. However, notice that both networks are capable of handling any $t \in (0, 1)$ using the bilateral cost volume in (2).

4 Experimental Results

We evaluate the performances of the proposed video interpolation algorithm on the Middlebury [1], Vimeo90K [32], UCF101 [28], and Adobe240-fps [29] datasets. We compare the proposed algorithm with state-of-the-art algorithms. Then, we conduct ablation studies to analyze the contributions of the proposed bilateral motion network and dynamic filter generation network.

4.1 Datasets

Middlebury: The Middlebury benchmark [1], the most commonly used benchmark for video interpolation, provides two sets: ‘Other’ and ‘Evaluation’. ‘Other’ contains the ground-truth for fine-tuning, while ‘Evaluation’ provides two frames selected from each of 8 sequences for evaluation.

Vimeo90K: The test set in Vimeo90K [32] contains 3,782 triplets of spatial resolution 256×448 . It is not used to train the model.

UCF101: The UCF101 dataset [28] contains human action videos of resolution 256×256 . Liu *et al.* [17] constructed the test set by selecting 379 triplets.

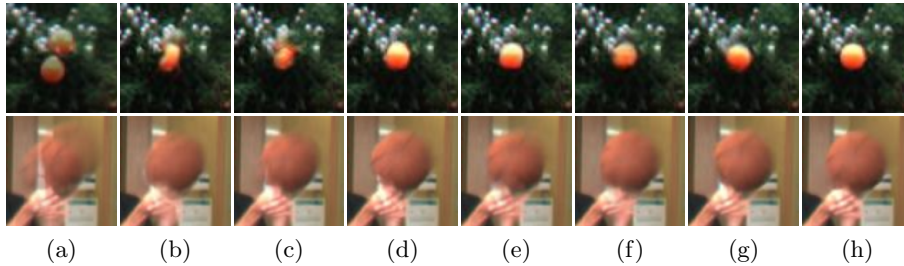
Adobe240-fps: Adobe240-fps [29] consists of high frame-rate videos. To assess the interpolation performance, we selected a test set of 254 sequences, each of which consists of nine frames.

4.2 Comparison with the state-of-the-arts

We assess the interpolation performances of the proposed algorithm in comparison with the conventional video interpolation algorithms: MIND [18], DVF [17],

Table 1. Quantitative comparisons on the Middlebury Evaluation set. For each metric, the numbers in **red** and **blue** denote the best and the second best results, respectively.

	[22]	[32]	[14]	[20]	[3]	[2]	Ours
Mequon	2.52	2.54	2.51	2.24	2.47	<u>2.38</u>	2.30
Schefflera	3.56	3.70	3.66	2.96	3.49	3.28	<u>3.07</u>
Urban	4.17	3.43	2.91	4.32	4.63	3.32	<u>3.17</u>
Teddy	5.41	5.05	5.05	4.21	4.94	4.65	<u>4.24</u>
Backyard	10.2	9.84	9.56	9.59	8.91	<u>7.88</u>	7.79
Basketball	5.47	5.34	5.37	5.22	<u>4.70</u>	4.73	4.08
Dumptruck	6.88	6.88	6.69	7.02	6.46	<u>6.36</u>	5.63
Evergreen	6.63	7.14	6.73	6.66	6.35	<u>6.26</u>	5.55
Average	5.61	5.49	5.31	5.28	5.24	<u>4.86</u>	4.48

**Fig. 7.** Visual comparison on the Middlebury Evaluation set. (a) Input, (b) SepConv- L_1 [22], (c) ToFlow [32], (d) SuperSloMo [14], (e) CtxSyn [20], (f) MEMC-Net* [3], (g) DAIN [2], and (h) BMBC (Ours).

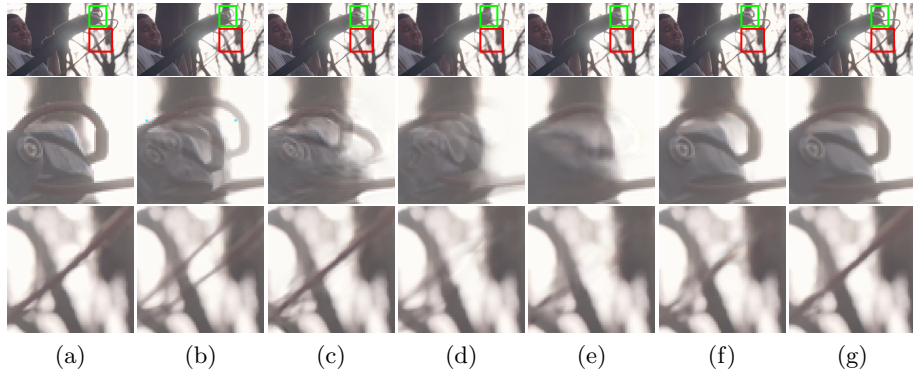
SpyNet [25], SepConv [22], CtxSyn [20], ToFlow [32], SuperSloMo [14], MEMC-Net [3], CyclicGen [16], and DAIN [2]. For SpyNet, we generated intermediate frames using the Baker *et al.*'s algorithm [1].

