Real-World Efficient Blind Motion Deblurring via Blur Pixel Discretization

Supplementary Material

A. Implementation Details

A.1. Our model variants

The default number of blocks of NAFNet [6] is 36 which consists of encoder blocks $\{1, 1, 1, 28\}$, middle block $\{1\}$ and decoder blocks $\{1, 1, 1, 28\}$. NAFNet with 32 widths and 64 widths are referred to as NAFNet-32 and NAFNet-64, respectively. To be consistent with the computational cost of NAFNet, we build SegDeblur-S (14.44 GMACs) and SegDeblur-L (62.68 GMACs) for the realistic datasets such as RealBlur [27], RSBlur [28] and ReLoBlur [20]. Since FFTFormer [15] shows the best performance on GoPro [25], we build SegFFTFormer (135.81 GMACs) for GoPro. Our discrete-to-continuous (D2C) converter of SegDeblur-S is based on NAFNet and consists of encoder blocks $\{8, 8, 8, 22\}$, middle block $\{8\}$ and decoder blocks $\{8, 8, 8, 8\}$ with 16 widths (10.07 GMACs). Also, our D2C converter of SegDeblur-L is also based on NAFNet and consists of encoder blocks $\{5, 5, 5, 18\}$, middle block $\{5\}$ and decoder blocks $\{5, 5, 5, 5\}$ with 48 widths (58.31 GMACs). For SegFFTFormer, we use the original FFTFormer (131.45 GMACs) for our D2C converter. On the other hand, our blur pixel discretizer is based on NAFNet and shared with our SegDeblur-S, SegDeblur-L and SegFFTFormer, which consists of encoder blocks $\{2, 2, 2, 20\}$, middle block $\{2\}$ and decoder blocks $\{2, 2, 2, 2\}$ with 16 widths (4.37 GMACs). For our kernel estimator, we use U-Net [29] with 16 widths (1.16 GMACs), which performs four times of downsampling by stride 2 operations. We remark that our kernel estimator is only used for training, such that it is not included in our computational cost. Overall, our models are summarized in Table A.1.

Our Model	Model (GMACs)		Total GMACs	Total # Params (M)
	Blur Pixel Discretizer	D2C Converter		
SegDeblur-S	4.37	10.07	14.44	12.30
SegDeblur-L	4.37	58.31	62.68	55.40
SegFFTFormer	4.37	131.45	135.81	20.60

Table A.1. Our model variants. We present a detailed description of our individual variants.

B. Additional Ablation Study

B.1. Effects on deblurring network architecture

We investigate that our blur segmentation map still works well on the other network architectures. To confirm this, we apply our blur segmentation map to the network architectures such as NAFNet [6], FFTFormer [15] and FMIMOUNet [24]. We train the deblurring model with RealBlur-J [27]. Note that the network architectures such as NAFNet, FFTFormer and FMIMOUNet are scaled down to around 16 GMACs. The results are given in Table B.1.

	Prior	RealBlur-J	
		PSNR ↑	SSIM ↑
NAFNet [6]		31.99	0.920
	✓	**32.53**	**0.927**
FFTFormer [15]		32.08	0.917
	✓	**32.73**	**0.926**
FMIMOUNet [24]		31.25	0.903
	✓	**32.30**	**0.920**

Table B.1. Ablation study on network architectures. [6, 15, 24]. All network architectures are based on 16 GMACs.

B.2. Effects on model size

Our model consists of blur pixel discretizer and D2C converter. Here, we raise a fundamental question: which model size should we increase to improve performance? To address this, we conduct experiments with various model sizes by scaling the size of blur pixel discretizer and D2C converter. The results are shown in Table B.2. We observe that the small model

size of the blur pixel discretizer is sufficient to provide high deblurring performance (compare the performance in 1st-3rd row results and 2nd-4th row results). This is because the classification task (e.g., blur pixel discretization) is easier than the regression task (e.g., discrete-to-continuous conversion). Meanwhile, the model size of the D2C converter is the most important ingredient to improve performance as shown in Table B.2.

