Explore Image Deblurring via Encoded Blur Kernel Space

Phong Tran[1]   Anh Tuan Tran[1,2]   Quynh Phung[1]   Minh Hoai[1,3]

[1]VinAI Research, Hanoi, Vietnam, [2]VinUniversity, Hanoi, Vietnam,
[3]Stony Brook University, Stony Brook, NY 11790, USA

{v.phongtt15,v.anhtt152,v.quynhpt29,v.hoainm}@vinai.io

Abstract

This paper introduces a method to encode the blur operators of an arbitrary dataset of sharp-blur image pairs into a blur kernel space. Assuming the encoded kernel space is close enough to in-the-wild blur operators, we propose an alternating optimization algorithm for blind image deblurring. It approximates an unseen blur operator by a kernel in the encoded space and searches for the corresponding sharp image. Unlike recent deep-learning-based methods, our system can handle unseen blur kernel, while avoiding using complicated handcrafted priors on the blur operator often found in classical methods. Due to the method's design, the encoded kernel space is fully differentiable, thus can be easily adopted in deep neural network models. Moreover, our method can be used for blur synthesis by transferring existing blur operators from a given dataset into a new domain. Finally, we provide experimental results to confirm the effectiveness of the proposed method. The code is available at https: //github.com/VinAIResearch/blur-kernel- space-exploring.

1. Introduction

Motion blur occurs due to camera shake or rapid movement of objects in a scene. Image deblurring is the task of removing the blur artifacts to improve the quality of the captured image. Image deblurring is an important task with many applications, especially during the current age of mobile devices and handheld cameras. Image deblurring, however, is still an unsolved problem, despite much research effort over the past decades.

Mathematically, the task of image deblurring is to recover the sharp image x given a blurry image y. One can assume the below mathematical model that relates x and y:

$$y = \hat{\mathcal{F}}(x, k) + \eta \approx \hat{\mathcal{F}}(x, k), \qquad (1)$$

where $\hat{\mathcal{F}}(\cdot, k)$ is the blur operator with the blur kernel k, and η is noise. In the simplest form, $\hat{\mathcal{F}}(\cdot, k)$ is assumed to
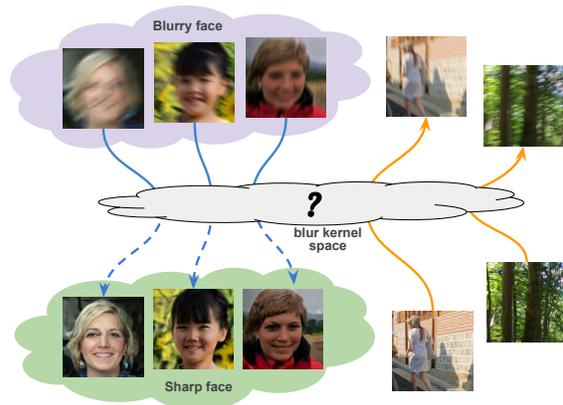
Figure 1. The space of blur kernels is the missing element for successful blur removal and synthesis. Previous image deblurring methods either overlooked the importance of this kernel space or made inadequate assumption about it. In this paper, we propose to learn this blur kernel space from a dataset of sharp-blurry image pairs (orange arrows) and leverage this encoded space for image deblurring (blue arrows).

be a convolution function with k being a convolution kernel and η being white Gaussian noise. Given a blurry image y, the deblurring task is to recover the sharp image x and optionally the blur operator $\hat{\mathcal{F}}(\cdot, k)$.

A popular approach to recover the sharp image is to use the Maximum A Posterior (MAP) estimate. That is to find x and k to maximize the posterior propbability $\mathbb{P}(x, k|y)$ assuming $\hat{\mathcal{F}}$ is known. This is equivalent to optimizing:

$$x, k = \underset{x,k}{\operatorname{argmax}} \, \mathbb{P}(y|x, k)\mathbb{P}(x)\mathbb{P}(k). \qquad (2)$$

However, this is an ill-posed problem and there are infinitely many pairs of (k, x) that lead to the same probability $\mathbb{P}(y|x, k)$, so the key aspect of the above MAP approach is to define proper models for the prior distributions $\mathbb{P}(x)$ and $\mathbb{P}(k)$. In fact, many deblurring methods focus on either designing handcrafted priors for x and k [2, 12, 20, 27] or learning the deep image prior [29, 37]. However, all of these works assume the blur operator is a convolutional operator,

and this assumption does not hold in practice. These MAP-based methods cannot handle complex in-the-wild blur operators and usually produce undesirable artifacts when testing on real-world blurry images.

