Real-World Efficient Blind Motion Deblurring via Blur Pixel Discretization

Insoo Kim[1,2] Jae Seok Choi[1] Geonseok Seo[1] Kinam Kwon[1] Jinwoo Shin[2*] Hyong-Euk Lee[1*]
[1]Samsung Advanced Institute of Technology (SAIT), South Korea
[2]Korea Advanced Institute of Science and Technology (KAIST), South Korea

Abstract

As recent advances in mobile camera technology have enabled the capability to capture high-resolution images, such as 4K images, the demand for an efficient deblurring model handling large motion has increased. In this paper, we discover that the image residual errors, i.e., blur-sharp pixel differences, can be grouped into some categories according to their motion blur type and how complex their neighboring pixels are. Inspired by this, we decompose the deblurring (regression) task into blur pixel discretization (pixel-level blur classification) and discrete-to-continuous conversion (regression with blur class map) tasks. Specifically, we generate the discretized image residual errors by identifying the blur pixels and then transform them to a continuous form, which is computationally more efficient than naively solving the original regression problem with continuous values. Here, we found that the discretization result, i.e., blur segmentation map, remarkably exhibits visual similarity with the image residual errors. As a result, our efficient model shows comparable performance to state-of-the-art methods in realistic benchmarks, while our method is up to 10 times computationally more efficient.

1. Introduction

Motion blur is caused by the camera motion and object movement within the exposure time. As the long exposure time is required to ensure the amount of light in low-light environments, it leads to significant blur degradation, which is a challenge to overcome. Given blur images, the blind image deblurring procedure aims to reconstruct sharp images.

In earlier years, many researchers have investigated accurate blur kernel estimations [3, 4, 13, 30, 33, 41]. Those kernel-based methods focus on estimating blur kernels and then exploiting them in the deconvolution process [12, 16], resulting in the final sharp images. They mainly train with synthesized motion blur images that may not hold in practice. In recent years, kernel-free methods have been more widely studied in motion deblurring tasks [6, 8, 11, 15, 17–19, 21, 24–26, 32, 34, 35, 38–40]. They achieve state-of-art performance despite their simplicity where they do not consider such blur kernels. However, their performance highly relies on network capacities [8, 21, 35, 38], which may restrict from practical usage. Some literature introduces self-generated prior information to further improve deblurring performance, but generating the prior information consumes additional huge computational costs [11, 19].

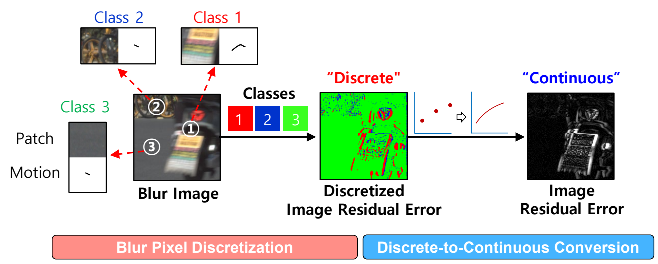


Figure 1. Blind motion deblurring via blur pixel discretization. The image residual error, i.e., blur-sharp pixel differences is estimated by our blur pixel discretization and discrete-to-continuous conversion, whose result is used to produce final deblurred images.

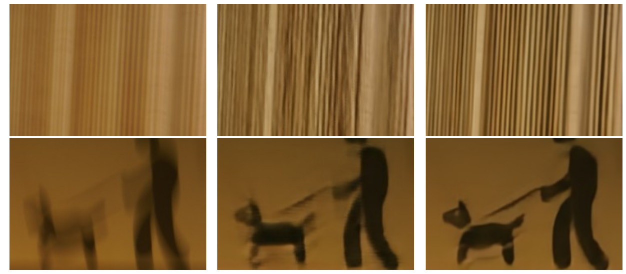
As recent advances in camera technology have enabled the capability to capture high-resolution images, such as 4K images, dealing with large motion has become increasingly important. To deploy a deblurring model on resource-constrained devices (e.g., mobile devices), an efficient model capable of handling large motion is required. Nevertheless, these models have not been thoroughly explored. In our earlier experiments, we found that the small-scale model suffers from significant performance drops in the case of large motion scenarios as shown in Table 1. In fact, the small-scale deblurring model without any prior information may be susceptible to distortions.

In this paper, we rethink the deblurring task by decomposing the original regression problem into blur pixel discretization and discrete-to-continuous (D2C) conversion problems to explore an efficient deblurring model for practical usage, as shown in Fig. 1. Our intuition is that we build a discrete version of the ground truth (GT), namely, a discretized image residual error (e.g., discrete blur-sharp pixel differences) by identifying the blur pixels and then transform it into a continuous form, which is computationally

*Corresponding authors

(a) (b) (c) (d)

Figure 2. Visual comparison in the object motion blur (above) and uniform blur (below): (a) Sharp image, (b) Blur image, (c) Image residual error, and (d) Blur segmentation map. We utilize three classes and the colors in blur segmentation map indicate classes.



(a) (b) (c)

Figure 3. Visual comparison results on the efficient models: (a) Blur image, (b) NAFNet [6], and (c) Proposed method. The computational cost of both models is around 16 GMACs. NAFNet is vulnerable to distortions since it has no blur class information, whereas our method produces more natural deblurring results.

more efficient than naïvely solving the deblurring regression problem. As shown in the upper side of Fig. 2 (c), the object movement regions (e.g., legs) highlight image residual errors, i.e., blur-sharp pixel differences, when compared to non-moving regions. In contrast, as shown in the lower side of Fig. 2 (c), despite the same blur for all pixels (i.e., uniform blur), the high-frequency region (e.g., edges) shows noticeable image residual errors rather than those of the low-frequency region (e.g., wall). Namely, the image residual error is characterized by its motion type (e.g., motion amount and direction) and spatial frequency (e.g., complexity of neighboring pixels). Using this *motion-frequency* property, the image residual error may be grouped into some categories, e.g., low or high levels.