MACs (G)			PSNR	SSIM
Blur Pixel Discretizer	D2C converter	Total		
4.37	10.07	14.44	32.53	0.927
4.37	58.31	62.68	**32.95**	**0.934**
10.07	4.37	14.44	32.18	0.919
58.31	4.37	62.68	32.65	0.923

Table B.2. Ablation study on model sizes of our method. We make several combinations of our blur pixel discretizer and D2C converter to evaluate which model type contributes to more performance improvement when varying the size of each model.

B.3. Cross-data validation

To verify the generalization ability, we conduct the cross-data test. Namely, we train with RealBlur-J [27] and evaluate on RSBlur [28]. As reported in Table B.3, our SegDeblur-L gives better performance than NAFNet [6] at similar computational cost. Although the performance gap (0.2 dB) seems small, this performance gap may require $4\times$ increase in computational cost in RSBlur, as demonstrated in Table 4.

Methods	GMACs	RealBlur (Train) - RSBlur (Test)	
		PSNR ↑	SSIM ↑
NAFNet [32]	63.64	30.56	0.806
UFPNet [11]	243.33	30.75	0.812
SegDeblur-L (ours)	62.68	**30.76**	**0.813**

Table B.3. Ablation study on the cross-data validation. We train with RealBlur-J [27] and evaluate on RSBlur [28] The best results are indicated in bold.

B.4. Different datasets for blur pixel discretizer and D2C converter

To confirm how different datasets affect the performance, we conduct experiments in which we use the blur pixel discretizers trained with RealBlur-J [27], RSBlur [28] and GoPro [25] and train the D2C converter with RealBlur-J. The results are presented in Table B.4. The results show that it is crucial to use the same data in both blur pixel discretizer and D2C converter. On the other hand, using the different datasets leads to significant performance drop as shown in Table B.4. We believe that this is due to vulnerability to unseen data in deep models as well as variations arising from different image sensors, lenses, ISPs and motion types.

Training Set		PSNR ↑	SSIM ↑
Blur Pixel Discretizer	D2C Converter		
GoPro [25]	RealBlur-J	31.89	0.919
RSBlur [25]		31.93	0.921
RealBlur-J [27]		**32.53**	**0.927**

Table B.4. Ablation study on different datasets. To confirm the effect on different datasets for two models, we train with GoPro [25], RSBlur [28] and RealBlur-J [27] for our blur pixel discretizer and train with RealBlur-J for our D2C converter. The best results are indicated in bold.

B.5. Performance of large motion test set in RSBlur

As shown in Table 1, we demonstrate that our method is robust to large motion test set in RealBlur-J [27]. To additionally verify the robustness of large motion scenarios for our method, we construct the large motion test set of RSBlur [28] by extracting the largest motion 300 image pairs. The results are presented in Table B.5. Our efficient model, i.e., SegDeblur-S, improves PSNR in the large motion set (30.04 $\rightarrow$ 30.39 dB) compared with the counterpart, NAFNet-32. Furthermore, its large motion performance is also comparable to that of the larger model, NAFNet-64 (30.39 dB).

Methods	GMACs	RSBlur	
		Total	Large
NAFNet-32 [6]	16.25	33.71	30.04
NAFNet-64 [6]	63.64	33.97	30.39
SegDeblur-S (ours)	14.44	33.96	30.39
SegDeblur-L (ours)	62.68	**34.21**	**30.82**

Table B.5. Performance of large motion test set in RSBlur [28]. "Total" means the whole set while "Large" denotes the large motion set.

