Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Supplementary Material for Real-Time Video Deblurring via Lightweight Motion Compensation

Hyeongseok Son[†]*, Junyong Lee[†], Sunghyun Cho, and Seungyong Lee[‡]

POSTECH

1. Analysis on Motion Compensation Network

1.1. Implementation Detail of Motion Compensation Module

The motion compensation network estimates motion between the features $\hat{f}_t^n$ and $\hat{f}_{t-1}^n$, which are the modulated structure features resulting from a motion layer. Inspired by the correlation layer of FlowNet [DFI*15], the network first constructs a matching cost volume based on the cosine similarity between the features. Specifically, given structure-injected features $\hat{f}_t^n$ and $\hat{f}_{t-1}^n$, the cosine similarity between the two features at x_1 in $\hat{f}_t^n$ and x_2 in $\hat{f}_{t-1}^n$ is defined as:

$$c^n(x_1,x_2) = \langle \hat{f}_t^n(x_1), \hat{f}_{t-1}^n(x_2) \rangle. \quad (1)$$

For computational efficiency, we only compute a partial cost volume with a limited search window. Given a maximum displacement D between two spatial locations x_1 and x_2, we compute a matching cost volume $C^n : \mathbb{R}^{w\times h\times(2D+1)^2}$, where $(2D+1)^2$ represents the number of correlations stacked along the channel dimension for every spatial location. Specifically, C^n is defined as:

$$C^n = \{c^n(x, x+d) | x \in [1,w]\times[1,h], d \in [-D,D]\times[-D,D]\}. \quad (2)$$

Then, we apply *argmax* to C^n along the channel dimension to find the best matches to convert them into 2D displacement map W^n as done in [JSZk15]. Finally, we warp f_{t-1}^N and obtain a motion-compensated feature map $\tilde{f}_{t-1}^{N,n} = f_{t-1}^N(x+W^n)$.

The computational cost of computing C^n with $D = 10$ in our implementation is equivalent to that of a single convolution layer with 21×21 kernels, while it does not include any learnable parameters.

1.2. Effect of Maximum Displacement D

The maximum displacement D determines the search window size for our motion compensation network when feature matching is applied between temporal features (Sec. 1.1). Consequently, a larger D leads the network to cover a large motion presented between frames, but it also increases the running time of the model. Table 1 compares the performances with different values of D, where both deblurring quality and computational time increases with D. The results show that, for models with $D > 10$, improvements in deblurring quality are marginal while increases in computational time are significant. We choose $D = 10$ for our final models, which show much better deblurring quality compared to the model with $D = 5$, while the running time of motion compensation is still fast.

Datasets	PSNR/SSIM of our 2-stacked model with different D			
	$D=5$	10	20	30
DVD [SDW*17]	30.22/0.921	30.33/0.923	30.33/0.923	30.32/0.922
GoPro [NKL17]	28.59/0.909	28.74/0.911	28.76/0.911	28.77/0.910
Total time (ms)	45	53	110	222
MC time (ms)	5	15	55	131

Table 1: *Effect of maximum displacement D. We compare models[1] trained with different D, and measure deblurring accuracy on the DVD and GoPro datasets. MC time indicates the running time taken by motion compensation networks $\mathcal{M}^1$ and $\mathcal{M}^2$.*

		2-stack	4-stack	10-stack
w/o skipping	PSNR/SSIM	30.33/0.923	30.76/0.928	31.09/0.933
	Total time (ms)	68	119	290
	MC time (ms)	30	60	152
w/ skipping	PSNR/SSIM	30.33/0.923	30.73/0.929	31.07/0.933
	Total time (ms)	54	84	171
	MC time (ms)	15	16	19

Table 2: *Effect of feature matching skipping. We measure deblurring accuracy (PSNR/SSIM) on the DVD dataset [SDW*17] for n-stacked models[†] with and without feature matching skipping. MC time indicates the running time of motion compensation networks.*

1.3. Faster Motion Compensation with Feature Matching Skipping

Although the computational overhead of our motion compensation network is small, it can be a burden if we stack a number of multi-task units. We can effectively resolve this problem by skipping the

[†] Equal contributions
[‡] Corresponding author
* Currently working at Samsung Advanced Institute of Technology

[1] Note that the models are trained for 300K iterations.

feature matching operation in some stacks (*e.g.*, $n > 1$). Specifically, we may use the motion pre-computed in an early stack for later stacks. As the feature matching operation occupies most computations for motion estimation, we can save quite a large computations by skipping the operation. Moreover, reusing the pre-computed motion still effectively boosts the deblurring quality, as our motion compensation network can produce moderate motion estimation results from early stacks (Figs 6e and 6f in the main paper).