An alternative approach is to directly learn a function that maps from a blurry image to the corresponding non-blurry image. This function can be a deep convolutional network and the parameters of the network can be learned using paired training data of blurry and non-blurry images [14, 15, 25, 36]. Unlike the MAP-based approach, this approach learns the inverse function of the blur operator directly without explicitly reasoning about the blur operator and the distribution of the blur kernel. Given the lack of an explicit representation for the space of the blur kernels, this approach does not generalize well beyond the set of individual blur kernels seen during training. This approach [14, 15, 25, 36] produces poor results when testing on blur operators that are not present in the training set. In our experiments, these deep-learning models degenerate to an identity map when testing on an out-of-domain blur operator; the recovered image is nearly identical to the input image. This is a known issue, and it is referred to as "the trivial solution" by traditional deblurring methods. The MAP-based methods tackle this problem by putting prior distributions on the sharp image and the blur kernel. However, those priors cannot be readily applied to the existing deep-learning models due to the lack of an explicit representation for the blur kernels.

In this paper, we propose to address the limitations of both aforementioned approaches as follows. First, we devise a deep-learning formulation with an explicit representation for the blur kernel and the blur operator. Second, we use a data-driven approach to learn the family of blur operators and the latent manifold of the blur kernels, instead of assuming that the blur operator is a convolutional operator as used in existing MAP-based methods. Specifically, we simultaneously learn a blur operator family $\mathcal{F}$ and a blur kernel extractor $\mathcal{G}$ such that:

$$y = \mathcal{F}(x, k) \quad \text{and} \quad k = \mathcal{G}(x, y). \tag{3}$$

Note in this paper, $\mathcal{F}$ is referred to as the blur operator *family*. For a specific blur kernel k, $\mathcal{F}(\cdot, k)$ is a specific blur operator from the family of blur operators. We call k the blur kernel of the blur operator $\mathcal{F}(\cdot, k)$. When the functional form of $\mathcal{F}$ is fixed, we will refer to a blur operator $\mathcal{F}(\cdot, k)$ by its blur kernel k if there is no confusion.

Once the blur operator family $\mathcal{F}$ has been learned, we can use it to deblur an input image y by finding x and k to satisfy the above equations using alternating optimization. Moreover, we can incorporate additional constraints on the solution space of x to generate more realistic results. For example, we can use a deep generative model to learn the manifold of natural images and constraint the solution space

to this manifold. The conceptual idea is illustrated in Fig. 1.

Our method can also be used for blur synthesis. This can be done by transferring the blur kernel of a sharp-blurry image pair to another image. Blur synthesis is useful in many ways. For example, we can transfer the real-world motion blur of an existing dataset [8] to another domain where it might be difficult to collect paired data. Blur synthesis can also be used for training data augmentation, improving the robustness of a downstream task such as face recognition or eye gaze estimation.

In short, the contributions of our paper are: (1) we propose a novel method to encode the blur kernel space for a dataset of blur-sharp image pairs, which can be used to deblur images that contain unseen blur operators; (2) we propose a novel blur synthesis method and demonstrate its utilities; and (3) we obtain state-of-the-art deblurring results on several datasets.

2. Related Work

2.1. Image deblurring

Image deblurring algorithms can be divided into two main categories: MAP-based and learning-based methods.

MAP-based blind image deblurring. In MAP-based methods, finding good priors for the sharp images and blur kernels ($\mathbb{P}(x)$ and $\mathbb{P}(k)$ in Eq. (2)) are two main focuses. For the sharp images, gradient-based prior is usually adopted since the gradient of natural images is highly sparse. In particular, Chan and Wong [2] proposed a total-variation (TV) penalty that encouraged the sparsity of the image gradient. Krishnan and Fergus [12] suggested that the image gradient followed Hyper-laplacian distribution. However, Levin et al. [17] showed that these gradient-based priors could favor blurry images over sharp ones and lead to the trivial solution, i.e., $x = y$ and k is the identity operator. Krishnan et al. [13] used $\ell 1/\ell 2$ regularization that gave sharp image the lowest penalty. Pan et al. [27] showed that the dark channel of a sharp image was usually sparser than the dark channel of the corresponding blurry image. Overall, these priors only model low-level statistics of images, which are neither adequate nor domain-invariant.

Recently, Ulyanov et al. [37] introduced Deep Image Prior (DIP) for image restoration tasks. A network G was learned so that each image I was represented by a fixed vector z such that $I = G_\theta(z)$. Ren et al. [29] proposed SelfDeblur method using two DIPs for x and k. Instead of using alternating optimization like other MAP-based methods, they jointly sought x and k using a gradient-based optimizer.

All aforementioned methods assumed the blur kernel was linear and uniform, i.e., it can be represented as a convolution kernel. However, this assumption is not true for real-world blur. Non-linear camera response functions can cause non-linear blur kernels while non-uniform

blur kernels appear when only a small part of the image moves. There were some attempts for non-uniform deblurring [3, 24, 32, 39], but they still assumed the blur was locally uniform, and they were not very practical given the high computational cost.