B.6. Comparison with kernel-based methods

As the kernel-based methods produce pixel-wise motion information such as motion trajectories and kernels, we can use such information as prior when training the deblurring model. To this end, we train the deblurring model (NAFNet-32) with such information and compare their results with our method. As shown in Table B.6, simply introducing prior information does not highly improve performance compared to NAFNet-32. Basically, estimating motion information for every pixel is a huge ill-posed problem. Therefore, their methods may produce inaccurate information, which is not helpful for subsequent deblurring tasks. On the other hand, our method overcomes the ill-posedness by introducing the latent sharp image and logarithmic fourier space as discussed in Section 3.2, which leads to better deblurring performance as shown in Table B.6.

	Prior	GMACs	PSNR ↑	SSIM ↑
NAFNet-32 [6]		16.25	31.99	0.920
Adaptive Basis [3]	✓	80.60	32.08	0.921
Exposure Trajectory [41]	✓	72.58	31.94	0.919
SegDeblur-S (ours)	✓	14.44	**32.53**	**0.927**

Table B.6. Comparison with kernel-based methods [3, 41].

C. Implementation details for mobile deployment

We modify some network architecture of our model according to the operations supported by AI accelerators in order to accelerate our method, which is called SegDeblur-S+. Specifically, SimpleGate, SCA, and LayerNorm in NAFNet [6] are removed. We train SegDeblur-S+ with a combination of RealBlur [27] and RSBlur [28]. We deploy our SegDeblur-S+ using 32-bit floating point precision on GPU of Qualcomm SM8550 chipset in Samsung Galaxy S23 without any quantization. The average inference time of our method is measured using TensorFlow Lite Benchmark Tool. Furthermore, we measure the averaged execution time in Samsung EnhanceX and Google Unblur based on the image size of 2000×2000. The comparison result for on-chip execution time is presented in Table C.1. Notably, our SegDeblur-S takes nearly twice of the execution time compared to the accelerated version of our method, SegDeblur-S+. This accounts for the importance of using the network architectures supported by AI accelerators for the mobile deployment. The on-chip execution time of SegDeblur-S+ (2.35s) lags behind that of Samsung EnhanceX (1.64s) and Google Unblur (2.03s). Our current implementation is based on on-chip GPU, such that it can be more accelerated if deployed on Neural Processing Unit (NPU). We conjecture that Samsung EnhanceX and Google Unblur are implemented on NPU to achieve faster processing.

Methods	Device	Time	GMACs
Samsung EnhanceX	-	1.64s	-
Google UnBlur	-	2.03s	-
SegDeblur-S+ (ours)	GPU	2.35s	14.05
SegDeblur-S (ours)		4.10s	14.44

Table C.1. On-chip execution time. We measure the averaged execution time on image size of 2000 × 2000.

D. Generalization to real-world blur images

We present more real-world examples to compare our method with the recent works and commercial applications. We capture real-world blur images with natural hand motions by Samsung Galaxy Note S20 Ultra. As shown in Fig. D.1, D.2 and D.3, the real-world blur images are well-reconstructed by our method. Meanwhile, the recent works and deblurring applications work well on some blur images but some other blur images are not well-recovered.

(a) Blur Input

(b) Samsung EnhanceX

(c) Google Unblur

(d) NAFNet-32 [6]

(e) FFTFormer-16 [15]

(f) SegDeblur-S (ours)

Figure D.1. Visual comparison results on real-world blur images.

(a) Blur Input

(b) Samsung EnhanceX

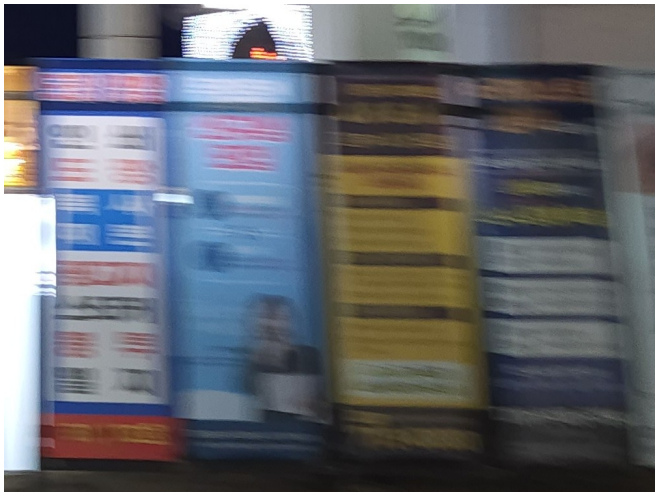
(c) Google Unblur

(d) NAFNet-32 [6]