Table 2 validates the effect of skipping feature matching on models stacked with different numbers of multi-task units. Models without the skipping scheme perform correlation-based feature matching for every stack, while models with the skipping scheme reuse the motion computed by the feature matching operation in the first stack for the rest of the stacks. As shown in the table, compared to the models that re-compute a motion in every stack, the models skipping feature matching operations show slightly lower deblurring quality but record significantly reduced running time.

1.4. Deeper Analysis on Structure Injection

In our structure injection scheme, instead of using only the simple addition of detail and structural features, we further process the features with the motion layer for a more effective fusion of those features. A question may naturally follow whether the motion layer is an essential component in the structure injection scheme. To answer the question, we conduct an additional ablation study on our final model with components comprising the structure injection scheme, the addition operation between the detail-level and structural feature maps, and the motion layer that combines the feature maps (Table 4). Note that, in the table, the model in the last row is the same final model used in the ablation study (Sec. 4.2.2) of the main paper.

In Table 4, using the motion layer without the addition operation of structural features (the second row of the table) does not have any effect other than slightly increasing the model size. On the other hand, the addition of detail and structural features (the third row) brings a significant increase in the deblurring quality while retaining the motion compensation quality. When the motion layer is attached (the last row), the deblurring quality further increases, thanks to the motion layer making the multi-task detail network more effectively focus on learning detail features by fully injecting the structural information into detail features.

2. More Results

Comparison on the REDS dataset In the main paper, we quantitatively compared our models with previous ones on the DVD dataset [SDW*17] and GoPro dataset [NKL17]. In this section, we provide an additional comparison on the REDS dataset [NBH*19]. Specifically, we compare our n-stacked models with previous video deblurring methods EST [ZYZB20] and PVD [SLL*21] trained with the REDS dataset. For the previous models, as the authors do not provide the model trained on the dataset, we trained the models with the code provided by the authors to reproduce the same quantitative results reported by the authors. For our models, we trained

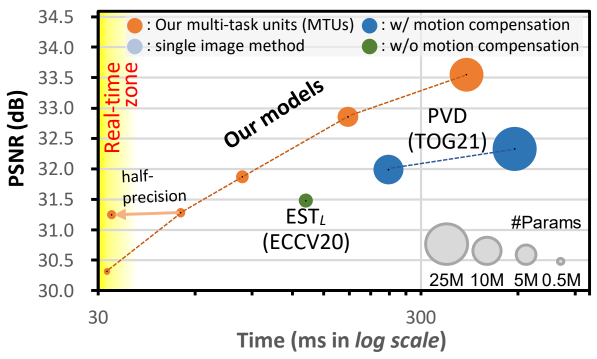


Figure 1: *Comparison on deblurring efficiency. Our models are indicated in orange circles, each of which is stacked with a different number of multi-task units.*

each model using the training set provided in the dataset, with the same training strategy described in Sec. 4.1 in the main paper.

Table 3 shows quantitative results. Note that EST_L denotes the model with larger parameters compared to the EST model used in the comparisons on the DVD dataset [SDW*17] and GoPro dataset [NKL17] (Tables 3 and 4 in the main paper, respectively). As the table indicates, compared to EST_L, our 4-stacked model shows better deblurring quality with a smaller model size and faster computation time. Compared to PVD, our 4-stacked model reports comparable deblurring quality even with much smaller model size and faster running time. Our 10-stacked model and the larger model (Ours_L^{10}) outperform PVD_L by a large margin but still have a smaller model size and faster inference time, validating the effectiveness of our approach.

Fig. 1 visualizes Table 3. The diagram shows the similar tendency to the cases trained with the DVD [SDW*17] and GoPro [NKL17] datasets (Figs. 1a and 1b of the main paper), where each of our model variants shows a much faster running time compared to previous methods with similar deblurring quality.

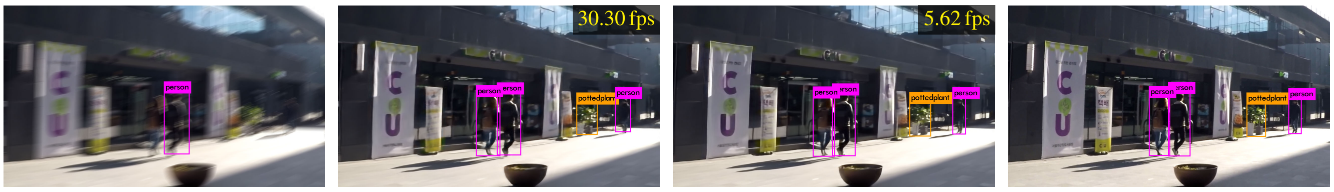
Application: Object Detection Thanks to the flexibility of our architecture, our network can cover from environments where high deblurring quality is desired to environments demanding for low-computation power. For the latter case, our lightweight real-time model can be leveraged to pre-process videos to improve higher-level vision tasks such as object detection for autonomous driving, where real-time processing is highly demanded.