Inspired by those observations, we propose a new deblurring scheme that decomposes the deblurring model into *blur pixel discretizer* and *D2C converter* as shown in Fig. 4. In the first stage, our blur pixel discretizer is trained to yield the discretized image residual error at a low computational cost, which is referred to as *blur segmentation map*. In the second stage, the D2C converter efficiently transforms the discrete version of image residual error into a continuous form, leading to better deblurring results. In particular, the blur pixels are identified by the following steps: (1) To reflect the nature of the image residual error, e.g., motion-frequency property, we first predict a *motion-related* several basis kernels and they are used for a deconvolution procedure with the *frequency-related* neighboring pixels, resulting in a set of deconvolved images, i.e., deconvolved class images. (2) We introduce a blur segmentation map that contains a per-pixel class index. Based on this information, we perform pixel-wise sampling from the deconvolved class images to predict the optimal deconvolved image. (3) As a result, all blur pixels are optimally categorized into classes through our blur segmentation map, which visually aligns with the image residual error as shown in Fig. 2 (c) and (d). Furthermore, we pay attention to the logarithmic fourier space to simplify the relationship between blur and sharp images so that the basis kernels are easily trained in our kernel estimator as shown in Fig. 4. Our contributions are summarized as follows:

- We propose a new deblurring scheme that decomposes a deblurring regression task into simpler tasks: blur pixel discretization and D2C conversion tasks, which is computationally more efficient than naïvely solving the deblurring regression problem.
- Our blur pixel discretizer produces the blur segmentation map, which reflects the nature of the image residual error. Hence, the proposed method can be interpreted as deblurring with GT-like information, leading to better deblurring results at a low computational cost.
- Our efficient deblurring model demonstrates its competitiveness not only even with a reduction of up to 10× in the computational cost compared to larger deblurring methods in realistic benchmarks but also in commercial applications such as Samsung EnhanceX and Google Unblur.

2. Related Works

Kernel-based methods. In blind motion deblurring tasks, kernel-based methods [4, 30, 31] directly learn a per-pixel kernel and then produce sharp images using the existing non-blind deblurring methods [10, 12, 16]. To estimate the blur kernels easier, a motion flow estimation [13] is studied to produce a motion flow map. Similarly, the exposure trajectory estimation framework [41] is investigated to estimate a set of motion offsets. The adaptive basis decomposition scheme [3] introduces a set of pixel-shared basis kernels and a set of pixel-wise mixing coefficients to efficiently estimate a per-pixel kernel. Overall, those kernel-based methods aim to generate sharp images by utilizing motion flows, trajectories, and basis kernels at a large cost, whereas ours focuses on providing per-pixel blur class information (distinct from the motion flows, trajectories and basis kernels) at a small cost.

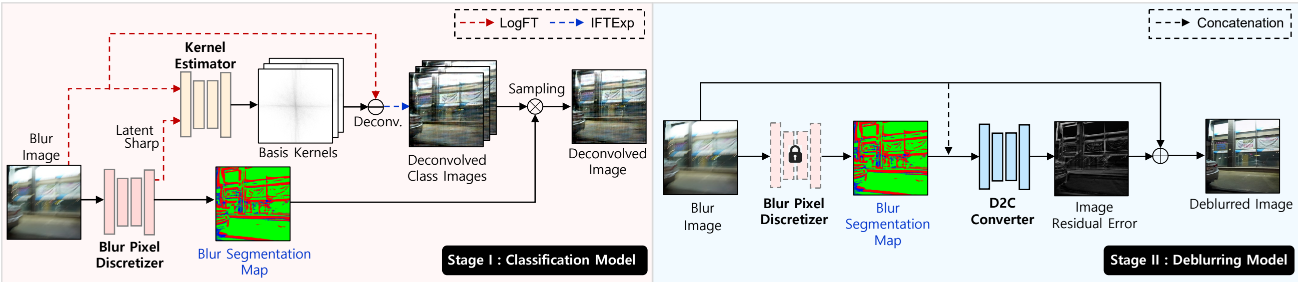


Figure 4. Our network architectures. The proposed method consists of two stages and each stage is shaded in different colors. In the first stage, we train our blur pixel discretizer and kernel estimator. In the second stage, we proceed to train our D2C converter with the frozen blur pixel discretizer. LogFT indicates the logarithmic fourier transform and IFTExp means the inverse logarithmic fourier transform.

Kernel-free methods. The kernel-free methods [6, 8, 14, 25] treat the image deblurring tasks as image-to-image translation tasks that map from blur to sharp in realistic blur datasets [25, 27]. Some pioneering works [25, 32] introduce the coarse-to-fine strategy that gradually recovers the sharp image from low-resolution to high-resolution images. Some frameworks that use multi-input and multi-output U-Net [29] are proposed to handle the coarse-to-fine strategy in an efficient way [8, 24, 39]. Transformer-based image deblurring methods [15, 21, 34, 38, 40] are studied to recover more image details by presenting their own ways to overcome the quadratic complexity of the self-attention modules. Some literature investigates the network architecture design to capture local and global information effectively [6, 35, 39, 40]. Recently, some methods introduce additional self-generated priors to improve deblurring performance [11, 19]. However, their priors are not directly related to the image residual error (e.g., GT) which may limit the performance improvement. On the other hand, our method generates a discretized image residual error (e.g. discrete GT), which results in better performance.

3. Real-World Efficient Motion Deblurring

In this section, we describe a new deblurring scheme that consists of blur pixel discretizer and discrete-to-continuous (D2C) converter. First, we explore a new perspective of motion deblurring tasks in Section 3.1. Then, we introduce a blur pixel discretizer that yields blur segmentation map in Section 3.2. In Section 3.3, a D2C converter with blur segmentation map is explained. Finally, we present the implementation details of our method in Section 3.4.

3.1. A new perspective of motion deblurring

As discussed in Section 1, the image residual error is determined by the motion (e.g., motion amount and direction) and frequency properties (e.g., complexity of neighboring pixels). To produce meaningful discrete image residual errors, we revisit a kernel-based method that is able to estimate motion kernels and consider neighboring pixels via a deconvolution procedure, resulting in sharp images.