(e) FFTFormer-16 [15]

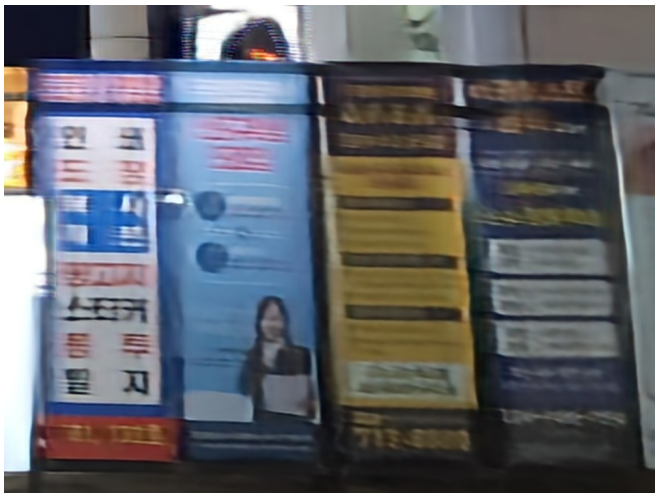
(f) SegDeblur-S (ours)

Figure D.2. Visual comparison results on real-world blur images.

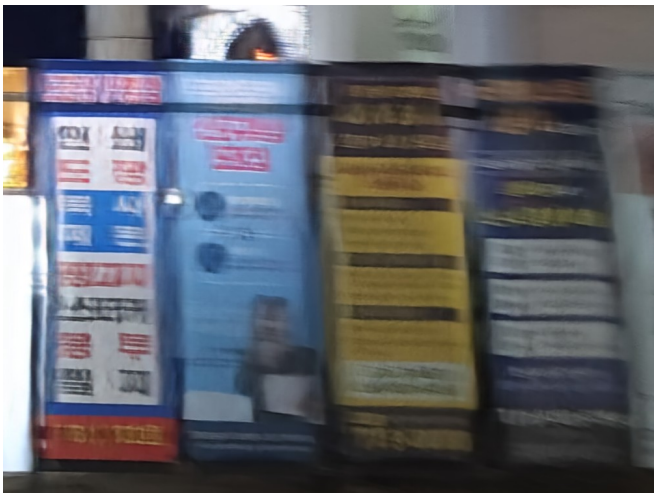


(a) Blur Input

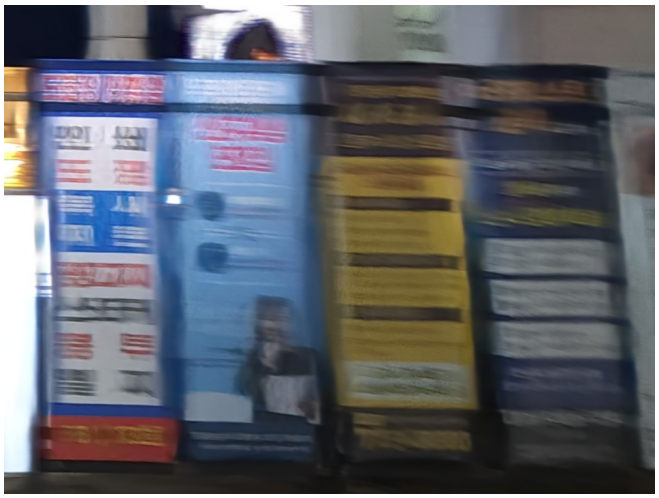
(b) Samsung EnhanceX



(c) Google Unblur



(d) NAFNet-32 [6]