Fig. 2 qualitatively shows the object detection results, for which we used our 2- and 10-stack models to deblur video frames of the GoPro dataset and applied object detection [RF18] to the deblurred frames. Although the 10-stack model showed better deblurring results compared to the 2-stack model, the 2-stack model is enough to improve the object detection quality, despite its real-time speed.

Additional Qualitative Results In the main paper, we qualitatively compared our models with previous methods on the DVD dataset [SDW*17], GoPro dataset [NKL17], and real-world blurred videos [CWL12]. In this section, we provide qualitative results on the REDS dataset [NBH*19] (Figs. 3 and 4). We also addition-

	EST_L🔵 [ZYZB20]	PVD🔵 [SLL*21]	PVD_L🔴 [SLL*21]	Ours^n (n: # of stacks)				Ours^n_{hp}	Ours^n_L
				$n=1$	2	4🔵	10🔴	2	10🔴
PSNR	31.48	31.99	32.33	30.32	31.28	31.87	32.86	31.25	**33.55**
SSIM	0.922	0.927	0.932	0.901	0.917	0.926	0.940	0.916	**0.948**
Params (M)	2.47	10.51	23.36	**0.54**	1.04	2.04	5.03	1.04	13.80
Time (ms)	132	238	585	**33**	53	86	171	**33**	416

Table 3: *Quantitative evaluation on the REDS dataset [NBH*19]. The colored dots denote models showing similar deblurring qualities in PSNR, where each of our models yields an upper bound PSNR at a much faster running time.*



(a) I^b_t (b) Ours^2_{hp} (c) Ours^{10} (d) I^{GT}_t

Figure 2: *Object detection results. Deblurred results of our (a) 2-stack and (b) 10-stack models improve the performance of the object detection task. Our real-time 2-stack model shows a comparable detection quality to that of the 10-stack model and can be useful when real-time processing is required (e.g., object detection for autonomous driving).*

Structure injection		Deblurring quality		Motion compensation accuracy	
addition	motion layer	PSNR	SSIM	PSNR	SSIM
		29.84	0.915	27.24	0.862
	✓	29.84	0.916	27.24	0.862
✓		30.26	0.921	26.20	0.891
✓	✓	30.30	0.923	27.25	0.864

Table 4: *Additional ablation study on our model with structure injection. Our structure injection consists of an addition operation of detail/structure features, and motion layer. The model in the last row is our final model, the same final model reported in the last row of Table 2 in the main paper.*

ally show qualitative results on the DVD dataset (Fig. 5), GoPro dataset [NKL17] (Fig. 6), and real-world blurred videos (Fig. 7).

3. Detailed Network Architecture

Table 5 shows our network architecture in detail. Each multi-task unit (**MTU** in the table) consists of three main components: *multi-task detail network* $\mathcal{F}^n$, *deblurring network* $\mathcal{D}^n$, and *motion compensation network* $\mathcal{M}^n$, where n is the index of a multi-task unit.

The multi-task detail network $\mathcal{F}^n$ is a lightweight encoder-decoder network based on the U-Net architecture [RFB15]. The encoder is composed of a convolution layer followed by two down-sampling convolution layers with stride two. The decoder has two up-sampling deconvolution layers followed by a convolution layer. The encoder and decoder are connected by skip-connections at the same levels. Between the encoder and decoder, the network has four residual blocks. The deblurring network $\mathcal{D}^n$ consists of a single convolution layer dubbed as a *deblur layer* followed by an element-wise summation operator connected with a long skip connection carrying I^b_t for the residual detail learning. The motion compensation network $\mathcal{M}^n$ consists of a single convolution layer dubbed as a *motion layer* followed by a motion compensation module having no learnable parameters.

References

[CWL12] Cho S., Wang J., Lee S.: Video deblurring for hand-held cameras using patch-based synthesis. *ACM Transactions on Graphics 31*, 4 (2012), 64:1–64:9. 2, 8

[DFI*15] Dosovitskiy A., Fischer P., Ilg E., Häusser P., Hazirbaş C., Golkov V., v.d. Smagt P., Cremers D., Brox T.: FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV* (2015). 1

[JSZk15] Jaderberg M., Simonyan K., Zisserman A., kavukcuoglu k.: Spatial transformer networks. In *Proc. NeurIPS* (2015). 1

[NBH*19] Nah S., Baik S., Hong S., Moon G., Son S., Timofte R., Lee K. M.: Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *Proc. CVPRW* (2019). 2, 3, 5

[NKL17] Nah S., Kim T. H., Lee K. M.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. CVPR* (2017). 1, 2, 3, 7