Consider a dataset $\mathcal{D} = \{(x, y)\}$, which contains a pair of blur image y and sharp image x in the continuous domain $\mathcal{X}$. Given a uniform blur kernel k, the sharp image x is reconstructed by $x = y \odot k$ where $\odot$ is a deconvolution operation. To extend it to non-uniform deblurring, the sharp pixel value x_i can be obtained by a per-pixel deconvolution, i.e., $x_i = \mathcal{P}_i y \odot k^{(i)}$ where $i = 1, 2, \dots, N$ is the pixel index, $\mathcal{P}_i$ is an operator to extract the patch (i.e., *neighboring pixels*) at a pixel i and $k^{(i)}$ is a per-pixel kernel at a pixel i. Since we aim to categorize the blur pixels into classes, we instead consider a set of uniform motion basis kernels, $k = \{k^{(r)}\}_{r=1}^{R}$ where r is the basis kernel index. Given the basis kernels k, a set of deconvolved images, i.e., deconvolved class images $\nu = \{\nu^{(1)}, \nu^{(2)}, \dots, \nu^{(R)}\}$ where $\nu^{(r)} = \{\nu_i^{(r)}\}_{i=1}^{N}$ are computed by $\nu = y \odot k$, and a deconvolved class pixel value $\nu_i^{(r)}$ is defined by

$$\nu_i^{(r)} = \mathcal{P}_i y \odot k^{(r)}. \tag{1}$$

To determine which class r is more relevant to a given pixel i, we introduce blur segmentation map $\rho = \{\rho_i\}_{i=1}^{N}$ that contains a per-pixel class index $\rho_i \in \{1, 2, \dots, R\}$ in the discrete domain $\mathcal{Z}$. Then, the deconvolved image $\tilde{x} = \{\nu_i^{(\rho_i)}\}_{i=1}^{N}$ is recovered by combining the relevant deconvolved class pixel values using individual per-pixel class indices. Since the deconvolved class pixel values are determined by the motion-related basis kernels $k^{(r)}$ and frequency-related neighboring pixels $\mathcal{P}_i y$ in (1), our blur segmentation map implicitly reflects the characteristics of the image residual errors, as shown in Fig. 2 (c) and (d).

3.2. Logarithmic fourier discretization model

To successfully train our classification model as shown in Fig. 4, we present two key strategies to reduce the nature of ill-posedness: (i) we utilize the logarithmic fourier space [7] to simplify the relationship between blur and sharp images so that the basis kernels are easily estimated, and (ii) we introduce a latent sharp image in order to restrict the number of feasible kernel solutions, thereby making it simpler to train our classification model. Those key techniques facilitate implementing our classification model at a low com-

Figure 5. Visual comparison results on RSBlur [28]. We compare our SegDeblur-S (14.44 GMACs) with FFTFormer-16 [15] (16.41 GMACs) and NAFNet-32 [6] (16.25 GMACs). Note that all methods are trained with RSBlur.

putational cost. Hence, we first consider a fourier operator $\mathcal{F}$. It has an inherent property that converts from deconvolution to division, i.e., $y \odot k \leftrightarrows \mathcal{F}(y)/\mathcal{F}(k)$ [2]. To further simplify the relationship between two fourier samples, we apply the logarithmic operation to the fourier space (i.e., logarithmic fourier transform $\mathcal{F}_\mathcal{L}$ of sample images), and then we obtain the relationship $y \odot k \leftrightarrows \mathcal{F}_\mathcal{L}(y) - \mathcal{F}_\mathcal{L}(k)$ [7].

Let $X = \mathcal{F}_\mathcal{L}(x)$ and $Y = \mathcal{F}_\mathcal{L}(y)$ be sharp and blur samples by the logarithmic fourier operator $\mathcal{F}_\mathcal{L}$. We are interested in estimating a set of logarithmic fourier basis kernels, $\tilde{K} = \{\tilde{K}^{(r)}\}_{r=1}^{R}$ where r is a basis kernel index, and blur segmentation map ρ modeled by classification model g, i.e.,

$$g : y \rightarrow (\tilde{K}, \rho). \tag{2}$$

Our classification model $g_{\psi,\phi}$ consists of two sub-modules for a blur pixel discretizer h_ϕ and kernel estimator k_ψ as shown in Fig. 4. To restrict the number of feasible kernel solutions, i.e., reduce the nature of ill-posedness, the blur pixel discretizer additionally produces the latent sharp image $\tilde{x}_\ell$ that is $h_\phi : y \rightarrow (\rho, \tilde{x}_\ell)$. Then, the kernel estimator k_ψ takes two inputs, e.g., blur and latent sharp samples $(Y, \mathcal{F}_\mathcal{L}(\tilde{x}_\ell))$ in the logarithmic fourier space and predicts a set of logarithmic fourier basis kernels $\tilde{K}$. Given the basis kernels $\tilde{K}$, a set of deconvolved images, i.e., deconvolved class images $\nu = \{\nu^{(1)}, \nu^{(2)}, \ldots, \nu^{(R)}\}$, is generated by using simple subtraction (e.g., deconvolution) and inverse logarithmic fourier transform ${\mathcal{F}_\mathcal{L}}^{-1}$, i.e.,

$$\nu = {\mathcal{F}_\mathcal{L}}^{-1}(Y - \tilde{K}). \tag{3}$$

Finally, we can extract the deconvolved pixels $\tilde{x}_d = \{\nu_i^{(\rho_i)}\}_{i=1}^{N}$ by choosing individual pixels within the deconvolved class images ν based on a per-pixel class index ρ_i. Namely, the deconvolved ith pixel of $\tilde{x}_d$ is determined by ith pixel of the chosen deconvolved class image $\nu^{(\rho_i)}$. Then, the model parameters ψ and ϕ are jointly optimized by minimizing the following loss:

$$L_{\texttt{class}}(\psi, \phi; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} d(\tilde{x}_d, x) + \lambda d(\tilde{x}_\ell, x), \tag{4}$$

where d is some distance or divergence in the image domain, e.g., the PSNR loss [6] and λ is a hyperparameter to control the contribution of a latent sharp image. In our method, the deblurring regression problem is tackled by reframing it as a blur classification task, which allows us to build our model $g_{\psi,\phi}$ at a low computational cost.