Table 1 shows the comparisons on the Middlebury Evaluation set [1], which are also available on the Middlebury website. We compare the average interpolation error (IE). A lower IE indicates better performance. The proposed algorithm outperforms all the state-of-the-art algorithms in terms of average IE score. Fig. 7 visually compares interpolation results. SepConv- L_1 [22], ToFlow [32], SuperSloMo [14], CtxSyn [20], MEMC-Net [3], and DAIN [2] yield blurring artifacts around the balls, losing texture details. On the contrary, the proposed algorithm reconstructs the clear shapes of the balls, preserving the details faithfully.

In Table 2, we provide quantitative comparisons on the UCF101 [28] and Vimeo90K [32] datasets. We compute the average PSNR and SSIM scores. The proposed algorithm outperforms the conventional algorithms by significant margins. Especially, the proposed algorithm provides 0.3 dB higher PSNR than DAIN [2] on Vimeo90K. Fig. 8 compares interpolation results qualitatively. Because the cable moves rapidly and the background branches make the motion estimation difficult, all the conventional algorithms fail to reconstruct the cable

Table 2. Quantitative comparisons on the UCF101 and Vimeo90K datasets.

	Runtime (seconds)	#Parameters (million)	UCF101 [17]		Vimeo90K [32]	
			PSNR	SSIM	PSNR	SSIM
SpyNet [25]	0.11	1.20	33.67	0.9633	31.95	0.9601
MIND [18]	0.01	7.60	33.93	0.9661	33.50	0.9429
DVF [17]	0.47	1.60	34.12	0.9631	31.54	0.9462
ToFlow [32]	0.43	1.07	34.58	0.9667	33.73	0.9682
SepConv- L_f [22]	0.20	21.6	34.69	0.9655	33.45	0.9674
SepConv- L_1 [22]	0.20	21.6	34.78	0.9669	33.79	0.9702
MEMC-Net [3]	0.12	70.3	34.96	0.9682	34.29	0.9739
CyclicGen [16]	0.09	3.04	35.11	0.9684	32.09	0.9490
CyclicGen_large [16]	-	19.8	34.69	0.9658	31.46	0.9395
DAIN [2]	0.13	24.0	34.99	0.9683	34.71	0.9756
BMBC (Ours)	0.77	11.0	35.15	0.9689	35.01	0.9764

**Fig. 8.** Visual comparison on the Vimeo90K test set. (a) Ground-truth, (b) ToFlow [32], (c) SepConv- L_f [22], (d) CyclicGen [16], (e) MEMC-Net* [3], (f) DAIN [2], and (g) BMBC (Ours).

properly. In contrast, the proposed algorithm faithfully interpolates the intermediate frame, providing fine details.

The proposed algorithm can interpolate an intermediate frame at any time instance $t \in (0, 1)$. To demonstrate this capability, we assess the $\times 2$, $\times 4$, and $\times 8$ frame interpolation performance on the Adobe240-fps dataset [29]. Because the conventional algorithms in Table 3, except for DAIN [2], can generate only intermediate frames at $t = 0.5$, we recursively apply those algorithms to interpolate intermediate frames at other t 's. Table 3 shows that the proposed algorithm outperforms all the state-of-the-art algorithms. As the frame rate increases, the performance gain of the proposed algorithm against conventional algorithms gets larger.

Table 3. Quantitative comparisons on the Adobe240-fps dataset for $\times 2$, $\times 4$, and $\times 8$ frame interpolation.

	$\times 2$		$\times 4$		$\times 8$	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ToFlow [32]	28.51	0.8731	29.20	0.8807	28.93	0.8812
SepConv- L_f [22]	29.14	0.8784	29.75	0.8907	30.07	0.8956
SepConv- L_1 [22]	29.31	0.8815	29.91	0.8935	30.23	0.8985
CyclicGen [16]	29.39	0.8787	29.72	0.8889	30.18	0.8972
CyclicGen_large [16]	28.90	0.8682	29.70	0.8866	30.24	0.8955
DAIN [2]	29.35	0.8820	29.73	0.8925	30.03	0.8983
BMBC (Ours)	29.49	0.8832	30.18	0.8964	30.60	0.9029

Table 4. PSNR comparison of combination of the intermediate candidates.

