Multi-Scale Memory-Based Video Deblurring: Supplementary Material

Bo Ji Angela Yao
National University of Singapore
{jibo,ayao}@comp.nus.edu.sg

1. Loss

We borrow the multi-scale content loss [4] to train our model, but replace the mean-square error with Charbonnier loss [1]:

$$\mathcal{L} = \frac{1}{S \cdot N} \sum_{s=1}^{S} \sum_{i=0}^{N} \frac{1}{M^s} \rho(R_i^s - Y_i^s), \tag{1}$$

where $\rho(x) = \sqrt{x^2 + \epsilon^2}, \epsilon = 10^{-3}$, R_i^s and Y_i^s denote the output and the ground-truth sharp frame at scale level s respectively, S and N denote the number of scales and training sequence and M^s is the number of pixels in the frame at scale level s.

2. Architecture

We provide the detailed architecture for each module in our proposal. The input image is of shape $H \times W \times 3$.

Downsampling module. The downsampling module $\mathcal{D}$ consists of one initial convolutional layer, residual dense blocks [8] and two strided convolution. Table 1 shows the architecture of $\mathcal{D}$.

Feature extraction module. Table 2 shows the architecture of the feature extraction module $\mathcal{F}$. The module starts with the initial convolutional layer and then feeds the features into multiple residual blocks [3]. To save the computational cost, we use 30 and 10 blocks for the forward and backward modules, respectively.

In the multi-scale design, $\mathcal{F}$ has different input channels at three scales, but the first convolutional layer maps the features into the same output channel. Since we start from the scale level $s = 3$, the input does not have access to the features from the previous scale and the input channel is fewer than those at higher scale levels.

Upsampling module. The upsampling module $\mathcal{U}$ contains two transposed convolution and one 5×4 convolutional layer. Each transposed convolution upscales the feature by a scale factor of 2. The architecture is presented in Table 3.

Key encoder. The key encoder $\mathcal{K}$ contains two convolutional layer, two residual blocks, and the first stage of the ResNet50 [2]. In Table 4, the *Conv2*, *Pooling* and *Stage1* rows represent the first stage of a pre-trained ResNet50. The output of ResNet50 is projected into the key space via a convolutional layer, where the dimension is reduced from 256 to 64 to save the computational budget for the matching.

Value encoder. The value encoder module $\mathcal{V}$ is similar to the key encoder $\mathcal{K}$. One difference is that the initial output channel is set to 32 rather than 48. The other difference is that we do not project the encoded value to another dimension as we do not calculate attention using the values. Table 5 shows the value encoder structure.

Decoder. Table 6 illustrates the architecture of the decoder $\mathcal{G}$. The pixel shuffle layer [6] upscales the features by a scale factor of 4.

3. Experimental Results

In this section, we provide additional experimental results. In Fig. 1, we provide the visual samples for the entire 6 consecutive frames to show the effectiveness of our model on handling large motion. We provide more visual comparisons from Fig. 2 to 5. For the video samples, please refer to the provided mp4 files.

In Fig. 6, we also show some examples in which our method fails to generate sharp results. The patterns in the region denoted by the red boxes are very similar, which leads to the inaccurate matching performance of the similarity matching. This problem exists not only for our method, but also for other alignment methods such as optical flow, as it causes some misalignment between various blurry objects.

Layer	Output	Downsampling Module
Conv1	$H \times W \times 3$	5×5, stride 1
RDB1	$H \times W \times 3$	$[3 \times 3, 16$, dense conv$] \times 3$
Conv2	$H/2 \times W/2 \times 12$	5×5, stride 2
RDB2	$H/2 \times W/2 \times 12$	$[3 \times 3, 24$, dense conv$] \times 3$
Conv3	$H/4 \times W/4 \times 48$	5×5, stride 2

Table 1. Downsampling module $\mathcal{D}$ architecture.

Layer	Output	Forward Module	Backward Module
Conv1	$H/4 \times W/4 \times 64$	3×3, stride 1	
ResBlock1	$H/4 \times W/4 \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 30$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 10$

Table 2. Feature extraction module $\mathcal{F}$ architecture.

Layer	Output	Upsampling Module
Transposed conv1	$H/2 \times W/2 \times 32$	3×3, stride 2
Transposed conv2	$H \times W \times 16$	3×3, stride 2
Conv1	$H \times W \times 3$	5×5, stride 1

Table 3. Upsampling module $\mathcal{U}$ architecture.

Layer	Output	Key Encoder
Conv1	$H/4 \times W/4 \times 48$	3×3, stride 1
ResBlock1	$H/4 \times W/4 \times 48$	$\begin{bmatrix} 3 \times 3, 48 \\ 3 \times 3, 48 \end{bmatrix} \times 2$
Conv2	$H/8 \times W/8 \times 64$	7×7, stride 2
Pooling	$H/16 \times W/16 \times 64$	3×3, max pool, stride 2
Stage1	$H/16 \times W/16 \times 256$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3	$H/16 \times W/16 \times 64$	3×3, stride 1

Table 4. Key encoder module $\mathcal{K}$ architecture.