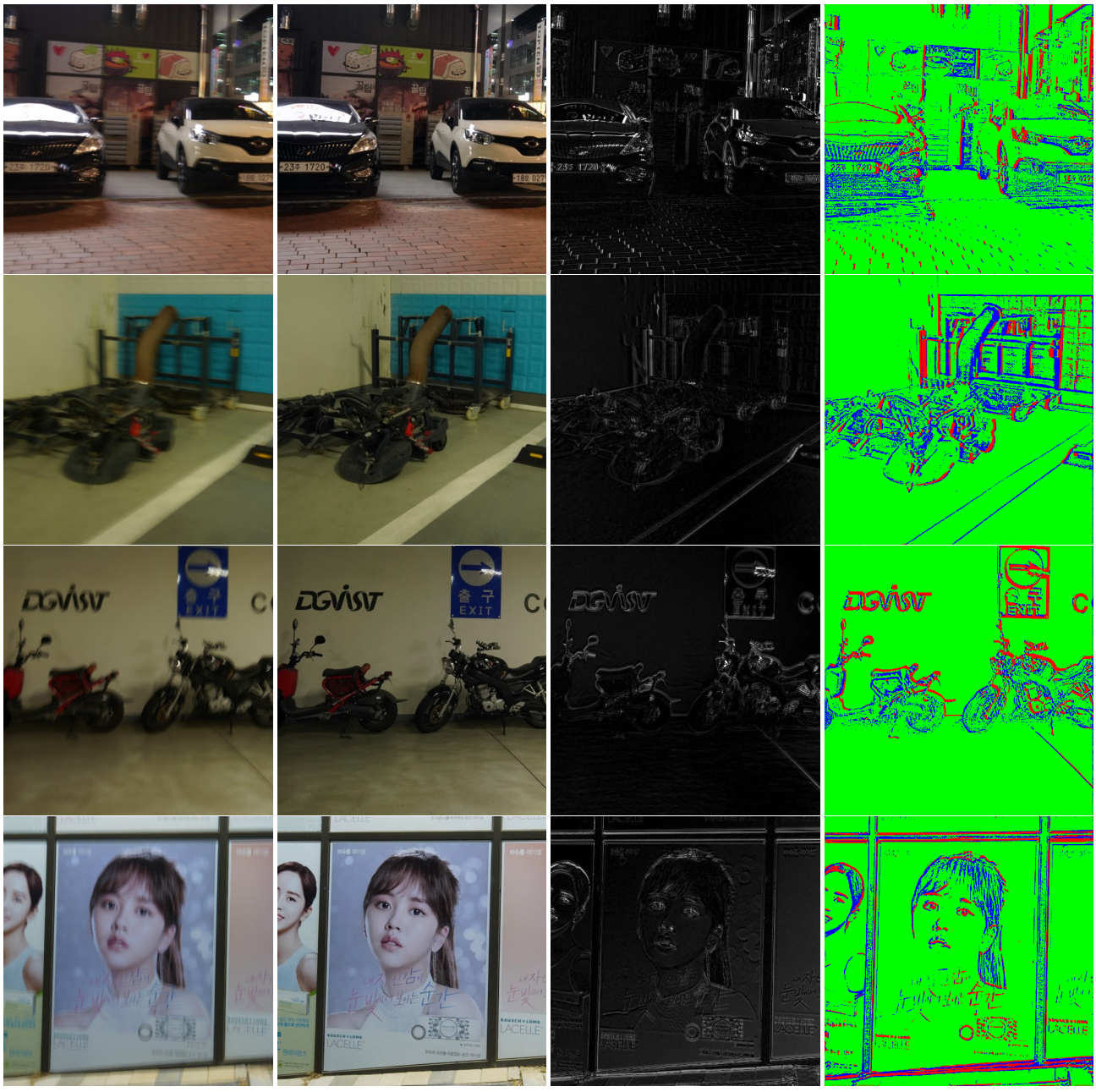


(e) FFTFormer-16 [15]