[NSL19] Nah S., Son S., Lee K. M.: Recurrent neural networks with intra-frame iterations for video deblurring. In *Proc. CVPR* (2019). 6, 7, 8

[PBT20] Pan J., Bai H., Tang J.: Cascaded deep video deblurring using temporal sharpness prior. In *Proc. CVPR* (2020). 6, 7, 8

[RF18] Redmon J., Farhadi A.: Yolov3: An incremental improvement. In *ArXiv* (2018). 2

[RFB15] Ronneberger O., Fischer P., Brox T.: U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI* (2015). 3

[SDW*17] Su S., Delbracio M., Wang J., Sapiro G., Heidrich W., Wang O.: Deep video deblurring for hand-held cameras. In *Proc. CVPR* (2017). 1, 2, 6, 8

[SLL*21] Son H., Lee J., Lee J., Cho S., Lee S.: Recurrent video deblurring with blur-invariant motion estimation and pixel volumes. *ACM Transactions on Graphics 40*, 5 (2021). 2, 3, 5, 6, 7, 8

[TGS*18] TAO X., GAO H., SHEN X., WANG J., JIA J.: Scale-recurrent network for deep image deblurring. In *Proc. CVPR* (2018). 6

[ZYZB20] ZHONG Z., YE G., ZHENG Y., BO Z.: Efficient spatio-temporal recurrent neural network for video deblurring. In *Proc. ECCV* (2020). 2, 3, 5, 6, 7, 8

[ZZP*19] ZHOU S., ZHANG J., PAN J., XIE H., ZUO W., REN J.: Spatio-temporal filter adaptive network for video deblurring. In *Proc. ICCV* (2019). 6, 8

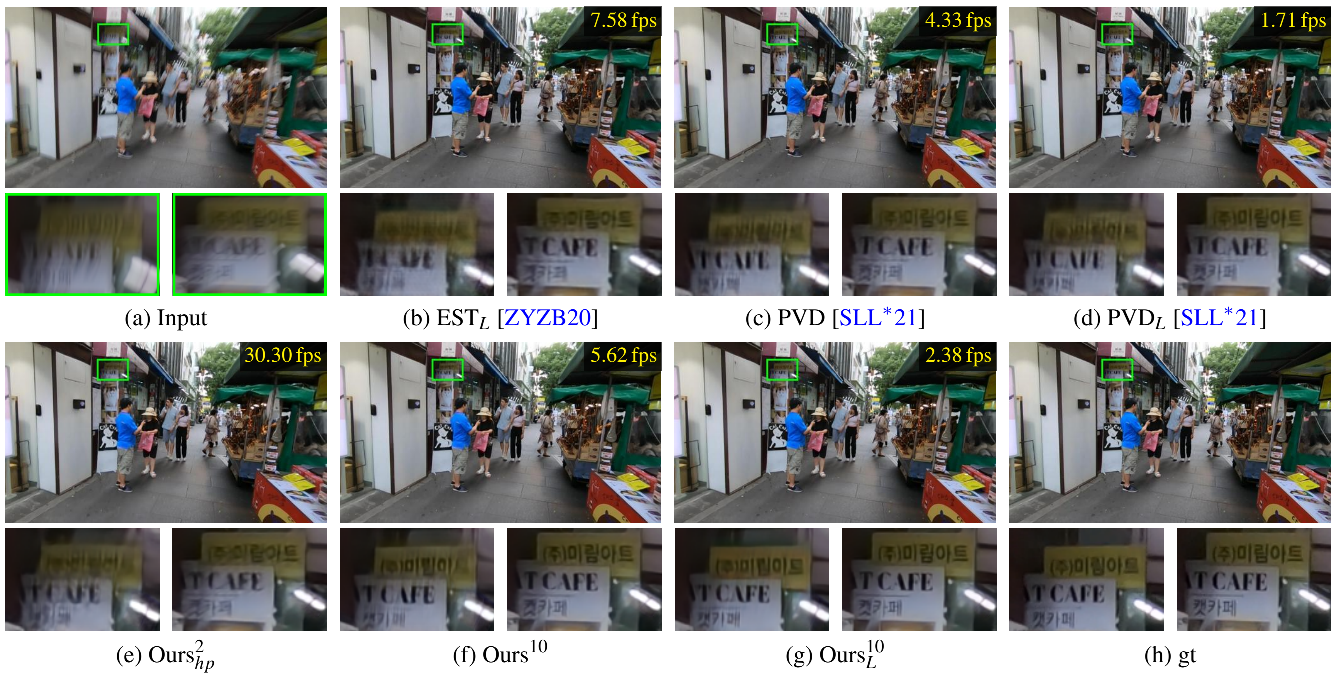


Figure 3: *Qualitative comparison on the REDS dataset [NBH*19]. Cropped images visualize regions in the green box of two consecutive frames.*