Intermediate candidates	UCF101 [28]	Vimeo90K [32]
	PSNR	PSNR
Appx4	34.99	34.72
BM	35.12	34.93
BM+Appx2	35.14	34.95
BM+Appx4	35.15	35.01

4.3 Model analysis

We conduct ablation studies to analyze the contributions of the three key components in the proposed algorithm: bilateral cost volume, intermediate motion approximation, and dynamic filter generation network. By comparing various combinations of intermediate candidates, we analyze the efficacy of the bilateral cost volume and the intermediate motion approximation jointly.

Intermediate candidates: To analyze the effectiveness of the bilateral motion estimation and the intermediate motion approximation, we train the proposed networks to synthesize intermediate frames using the following combinations:

- Appx4: Four intermediate candidates, obtained using approximated bilateral motions in (3)~(6), are combined.
- BM: Two intermediate candidates, obtained using bilateral motions, are combined.
- BM+Appx2: In addition to BM, two more candidates obtained using approximated bilateral motions $V_{t \rightarrow 0}^{fw}$ in (4) and $V_{t \rightarrow 1}^{bw}$ in (6) are used.
- BM+Appx4: Six intermediate candidates are used as well (proposed model).

Table 4 compares these models quantitatively. First, Appx4 shows the worst performance, while it is still comparable to the state-of-the-art algorithms. Second, BM provides better performance than Appx4 as well as the state-of-the-art algorithms, which confirms the superiority of the proposed BM to the approximation. Third, we can achieve even higher interpolation performance with more

Table 5. Analysis of the dynamic filter generation network. In all settings, the six warped frames are input to the network.

Kernel size	Input to filter generation network		UCF101 [28]	Vimeo90K [32]
	Input frames	Context maps	PSNR	PSNR
5×5			34.98	34.81
3×3	✓		35.09	34.90
5×5	✓		35.08	34.96
7×7	✓		35.02	34.98
5×5	✓	✓	35.15	35.01

intermediate candidates obtained through the motion approximation.

Dynamic blending filters: We analyze the optimal kernel size and the input to the dynamic filter generation network. Table 5 compares the PSNR performances of different settings. First, the kernel size has insignificant impacts, although the computational complexity is proportional to the kernel size. Next, when additional information (input frames and context maps) is fed into the dynamic filter generation network, the interpolation performance is improved. More specifically, using input frames improves PSNRs by 0.10 and 0.15 dB on the UCF101 and Vimeo90K datasets. Also, using context maps further improves the performances by 0.07 and 0.05dB. This is because the input frames and context maps help restore geometric structure and exploit rich contextual information.

5 Conclusions

We developed a deep-learning-based video interpolation algorithm based on the bilateral motion estimation, which consists of the bilateral motion network and the dynamic filter generation network. In the bilateral motion network, we developed the bilateral cost volume to estimate accurate bilateral motions. In the dynamic filter generation network, we warped the two input frames using the estimated bilateral motions and fed them to learn filter coefficients. Finally, we synthesized the intermediate frame by superposing the warped frames with the generated blending filters. Experimental results showed that the proposed algorithm outperforms the state-of-the-art video interpolation algorithms on four benchmark datasets.

Acknowledgements

This work was supported in part by the Agency for Defense Development (ADD) and Defense Acquisition Program Administration (DAPA) of Korea under grant UC160016FD and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIP) (No. NRF-2018R1A2B3003896 and No. NRF-2019R1A2C4069806).