(f) SegDeblur-S (ours)

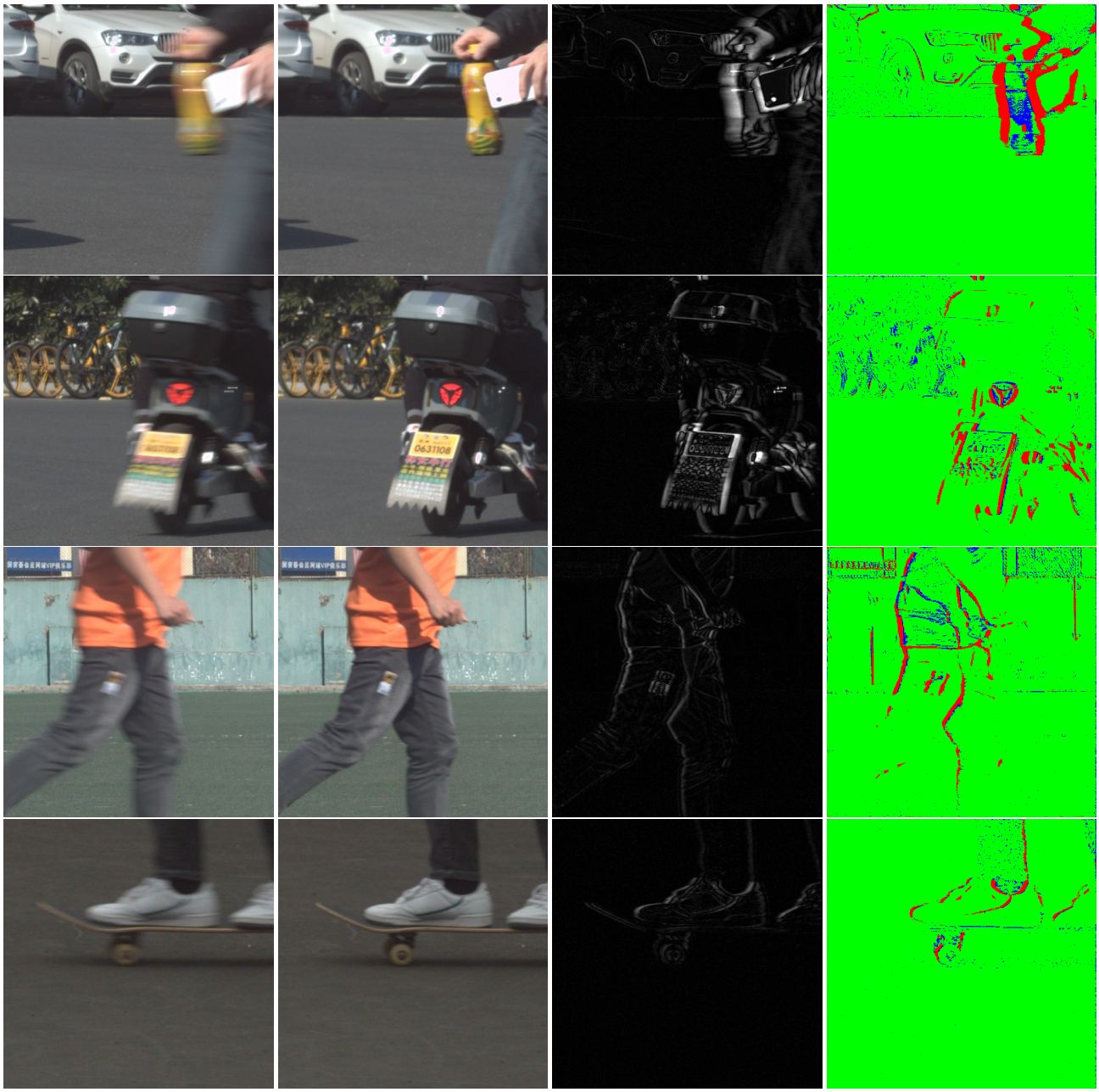
Figure D.3. Visual comparison results on real-world blur images.