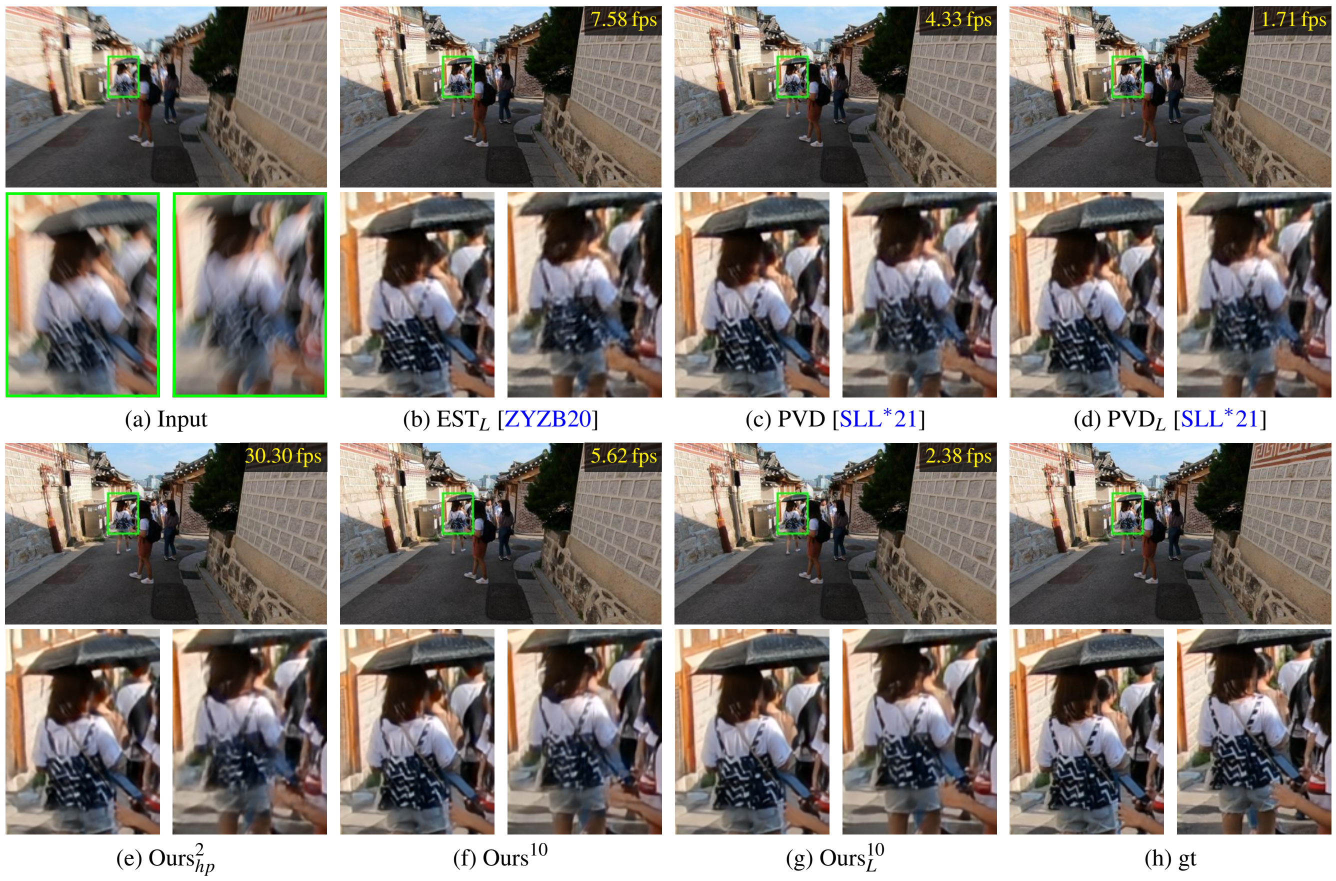


Figure 4: *Additional qualitative comparison on the REDS dataset [NBH*19].*

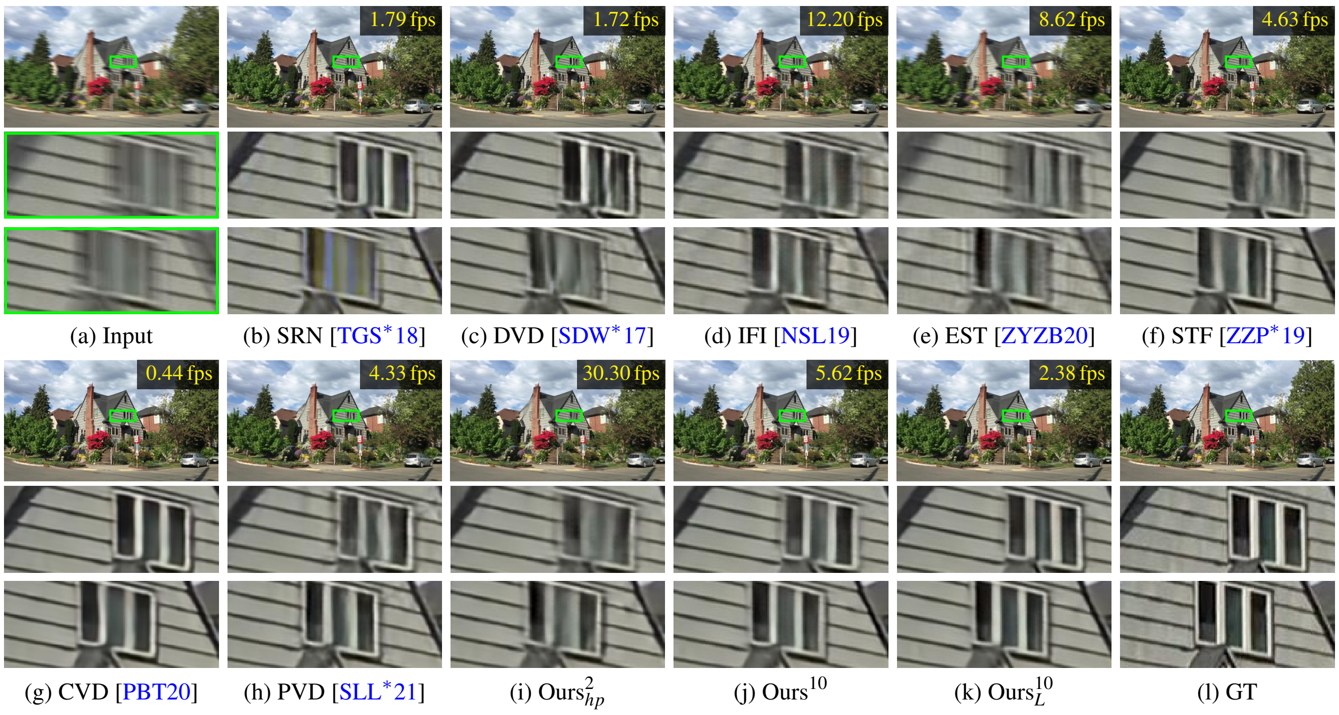