3.3. Discrete-to-continuous conversion model

As discussed in Section. 3.2, we demonstrate that our classification model produces a discretized image residual error (i.e., discrete version of GT) under a small computational cost (4 GMACs). Therefore, it may allow for less computational cost of the subsequent task, i.e., D2C converter, to transform such discretized image residual error ρ in the discrete domain into the image residual error e in the continuous domain, i.e., $f_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, rather than directly regressing the image residual error without such GT-like information. As a result, we can achieve an efficient deblurring model (14 GMACs) whose performance is still comparable to that of state-of-the-art methods as shown in Table 3.

Given the discretized image residual error (i.e., blur segmentation map ρ) and blur image y, the final deblurred image $\hat{x}$ is estimated via our D2C converter f_θ as follows:

$$\hat{x} = y + f_\theta(\rho, y), \tag{5}$$

where $f_\theta(\rho, y)$ is the estimated image residual error $\hat{e}$ and $\hat{x}$ denotes the final deblurred image. Then, we derive our deblur loss that minimizes the distance between the real sharp and deblurred images by

$$L_{\texttt{deblur}}(\theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} d(y + f_\theta(\rho, y), x), \tag{6}$$

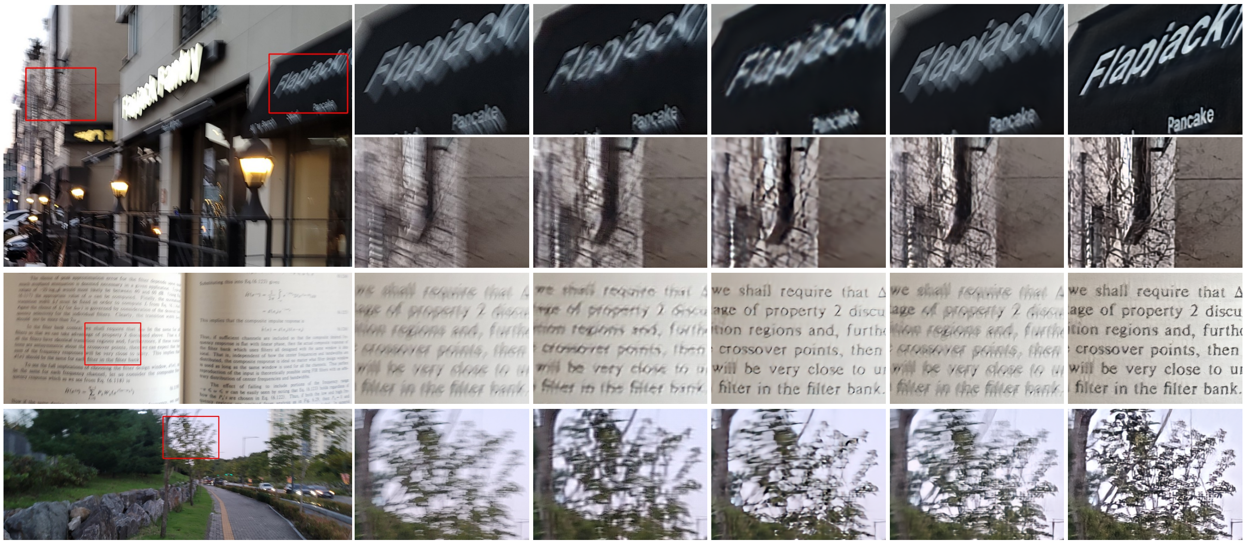


Figure 6. Visual comparison results on real-world blur images. NAFNet-32+ and SegDeblur-S+ are the accelerated (faster) models for the deployment and they are evaluated for fair comparison with the commercial applications. This will be more discussed in Section 4.4.

where d is some distance or divergence in the image domain, e.g., the PSNR loss [6]. Note that the classification model $g_{\psi,\phi}$ is not updated by (6). Since our classification model produces a blur segmentation map based on a deconvolution procedure, it may be more grounded in physical blur modeling than the simple blur-to-sharp mapping that the kernel-free methods do. Hence, Under an efficient model, the proposed method equipped with our blur segmentation map reconstructs more natural deblurred results as shown in Fig. 3 (c) than the kernel-free method without any physical blur information as shown in Fig. 3 (b).

3.4. Implementation details

In this section, we present the implementation details of our method. We use NAFNet [6] as our blur pixel discretizer and D2C converter while U-Net [29] is chosen as our kernel estimator. We adopt a two-stage training strategy for our method. In the first stage, we jointly train our blur pixel discretizer h_ϕ and kernel estimator k_ψ, with blur images y by minimizing (4), as shown in Fig. 4. The element of the blur segmentation map, i.e., ρ_i, is structured as a one-hot vector. We use *argmax* operation to obtain the per-pixel class index. Then, with this class information, we conduct pixel-wise sampling from the deconvolved class images to recover the deconvolved image $\tilde{x}_d$. In the second stage, we optimize our D2C converter f_θ with the blur segmentation map ρ and blur images y by using (6), as shown in Fig. 4. Note that the blur segmentation map estimated by the blur pixel discretizer is represented as probabilities, for example, [0.5,0.2,0.2,0.1] for each pixel. In the second and test stages, it is reconstructed by setting the maximum value to 1 and all other values to 0 to provide clearer blur class information to the following D2C converter. Furthermore, our blur pixel discretizer h_ϕ is frozen, and our kernel estimator k_ψ and the following operations such as logarithmic fourier transform, deconvolution, and sampling are discarded in the second and test stages.