E. Visual results on blur segmentation map



(a) Blur Image (b) Sharp Image (c) Image Residual Error (d) Blur Segmentation Map

Figure E.1. Blur segmentation map results on camera motion examples for RealBlur [27].



(a) Blur Image (b) Sharp Image (c) Image Residual Error (d) Blur Segmentation Map

Figure E.2. Blur segmentation map results on object motion examples for ReLoBlur [20].

F. Qualitative results for large deblurring models

(a) Blur Input

(b) MAXIM-3S [35]

(c) UFPNet [11]

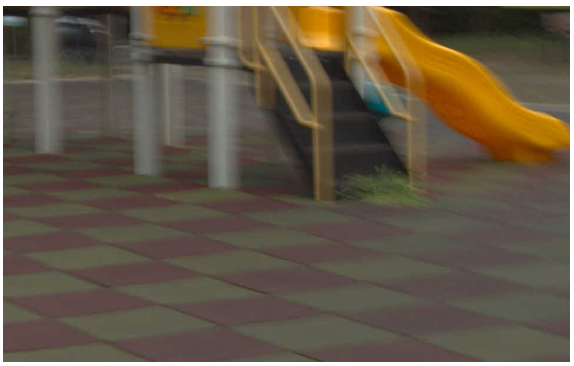
(d) SegDeblur-L (ours)

(a) Blur Input

(b) MAXIM-3S [35]

(c) UFPNet [11]

(d) SegDeblur-L (ours)



(a) Blur Input

(b) MAXIM-3S [35]

(c) UFPNet [11]

(d) SegDeblur-L (ours)