Figure 5: *Additional qualitative comparison on the DVD dataset [SDW*17].*

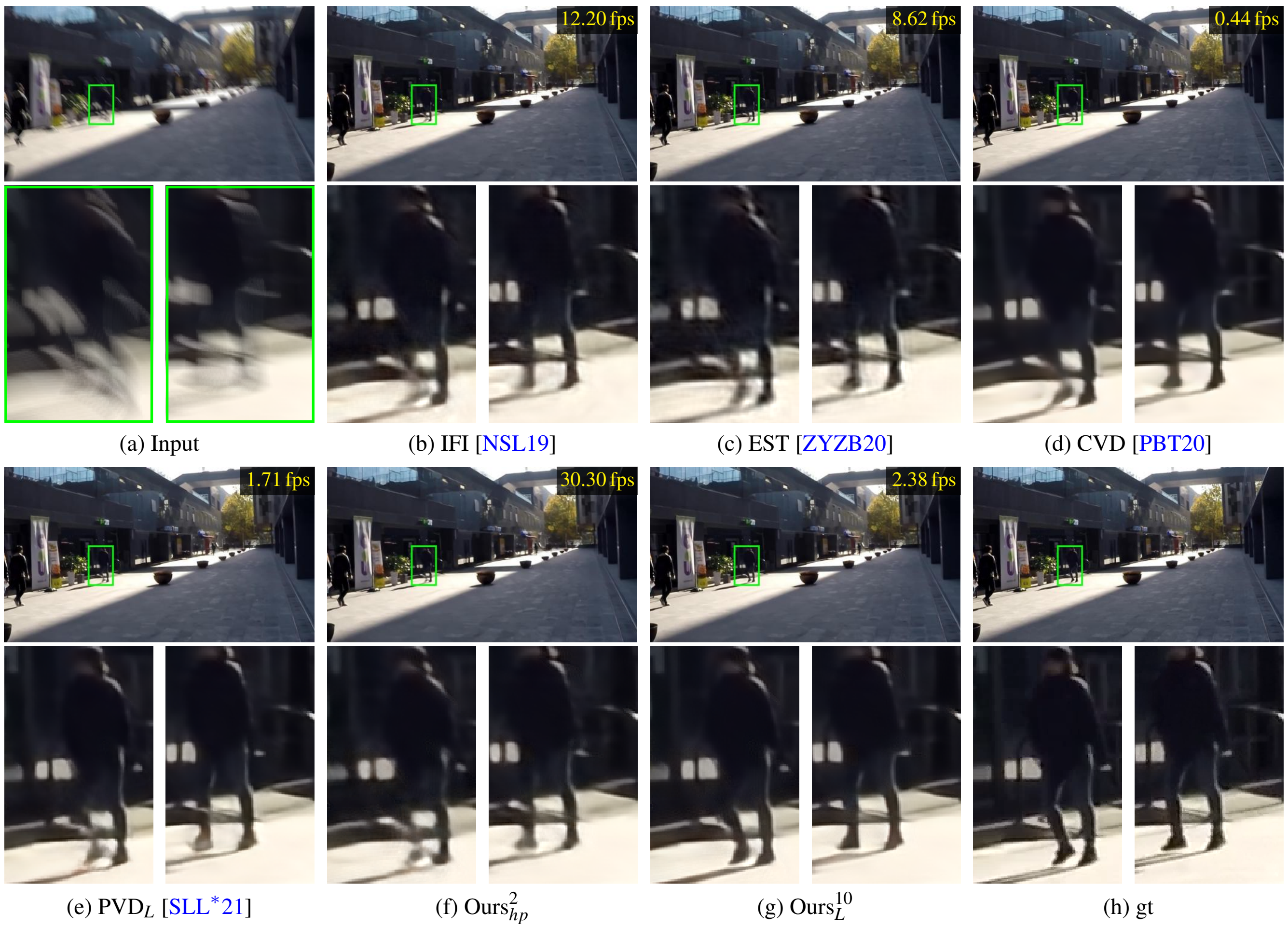


Figure 6: *Additional qualitative comparison