Layer	Output	Value Encoder
Conv1	$H/4 \times W/4 \times 32$	3×3, stride 1
ResBlock1	$H/4 \times W/4 \times 32$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 2$
Conv2	$H/8 \times W/8 \times 64$	7×7, stride 2
Pooling	$H/16 \times W/16 \times 64$	3×3, max pool, stride 2
Stage1	$H/16 \times W/16 \times 256$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$

Table 5. Value encoder module $\mathcal{V}$ architecture.

Layer	Output	Decoder
Conv1	$H/16 \times W/16 \times 64$	3×3, stride 1
ResBlock1	$H/16 \times W/16 \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
PixelShuffle	$H/4 \times W/4 \times 64$	3×3, stride 1

Table 6. Decoder module $\mathcal{G}$ architecture.

References

[1] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172. IEEE, 1994. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[3] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 1

[4] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891, 2017.

(a) Input

(b) DBN [7]

(c) CDVD-TSP [5]

(d) Ours

(e) Ground-truth

Figure 1. Qualitative comparisons on the extremely difficult frames of the original GOPRO dataset [4]. We present the entire 6 consecutive frames for comparison. For these 6 images, only our results can be seen to have roughly 5 characters.

1, 3, 4, 5, 6, 7

[5] Jinshan Pan, Haoran Bai, and Jinhui Tang. Cascaded deep video deblurring using temporal sharpness prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3043–3051, 2020. 3, 4, 5, 6, 7

[6] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 1

[7] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1279–1288, 2017. 3, 4, 5, 6, 7

[8] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. 1

[9] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020. 4, 5, 6, 7

(a) Input (b) DBN [7]

(c) ESTRNN [9] (d) CDVD-TSP [5]

(e) Ours (f) Ground-truth

Figure 2. Qualitative comparisons on the original GOPRO dataset [4]. The white characters 'RA', the yellow marker and the characters 'BA' on the black board are slightly visible with our method.

(a) Input (b) DBN [7]

(c) ESTRNN [9] (d) CDVD-TSP [5]

(e) Ours (f) Ground-truth

Figure 3. Qualitative comparisons on the original GOPRO dataset [4]. Our method is able to handle very detailed areas, such as tree branches.

(a) Input

(b) DBN [7]

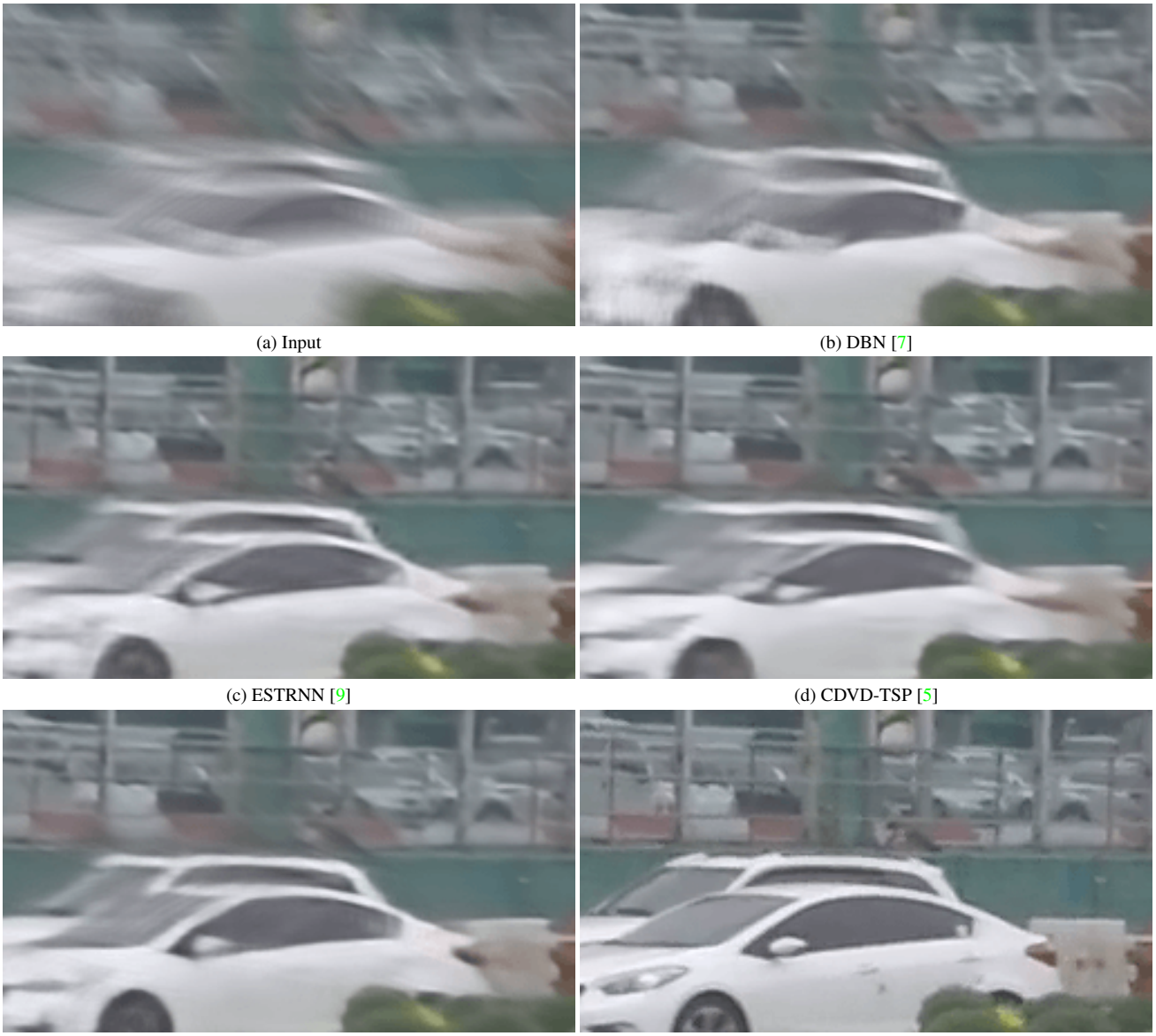
(c) ESTRNN [9]