(a) Blur Input

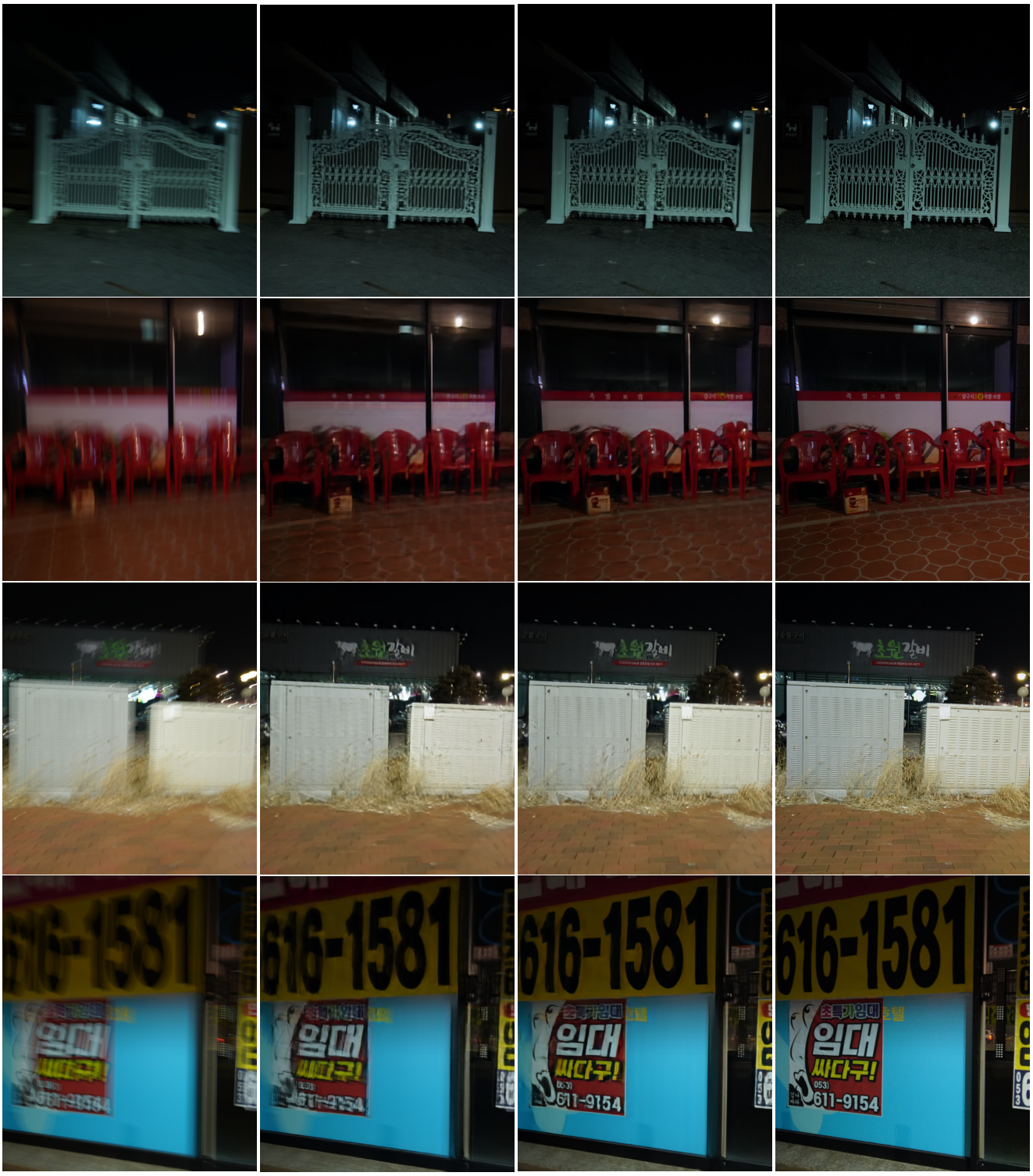
(b) MAXIM-3S [35]

(c) UFPNet [11]

(d) SegDeblur-L (ours)

Figure F.1. Qualitative results for large deblurring models in RSBlur [28].

G. Qualitative results for efficient deblurring models



(a) Blur Input (b) NAFNet-32 [6] (c) SegDeblur-S (ours) (d) Ground-Truth Sharp

Figure G.1. Qualitative results for efficient deblurring models in RealBlur [27].

(a) Blur Input

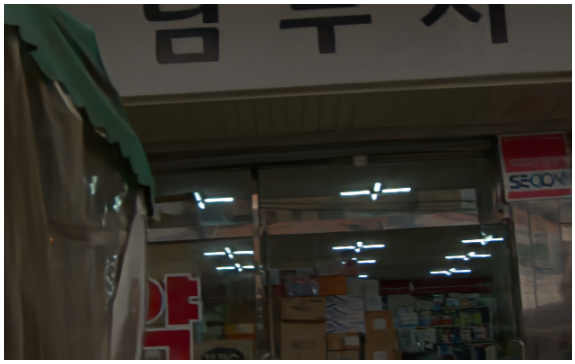
(b) FFTFormer-16 [15]

(c) SegDeblur-S (ours)

(d) Ground-Truth Sharp

(a) Blur Input

(b) FFTFormer-16 [15]



(c) SegDeblur-S (ours)

(d) Ground-Truth Sharp

(a) Blur Input

(b) FFTFormer-16 [15]

(c) SegDeblur-S (ours)

(d) Ground-Truth Sharp

(a) Blur Input

(b) FFTFormer-16 [15]

(c) SegDeblur-S (ours)

(d) Ground-Truth Sharp

Figure G.2. Qualitative results for efficient deblurring models in RSBlur [28].