on the GoPro dataset [NKL17].*

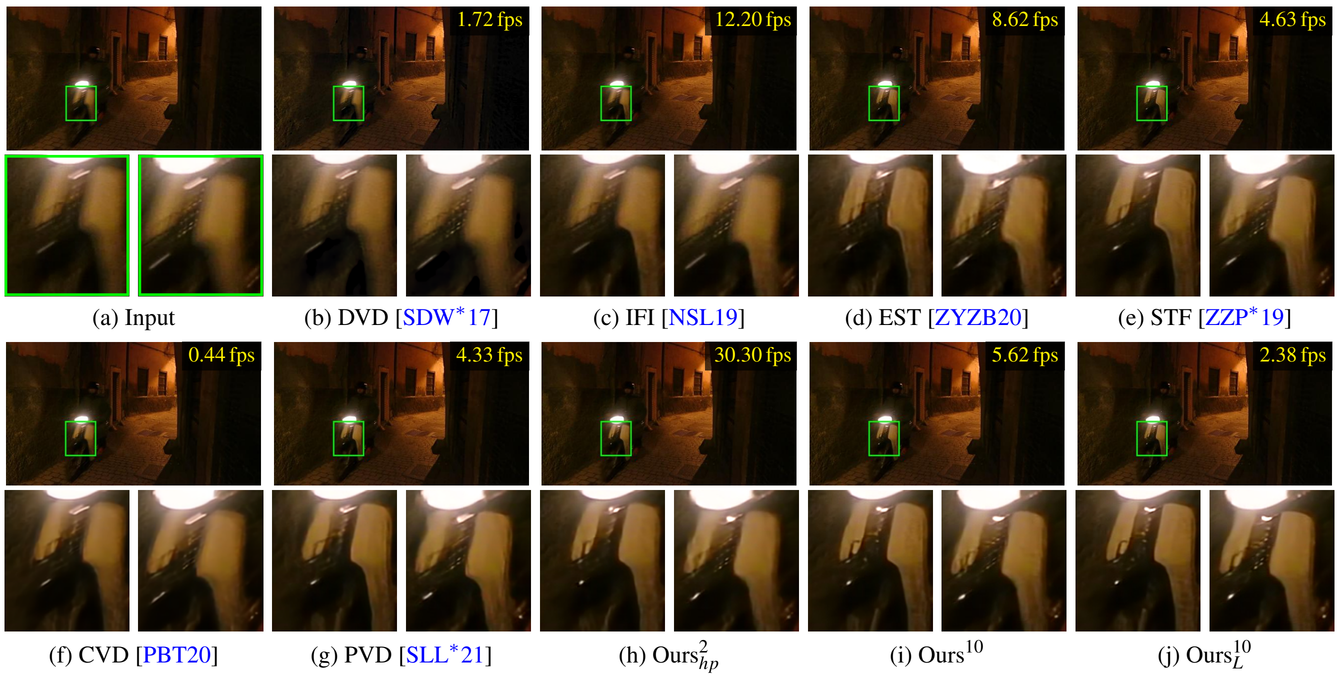


Figure 7: *Additional qualitative comparison on real-world blurred videos [CWL12].*