4. Experiments

4.1. Experimental setup

Dataset and evaluation metrics. We use RealBlur [27], RSBlur [28], ReLoBlur [20] and GoPro [25] datasets for training and evaluation. RealBlur, RSBlur and ReLoBlur are realistic motion blur datasets that capture blur and sharp images in the same scene by using the beam splitter, whereas GoPro is a synthetic dataset where the blur image is generated by averaging sharp video frames captured by a high-speed camera. RealBlur consists of RealBlur-J (sRGB domain) and RealBlur-R (RAW domain). Each RealBlur type comprises 3,758 and 980 image pairs for training and test sets, respectively. RSBlur contains 8,878 and 3,360 blur-sharp image pairs for training and test sets, respectively. ReLoBlur consists of 2,010 and 395 image pairs for training and test sets, respectively. GoPro contains 2,103 and 1,111 blur-sharp image pairs for training and test sets, respectively. To verify the performance of the proposed method, we measure Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [37]. To measure model efficiency, we compute the number of network parameters and Multiply–ACcumulate operations (MACs) based on the image size of 256×256. Also, we compute on-chip execution time based on the image size of 2000×2000 when compared with commercial applications.

Methods	Params	GMACs			RealBlur-J	
		Prior	Deblur	Total	Total	Large
NAFNet-32 [6]	16.00	-	16.25	16.25	31.99	28.27
NAFNet-64 [6]	63.50	-	63.64	63.64	32.50	29.06
SegDeblur-S (ours)	12.30	4.37	10.07	14.44	32.53	29.12
SegDeblur-L (ours)	55.40	4.37	58.31	62.68	**32.95**	**29.77**

Table 1. The effect of blur segmentation map compared against NAFNet [6]. We present prior (blur pixel discretizer), deblur (D2C converter or NAFNet itself), and total model GMACs, respectively. We evaluate the methods on the whole and large motion test set of RealBlur-J [27], which denotes as "Total" and "Large".

	Prior	GMACs	PSNR ↑	SSIM ↑
HiNet [5]		170.71	32.12	0.921
MSDI-Net [19]	✓	336.43	32.35	0.923
NAFNet* [6]		63.64	33.12	0.930
UFPNet* [11]	✓	243.33	33.35	0.934
SegDeblur-L* (ours)	✓	62.68	**33.51**	**0.938**

Table 2. Comparison with prior-based methods [11, 19]. "*" denotes that the methods use the test-time local converter (TLC) [9]. Although our prior knowledge (blur segmentation map) is generated from a small-scale model, it significantly improves performance compared to the others.

Network architecture variants. NAFNet [6] consists of encoder blocks $\{1, 1, 1, 28\}$, middle block $\{1\}$ and decoder blocks $\{1, 1, 1, 28\}$. NAFNet with 32 and 64 widths are denoted as NAFNet-32 and NAFNet-64. Similarly, we build SegDeblur-S (14.44 GMACs) and SegDeblur-L (62.68 GMACs) for realistic datasets such as RealBlur [27], RSBlur [28] and ReLoBlur [20]. Since FFTFormer [15] is the best performance on GoPro [25], we build SegFFTFormer (135.81 GMACs) for GoPro. Our network variants are described in detail in Section A.1 of Appendix.

Implementation details. We train with RealBlur [27], RSBlur [28], ReLoBlur [20] and GoPro [25] randomly cropped by 256×256. We train our SegDeblur-S and L up to $1,000$ and $2,000$ epochs while our SegFFTFormer is trained up to $60,000$ iterations as in [15]. The batch size is 16 with 1 GPU for SegDeblur-S, 32 with 4 GPUs for SegDeblur-L, and 16 with 8 GPUs for SegFFTFormer. Our blur pixel discretizer, kernel estimator and D2C converter are optimized by the AdamW [23] algorithm ($\beta_1 = 0.9$, $\beta_2 = 0.9$ and weight decay $1e^{-3}$) with the cosine annealing schedule ($1e^{-3}$ to $1e^{-7}$) [22] gradually reduced for total iterations of each dataset. We employ the hyperparameter as $\lambda = 1.0$. We use the number of classes as $R = 16$ for RealBlur and $R = 8$ for the other datasets. We use RealBlur-J for all ablation studies.

4.2. Blind motion deblurring for practical usage

Model efficiency and large motion scenarios. As shown in Table 1, simply scaling down model size significantly drops the performance on large motion scenarios. We investigate that our efficient model is robust to large motion scenarios compared to the kernel-free method at the same cost. We manually extract the largest motion 104 image pairs from RealBlur-J [27] to construct the large motion blur test set. We measure PSNR and SSIM for model performance while MACs is measured for model efficiency. As shown in Table 1, our efficient model, i.e., SegDeblur-S, improves PSNR in the large motion set (28.27 → 29.12 dB) and total set of RealBlur-J (31.99 → 32.53 dB), whose performance is also comparable to that of the larger model, NAFNet-64. This explains the necessity of our blur segmentation map particularly when training an efficient model.

Comparison to other prior-based methods. Since our method is viewed as a prior-based deblurring method, we compare it with other prior-based deblurring methods such as UFPNet [11] and MSDI-Net [19]. UFPNet is based on NAFNet [6] and produces the non-uniform blur kernel as prior information. MSDI-Net is built upon HiNet [5] and provides the degradation representation as prior information. Although such prior information is beneficial, the computational costs to estimate them and fuse them are highly expensive as shown in Table 2, which is insufficient for practical usage. In contrast, our method only requires 4 GMACs to generate a blur segmentation map for prior information. As a result, our SegDeblur-L (62 GMACs) achieves 33.51 dB which is even better than 33.35 dB in UFPNet (243 GMACs). Despite the most cost-effective method for generating prior information, our blur segmentation map has the greatest impact on improving deblurring performance compared to the other methods.