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.* **92**(1), 1–31 (Mar 2011), <http://vision.middlebury.edu/flow/eval/> 10, 11
2. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: *Proc. IEEE CVPR*. pp. 3703–3712 (Jun 2019) 1, 2, 3, 6, 8, 11, 12, 13
3. Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE Trans. Pattern Anal. Mach. Intell.* (Sep 2019) 1, 2, 3, 8, 11, 12
4. Choi, B.D., Han, J.W., Kim, C.S., Ko, S.J.: Motion-compensated frame interpolation using bilateral motion estimation and adaptive overlapped block motion compensation. *IEEE Trans. Circuits Syst. Video Technol.* **17**(4), 407–416 (Apr 2007) 1, 2
5. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: *Proc. IEEE ICCV*. pp. 2758–2766 (Dec 2015) 1, 3, 5, 6
6. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: DeepStereo: Learning to predict new views from the world’s imagery. In: *Proc. IEEE CVPR*. pp. 5515–5524 (Jun 2016) 1
7. G. Lu, X. Zhang, L.C., Gao, Z.: Novel integration of frame rate up conversion and hevc coding based on rate-distortion optimization. *IEEE Trans. Image Process.* **27**(2), 678–691 (Feb 2018) 1
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proc. IEEE CVPR*. pp. 770–778 (Jun 2016) 8
9. Hosni, A., Rhemann, C., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(2), 504–511 (Feb 2013) 5
10. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of optical flow estimation with deep networks. In: *Proc. IEEE CVPR*. pp. 2462–2470 (Jul 2017) 1, 3, 6
11. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: *Proc. NIPS*. pp. 2017–2025 (2015) 4
12. Jeong, S.G., Lee, C., Kim, C.S.: Motion-compensated frame interpolation based on multihypothesis motion estimation and texture optimization. *IEEE Trans. Image Process.* **22**(11), 4497–4509 (Nov 2013) 1
13. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. *NIPS* pp. 667–675 (2019) 8
14. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In: *Proc. IEEE CVPR*. pp. 9000–9008 (Jun 2018) 1, 2, 3, 6, 7, 11
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: *Proc. ICLR* (May 2015) 9
16. Liu, Y.L., Liao, Y.T., Lin, Y.Y., Chuang, Y.Y.: Deep video frame interpolation using cyclic frame generation. In: *Proc. AAAI* (Jan 2019) 1, 2, 8, 11, 12, 13
17. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: *Proc. IEEE ICCV*. pp. 4463–4471 (Oct 2017) 1, 9, 10, 12

18. Long, G., Kneip, L., Alvarez, J.M., Li, H., Zhang, X., Yu, Q.: Learning image matching by simply watching video. In: Proc. ECCV. pp. 434–450 (Oct 2016) [1](#), [2](#), [10](#), [12](#)
19. Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: PhaseNet for video frame interpolation. In: Proc. IEEE CVPR. pp. 498–507 (Jun 2018) [2](#)
20. Niklaus, S., Liu, F.: Context-aware synthesis for video frame interpolation. In: Proc. IEEE CVPR. pp. 1701–1710 (Jun 2018) [1](#), [2](#), [8](#), [11](#)
21. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: Proc. IEEE CVPR. pp. 670–679 (Jul 2017) [1](#), [2](#)
22. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proc. IEEE ICCV. pp. 261–270 (Oct 2017) [1](#), [2](#), [11](#), [12](#), [13](#)
23. P. Charbonnier, L. Blanc-Feraud, G.A., Barlaud, M.: Two deterministic half-quadratic regularization algorithms for computed imaging. In: Proc. IEEE ICIP. pp. 168–172 (Nov 1994) [9](#)
24. Qifeng Chen, V.K.: Full flow: Optical flow estimation by global optimization over regular grids. In: Proc. IEEE CVPR. pp. 4706–4714 (Jun 2016) [6](#)
25. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: Proc. IEEE CVPR. pp. 4161–4170 (Jul 2017) [1](#), [11](#), [12](#)
26. Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., Zha, H.: Unsupervised deep learning for optical flow estimation. In: Proc. AAAI (Feb 2017) [9](#)
27. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Proc. Int. Conf. Medical Image Comput. Comput.-Assisted Intervention. pp. 234–241 (Oct 2015) [8](#)
28. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012) [10](#), [11](#), [13](#), [14](#)
29. Su, S., Delbracio, M., Wang, J., Sapiro, G., Heidrich, W., Wang, O.: Deep video deblurring for hand-held cameras. In: Proc. IEEE CVPR. pp. 1279–1288 (Jun 2017) [10](#), [12](#)
30. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: Proc. IEEE CVPR. pp. 8934–8943 (Jun 2018) [1](#), [3](#), [4](#), [5](#), [6](#)
31. Xu, J., Ranftl, R., Koltun, V.: Accurate optical flow via direct cost volume processing. In: Proc. IEEE CVPR. pp. 1289–1297 (Jun 2017) [3](#), [5](#), [6](#)
32. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video enhancement with task-oriented flow. IJCV **127**(8), 1106–1125 (Feb 2019) [1](#), [2](#), [10](#), [11](#), [12](#), [13](#), [14](#)
33. Yu, J., Harley, A., Derpanis, K.: Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In: Proc. ECCV. pp. 3–10 (Oct 2016) [9](#)
34. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: Proc. IEEE CVPR. pp. 2472–2481 (Jun 2018) [8](#)