Overall network architecture

input	type	act	output	k	c	s	p	#
I_t^b	*conv*	*relu*	f_t^b	3	26	1	1	1
$[I_{t-2}^b \cdot I_{t-1}^b \cdot I_{t+1}^b \cdot I_{t+2}^b \cdot I_{t-1}^r]$	*conv*	*relu*	f_{t-1}	3	26	1	1	1
$[f_t^b \cdot f_{t-1}]$ `if` $n=1$ `else` $[f_t^{n-1} \cdot \tilde{f}_{t-1}^{N,n-1}]$	**MTU**^{n}	-	$[f_t^n \cdot \tilde{f}_{t-1}^{N,n}]$	-	52	-	-	N-1
$[f_t^b \cdot f_{t-1}]$ `if` $n=1$ `else` $[f_t^{n-1} \cdot \tilde{f}_{t-1}^{N,n-1}]$	**MTU**^{N}	-	I_n^r	-	3	-	-	1

Multi-tasking unit MTU^n

input	type	act	output	k	c	s	p	#
$[f_t^b \cdot f_{t-1}]$ `if` $n=1$ `else` $[f_t^{n-1} \cdot \tilde{f}_{t-1}^{N,n-1}]$	$\mathcal{F}^n$	-	f_t^n	-	-	-	-	1
f_t^n, f_t^b, f_{t-1}^N	$\mathcal{M}^n$	-	$[f_t^n \cdot \tilde{f}_{t-1}^{N,n}]$	-	-	-	-	1
$[f_t^n \cdot \tilde{f}_{t-1}^{N,n}]$	$\mathcal{D}^n$	-	$I_t^{r,n}$	-	-	-	-	1

Multi-task detail network $\mathcal{F}^n$

input	type	act	output	k	c	s	p	#
$[f_t^b \cdot f_{t-1}]$ `if` $n=1$ `else` $[f_t^{n-1} \cdot \tilde{f}_{t-1}^{N,n-1}]$	*conv*	*relu*	conv$_1$	5	26	1	1	1
conv$_1$	*conv*	*relu*	conv$_2$	3	26	2	1	1
conv$_2$	*conv*	*relu*	res$_0$	3	26	2	1	1
res$_0$	*identity*	-	skip	-	-	-	-	-
res$_0$	*conv*	*relu*	res$_{1-1}$	3	52	1	1	
res$_{1-1}$	*conv*	*relu*	res$_{1-2}$	3	52	1	1	8
res$_{1-2}$, res$_0$	*sum*	-	res$_0$	-	-	-	-	
res$_0$, skip	*sum*	-	res	-	-	-	-	1
res	*deconv*	*relu*	deconv$_1$	4	26	2	1	1
deconv$_1$, conv$_2$	*sum*	-	deconv$_1$	-	-	-	-	1
deconv$_1$	*deconv*	*relu*	deconv$_2$	4	26	2	1	1
deconv$_2$, conv$_1$	*sum*	-	f_t^n	-	-	-	-	1

Motion compensation network $\mathcal{M}^n$

input	type	act	output	k	c	s	p	#
f_t^n, f_t^b	*sum*	-	sum$_0$	-	-	-	-	
sum$_0$	*conv*	*relu*	$\hat{f}_t^n$	5	52	4	1	1
$\hat{f}_t^n, \hat{f}_{t-1}^n, f_{t-1}^N$	**MC**	-	$[f_t^n \cdot \tilde{f}_{t-1}^{N,n}]$	-	-	-	-	

Deblurring network $\mathcal{D}^n$

input	type	act	output	k	c	s	p	#
$[f_t^n \cdot \tilde{f}_t^{N,n}]$	*conv*	-	$I_t^{res,n}$	3	3	1	1	1
$I_t^{res,n}, I_t^b$	*sum*	-	$I_t^{r,n}$	-	-	-	-	1

Table 5: *Detailed network architectures.* ***MTU***^{n} *stands for the n-th multi-task unit in the N-stacked* ***MTUs****, and* ***MC*** *means the motion compensation module. In the columns,* `input`*,* `type`*,* `act`*,* `output`*,* `k`*,* `c`*,* `s`*,* `p` *and # denote the input, type, activation function, output, kernel size, out-channels, stride, padding, and repeating number of a layer, respectively. For the layer types, we have conv, deconv, identity, and sum, which denote convolution, deconvolution, identity, and element-wise summation layers, respectively.* $[\cdot]$ *indicates the concatenation operation in the channel direction.*