Methods	GMACs	RealBlur-J		RealBlur-R	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑
MPRNet [39]	777.01	31.76	0.922	39.31	0.972
MIMO-UNet+ [8]	154.41	31.92	0.919	-	-
FMIMO-UNet [24]	80.21	32.65	0.931	40.01	0.972
Stripformer [34]	169.89	32.48	0.929	39.84	0.974
MAXIM-3S [35]	169.50	32.84	0.935	39.45	0.962
FFTFormer [21]	131.45	32.62	0.932	40.20	0.973
GRL-B [21]	1285.28	32.82	0.932	40.20	0.974
NAFNet-64 [6]	63.64	32.50	0.928	39.89	0.973
SegDeblur-S (ours)	14.44	32.53	0.927	39.75	0.973
SegDeblur-L (ours)	62.68	32.95	0.934	40.21	0.975
SegDeblur-L* (ours)	62.68	**33.51**	**0.938**	**40.79**	**0.976**

Table 3. Comparison results on RealBlur-J [27] and RealBlur-R [27] datasets. "*" denotes that the method uses the test-time local converter (TLC) [9]. The best results are indicated in bold.

4.3. Comparison to kernel-free methods

Results on RealBlur. As discussed in Section 4.2, our blur segmentation map aids in improving deblurring performance under an efficient model. In this section, we confirm whether our blur segmentation map is still beneficial in larger networks. To this end, we train and evaluate with RealBlur-J and -R dataset [27]. We measure PSNR and SSIM for model performance while MACs is measured for

Methods	GMACs	RSBlur	
		PSNR ↑	SSIM ↑
SRN-Deblur [32]	1434.82	32.53	0.840
MIMO-UNet+ [8]	154.41	33.37	0.856
MPRNet [39]	777.01	33.61	0.861
Restormer [40]	141.00	33.69	0.863
Uformer-B [38]	89.50	33.98	0.866
NAFNet-64 [6]	63.64	33.97	0.866
SegDeblur-S (ours)	14.44	33.96	0.865
SegDeblur-L (ours)	62.68	34.21	0.870
SegDeblur-L* (ours)	62.68	**34.63**	**0.876**

Table 4. Comparison results on RSBlur [28] dataset. "*" denotes that the method uses the test-time local converter (TLC) [9]. The best results are indicated in bold.

Methods	GMACs	ReLoBlur	
		PSNR ↑	SSIM ↑
SRN-Deblur [32]	1434.82	34.30	0.923
DeblurGAN-v2 [18]	411.34	33.85	0.902
HiNet [5]	170.71	34.36	0.915
MIMO-UNet [8]	154.41	34.52	0.925
LBAG+ [20]	154.44	34.85	0.925
NAFNet-64 [6]	63.64	34.55	0.926
SegDeblur-S (ours)	14.44	34.86	0.923
SegDeblur-L (ours)	62.68	**35.34**	**0.927**

Table 5. Comparison results on ReLoBlur [20] dataset. The best results are indicated in bold.

model efficiency. The experimental results of RealBlur-J and -R are summarized in Table 3. The results show that our SegDeblur-L improves performance from 32.50 to 32.95 dB (RealBlur-J) and 39.89 to 40.21 (RealBlur-R).

Results on RSBlur. RSBlur [28] enables us to conduct a comprehensive evaluation of deblurring models since it contains realistic high-resolution images around 1920×1200, large motion scenarios, and various blur types such as object and camera motions. To this end, we train and evaluate with RSBlur. As shown in Table 4, our SegDeblur-L achieves the best performance against other methods. Furthermore, it is noticeable that the performance of our efficient model, i.e., SegDeblur-S (14 GMACs), is comparable to that of the best-performing methods such as NAFNet-64 (64 GMACs) and Uformer-B (89 GMACs). As illustrated in Fig. 5, our efficient model shows better qualitative results compared to the other efficient deblurring models [6, 15].

Results on ReLoBlur. We experiment with ReLoBlur [20], which consists of various object motion blur images. To verify that our blur segmentation map is also helpful for object motion blur dataset, we train and evaluate with ReLoBlur. As shown in Table 5, our SegDeblur-S (14 GMACs) achieves the performance comparable to that of the best-performing method, LBAG+ (154.44 GMACs) while requiring only 10% of the computational cost.

Results on GoPro. We do experiments on GoPro [25] which is a synthetic dataset. To confirm that our method still works well on the transformer-based model, we instead

Methods	GMACs	GoPro	
		PSNR ↑	SSIM ↑
SRN-DeblurNet [32]	1434.82	30.26	0.932
DeblurGAN-v2 [18]	411.34	29.55	0.934
MPRNet [39]	777.01	32.66	0.959
MIMO-UNet+ [8]	154.41	32.45	0.957
HINet [5]	170.71	32.77	0.959
Restormer [40]	141.00	32.92	0.961
Uformer-B [38]	89.50	32.97	0.967
Stripformer [34]	169.89	33.08	0.962
MAXIM-3S [35]	169.50	32.86	0.961
NAFNet-64 [6]	63.64	33.69	0.966
FNAFNet-64 [24]	72.40	33.85	0.967
MSDI-Net[19]	336.43	33.28	0.964
UFPNet[11]	243.33	34.06	0.968
GRL-B[21]	1285.28	33.93	0.968
FFTformer[15]	131.45	34.14	0.968
SegFFTformer (ours)	135.81	**34.38**	**0.970**

Table 6. Comparison results on GoPro [25] dataset. We train FFTformer [15] with our blur segmentation map which is referred to as SegFFTformer. The best results are indicated in bold.

choose the baseline, FFTFormer [15] which also gives the best performance on this dataset. We train FFTformer with our blur segmentation map and demonstrate that our method improves performance from 34.14 to 34.38 dB only with additional 4 GMACs, as denoted in Table 6. Note that the performance of FFTFormer in our experiment is 34.14 dB, slightly below the reported performance of 34.21 dB.