(d) CDVD-TSP [5]

(e) Ours

(f) Ground-truth

Figure 4. Qualitative comparisons on the original GOPRO dataset [4]. Our method does not mix the characters together when restoring a text sequence, such as the number “182” in the figure. In our method, there is a spacing between the number “2” and the number “8”. In other methods, the three numbers are stuck together.

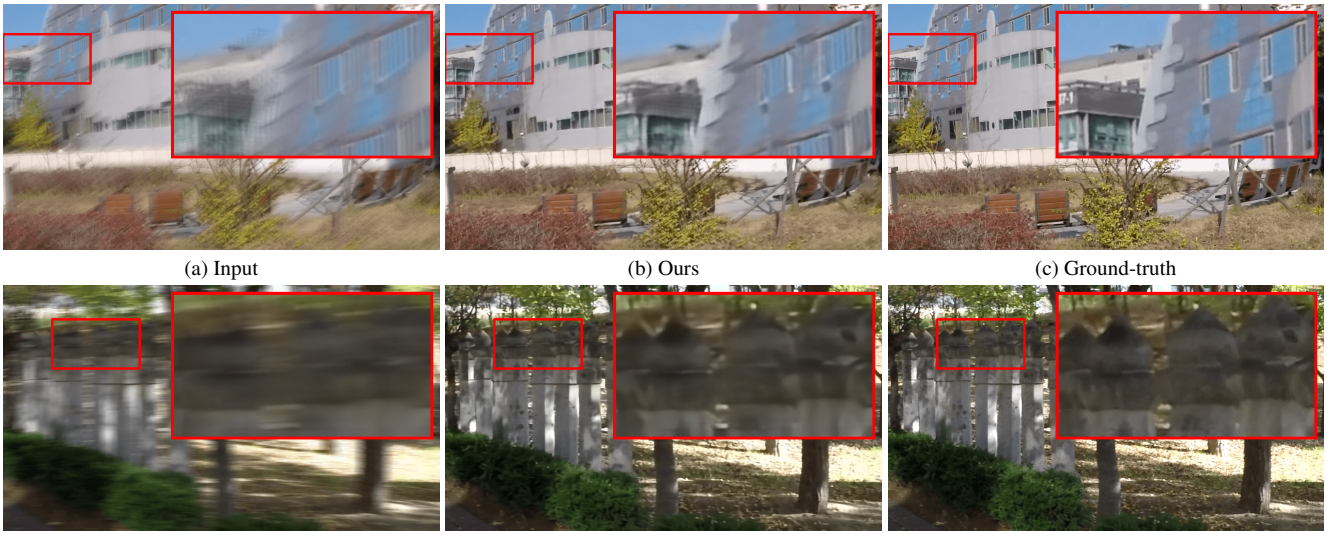


(a) Input (b) DBN [7]

(c) ESTRNN [9] (d) CDVD-TSP [5]

(e) Ours (f) Ground-truth

Figure 5. Qualitative comparisons on the original GOPRO dataset [4]. Our method does not have a dragging effect when dealing with large motion, *e.g*. the front and back part of the car in white.



(a) Input (b) Ours (c) Ground-truth

(d) Input (e) Ours (f) Ground-truth

Figure 6. Hard samples. The zoomed in regions do not have sharp edges and details due to the limitation of the similarity matching.