4.4. Comparison to commercial applications

The goal of this experiment is to confirm whether our method achieves model performance and efficiency comparable to those of commercial applications such as EnhanceX in Samsung Galaxy S23 and Unblur in Google Pixel 8. To accelerate our SegDeblur-S, we modify some network architectures of our method (called SegDeblur-S+) because some operations are not supported by AI acceleration, which will be more discussed in Section C of Appendix. 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 without additional quantization. As shown in Fig. 6, our method visually produces compelling results on real-world blur images compared to those of commercial applications and the other accelerated model (NAFNet-32+). Furthermore, the on-chip execution times of ours is 2.35s while the execution times of Samsung EnhanceX (1.64s) and Google Unblur (2.03s) are measured. Note that there is greater potential for further accelerating ours if deployed on Neural Processing Unit (NPU).

4.5. Ablation study

Contribution by each module. In this section, we explore which module of our method has more impact on performance improvement. As described in Table 7, with no mod-

(i) Blur Segmentation Map		✓	✓	✓
(ii) Latent Sharp Loss			✓	✓
(iii) Logarithmic Fourier Transform				✓
PSNR (dB)	31.99	31.96	32.17	**32.53**

Table 7. Ablation study on our components. Experimental results show that all components are necessary and positively combined to provide the best performance as indicated in bold.

ule, it indicates the original NAFNet-32 [6]. When adding (i), it uses blur segmentation map but not considers latent sharp image and logarithmic fourier space. The method including (i) and (ii) mean SegDeblur-S but uses fourier space instead of logarithmic fourier space. The final method containing all modules denotes our SegDeblur-S. As shown in Table 7, we observe that adding the module such as (ii) or (iii) provides a meaningful performance improvement from 31.96 to 32.17 dB and from 32.17 to 32.53 dB, respectively. This explains the necessity of utilizing a latent sharp image and logarithmic fourier space. Furthermore, simply estimating blur segmentation map is not helpful since it may suffer from the ill-posed problem as discussed in Section 3.2 (see the results of the first and second columns).

Other prior information. Our classification model not only generates a blur segmentation map but also provides a latent sharp image and deconvolved image. As they are estimated by the sum of their image residual error and blur image, the latent quantization map (e.g., quantizing the image residual error in the latent sharp image) and our blur segmentation map can be seen as their discrete counterparts, i.e., latent sharp image - latent quantization map and deconvolved image - blur segmentation map. In essence, the motion-frequency property is used for generating the deconvolved image while the latent sharp image does not consider it as shown in Fig. 4. As a result, the deconvolved image yields a better subsequent deblurring performance (32.44 dB) than the latent sharp image (32.18 dB) as shown in Fig. 7. Similarly, the case of latent quantization map (32.27 dB) lags behind the performance compared with that of our blur segmentation map (32.53 dB). This accounts for the necessity of the motion-frequency properties when generating GT-like priors. Furthermore, the discretized representations such as latent quantization map and blur segmentation map lead to a slight performance improvement compared with the continuous counterparts such as latent sharp and deconvolved images, e.g., 32.18 $\rightarrow$ 32.27 dB and 32.44 $\rightarrow$ 32.53 dB as shown in Fig. 7.

The number of classes. We examine the effect of the number of classes R. We conduct experiments with $R = \{1, 2, 4, 8, 16, 32\}$. As shown in Fig. 8 (a), we observe that the performance of $R = 1$ (31.91 dB) closely approximates that of deblurring without prior information (NAFNet-32), measured at 31.99 dB. As the number of classes increases, it positively impacts the performance of the deblurring model. On the other hand, a sufficient number of classes, e.g., more than 8, produces a meaningful deblurring performance.

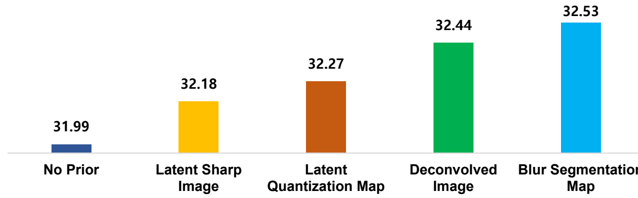


Figure 7. Ablation study on several prior information. Our blur segmentation map gives the most contribution to improving deblurring performance.

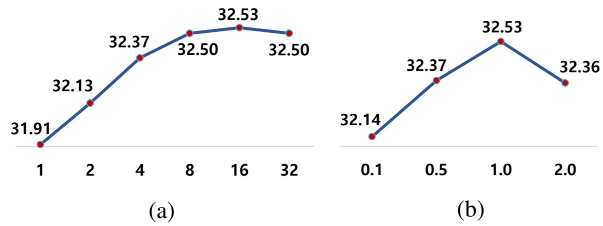


Figure 8. Ablation study on (a) # of classes R and (b) hyperparameter λ. The best results reside on $R = 16$ and $\lambda = 1.0$.

Effects on λ. We investigate which hyperparameter λ generates the most optimal blur segmentation map. We experiment with $\lambda = \{0.1, 0.5, 1.0, 2.0\}$. Given the blur segmentation map varying λ from 0.1 to 2.0, the final deblurring results are shown in Fig. 8 (b). The result shows that the constraint of the latent sharp image is significant for reducing ill-posedness as discussed in Section 3.2 and $\lambda = 1.0$ gives the best performance. Therefore, we choose the hyperparameter λ as 1.0 in our experiments. For more insights on the proposed method, we conduct more ablation studies in Section B of Appendix.

5. Conclusions

In this paper, we propose a new deblurring scheme that decomposes a deblurring regression problem into discretization and discrete-to-continuous conversion problems. We present a blur segmentation map that reflects the characteristics of the image residual error, which supports building an efficient model. We demonstrate the competitiveness of our efficient model when compared with the larger deblurring model and commercial applications. The proposed method can be extended to other deblurring tasks such as video deblurring [36] and defocus deblurring [1].

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST); No.2021-0-02068, Artificial Intelligence Innovation Hub).

References

[1] Abdullah Abuolaim and Michael S Brown. Defocus deblurring using dual-pixel data. *The European Conference on Computer Vision (ECCV)*, pages 111–126, 2020. 8

[2] E Oran Brigham and RE Morrow. The fast fourier transform. *IEEE spectrum*, 4(12):63–70, 1967. 4

[3] Guillermo Carbajal, Patricia Vitoria, Mauricio Delbracio, Pablo Musé, and José Lezama. Non-uniform blur kernel estimation via adaptive basis decomposition. *arXiv preprint arXiv:2102.01026*, 2021. 1, 2, 13

[4] Ayan Chakrabarti. A neural approach to blind motion deblurring. *The European Conference on Computer Vision (ECCV)*, pages 221–235, 2016. 1, 2

[5] Liangyu Chen, Xin Lu, Jie Zhang, Xiaojie Chu, and Chengpeng Chen. Hinet: Half instance normalization network for image restoration. *The IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 182–192, 2021. 6, 7

[6] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. *The European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 21

[7] Donald G Childers, David P Skinner, and Robert C Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428–1443, 1977. 3, 4

[8] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. *The IEEE International Conference on Computer Vision (ICCV)*, pages 4641–4650, 2021. 1, 3, 6, 7

[9] Xiaojie Chu, Liangyu Chen, Chengpeng Chen, and Xin Lu. Revisiting global statistics aggregation for improving image restoration. *The European Conference on Computer Vision (ECCV)*, 2022. 6, 7

[10] Jiangxin Dong, Stefan Roth, and Bernt Schiele. Deep wiener deconvolution: Wiener meets deep learning for image deblurring. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1048–1059, 2020. 2

[11] Zhenxuan Fang, Fangfang Wu, Weisheng Dong, Xin Li, Jinjian Wu, and Guangming Shi. Self-supervised non-uniform kernel estimation with flow-based motion prior for blind image deblurring. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18105–18114, 2023. 1, 3, 6, 7, 12, 19, 20

[12] DA Fish, AM Brinicombe, ER Pike, and JG Walker. Blind deconvolution by means of the richardson–lucy algorithm. *JOSA A*, 12(1):58–65, 1995. 1, 2

[13] Dong Gong, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton Van Den Hengel, and Qinfeng Shi. From motion blur to motion flow: A deep learning solution for removing heterogeneous motion blur. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2319–2328, 2017. 1, 2

[14] Michal Hradiš, Jan Kotera, Pavel Zemcık, and Filip Šroubek. Convolutional neural networks for direct text deblurring. *The British Machine Vision Conference (BMVC)*, 10(2), 2015. 3

[15] Lingshun Kong, Jiangxin Dong, Mingqiang Li, Jianjun Ge, and Jinshan Pan. Efficient frequency domain-based transformers for high-quality image deblurring, 2023. 1, 3, 4, 6, 7, 11, 14, 15, 16, 22, 23

[16] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. *Advances in neural information processing systems (NIPS)*, 22, 2009. 1, 2

[17] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8183–8192, 2018. 1

[18] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. *The IEEE International Conference on Computer Vision (ICCV)*, pages 8878–8887, 2019. 7

[19] Dasong Li, Yi Zhang, Ka Chun Cheung, Xiaogang Wang, Hongwei Qin, and Hongsheng Li. Learning degradation representations for. *The European Conference on Computer Vision (ECCV)*, pages 736–753, 2022. 1, 3, 6, 7

[20] Haoying Li, Ziran Zhang, Tingting Jiang, Peng Luo, Huajun Feng, and Zhihai Xu. Real-world deep local motion deblurring. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2023. 5, 6, 7, 11, 18

[21] Yawei Li, Yuchen Fan, Xiaoyu Xiang, Denis Demandolx, Rakesh Ranjan, Radu Timofte, and Luc Van Gool. Efficient and explicit modelling of image hierarchies for image restoration. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 3, 6, 7

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019. 6

[24] Xintian Mao, Yiming Liu, Fengze Liu, Qingli Li, Wei Shen, and Yan Wang. Intriguing findings of frequency selection for image deblurring. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2023. 1, 3, 6, 7, 11

[25] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3883–3891, 2017. 3, 5, 6, 7, 11, 12

[26] Seungjun Nah, Sanghyun Son, Jaerin Lee, and Kyoung Mu Lee. Clean images are hard to reblur: Exploiting the ill-posed inverse task for dynamic scene deblurring. *International Conference on Learning Representations (ICLR)*, 2022. 1

[27] Jaesung Rim, Haeyun Lee, Jucheol Won, and Sunghyun Cho. Real-world blur dataset for learning and benchmarking deblurring algorithms. *The European Conference on Computer Vision (ECCV)*, pages 184–201, 2020. 3, 5, 6, 7, 11, 12, 13, 17, 21

[28] Jaesung Rim, Geonung Kim, Jungeon Kim, Junyong Lee, Seungyong Lee, and Sunghyun Cho. Realistic blur synthesis for learning image deblurring. *The European Conference on Computer Vision (ECCV)*, 2022. 4, 5, 6, 7, 11, 12, 13, 20, 23

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation.

International Conference on Medical image computing and computer-assisted intervention, pages 234–241, 2015. 3, 5, 11

[30] Christian J Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Schölkopf. Learning to deblur. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1439–1451, 2015. 1, 2

[31] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 769–777, 2015. 2

[32] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8174–8182, 2018. 1, 3, 7, 12

[33] Phong Tran, Anh Tuan Tran, Quynh Phung, and Minh Hoai. Explore image deblurring via encoded blur kernel space. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11956–11965, 2021. 1

[34] Fu-Jen Tsai, Yan-Tsung Peng, Yen-Yu Lin, Chung-Chi Tsai, and Chia-Wen Lin. Stripformer: Strip transformer for fast image deblurring. *The European Conference on Computer Vision (ECCV)*, 2022. 1, 3, 6, 7

[35] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5769–5780, 2022. 1, 3, 6, 7, 19, 20

[36] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019. 8

[37] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[38] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17683–17693, 2022. 1, 3, 7

[39] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14821–14831, 2021. 3, 6, 7

[40] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5728–5739, 2022. 1, 3, 7

[41] Youjian Zhang, Chaoyue Wang, Stephen J Maybank, and Dacheng Tao. Exposure trajectory recovery from motion blur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7490–7504, 2021. 1, 2, 13