Learning-based deblurring. Many deep deblurring models have been proposed over the past few years. Nah et al. [25] proposed a multi-scale network for end-to-end image deblurring. It deblurred an image in three scale levels; the result from the lower level was used as an input of its upper level. Similarly, Tao et al. [36] employed a scale-recurrent structure for image deblurring. GAN [5] was first used for image deblurring in [14], whereas a high-quality image was generated conditioned on the blurry input image. Kupyn et al. [15] introduced DeblurGANv2, which used Feature Dynamic Networks [19] to extract image features and two discriminators for global and patch levels. DeblurGANv2 achieved impressive run-time while maintaining reasonable results on common benchmarks. There were also works on multi-frame deblurring [35, 38, 43] and domain-specific deblurring [7, 18, 29, 33, 34, 40, 41].

Unfortunately, deep-learning models do not perform well for cross-domain tasks. For example, models trained on the REDS dataset [26] perform poorly on GOPRO [25], despite the visual similarity between the two datasets. As a result, deep deblurring models have not been used in real-world applications. This kernel overfitting phenomenon has not been explained in prior works.

2.2. GAN-inversion image restoration

Image manifolds generated by GANs [5] were used to approximate the solution space for image restoration problem in recent works [23, 28]. They sought an image in the manifold such that its degradation version was the closest to the provided low-quality image. The benefits of this method are twofold. First, this method guarantees a sharp and realistic outcome. Meanwhile, image restoration is ill-posed with multiple solutions, and the common image restoration methods often yield a blurry result towards the average of all possible solutions [23]. Second, in the case of blind deblurring, this method bypasses the kernel overfitting issue in deep image restoration models.

Existing works in this direction, however, just cover simple known degradations such as bicubic downsampling. To handle the challenging in-the-wild motion-blur degradation, we first need to model the family of blur operators.

2.3. Blur synthesis

To train deep deblurring models, large-scale and high-quality datasets are needed. But it is hard to capture pairs of corresponding sharp and blurry images in real life, so blur synthesis has been widely used. Assuming uniform blur (i.e., a convolutional blur kernel), a common approach

is to synthesize the trajectory of the blur kernel and apply this synthetic kernel on the sharp image set. Chakrabarti [1] generated blur trajectories by randomly sampling six points on a grid and connected those points by a spline. Schuler et al. [31] sampled blur trajectories by a Gaussian process. These methods could only synthesize uniform blur and they did not take the scene structure into account. Therefore, synthesized blurry images are unrealistic.

More sophisticated blur synthesis algorithms rely on the blur generation process in the camera model. In particular, an image in color space can be modeled as: $I = g\left(\frac{1}{T}\int_0^T S(t)dt\right)$, where $S(t)$ is the sensor signal at time t, T is the exposure time, and g the camera response function. Nah et al. [25] approximated g by the gamma function $g(x) = x^{\frac{1}{\gamma}}$. They converted a frame I to its corresponding signal sensor $g^{-1}(I)$, averaged consucutive frames in that signal domain, then converted it back to the color space. The REDS dataset [26] was synthesized similarly but with an increased video temporal resolution and a more sophisticated camera response function.

To reduce the gap between synthetic and real-world blur, Jaesung Rim and Cho [8] proposed a real-world blur dataset that was captured by two identical cameras with different shutter speeds. However, the data collection process was complicated, requiring elaborate setup with customized hardware.

3. Methodology

In this section, we first describe a method to learn the blur operator family $\mathcal{F}$ that explains the blurs between paired data of sharp-blurry images. We will then explain how the blur operator family can be used for removing or synthesizing blur.

3.1. Learning the blur operator family

Given a training set of n data pairs $\{(x_i, y_i)\}_{i=1}^n$, our goal is to learn a blur operator family that models the blur between the sharp image x_i and the corresponding blurry image y_i for all i's. Each pair is associated with a latent blur kernel k_i; and the blurry image y_i is obtained by applying the blur operator family on the sharp image x_i with the blur kernel k_i as parameters, i.e., $y_i = \mathcal{F}(x_i, k_i)$. Traditional MAP-based methods often assume $\mathcal{F}(\cdot, k_i)$ to be the convolutional operator and k_i a convolutional kernel, but this assumption does not hold for real blurs in the wild.

Learning $\mathcal{F}$ is challenging because $\{k_i\}$ are latent variables. Fortunately, each k_i is specific to a sharp-blurry image pair, so we can assume k_i can be recovered by a kernel extractor function $\mathcal{G}$, i.e., $k_i = \mathcal{G}(x_i, y_i)$. We can learn both the blur operator family $\mathcal{F}$ and the kernel extractor $\mathcal{G}$ by minizing the differences between the synthesized blurry image $\mathcal{F}(x_i, \mathcal{G}(x_i, y_i))$ and the actual blurry image y_i. In
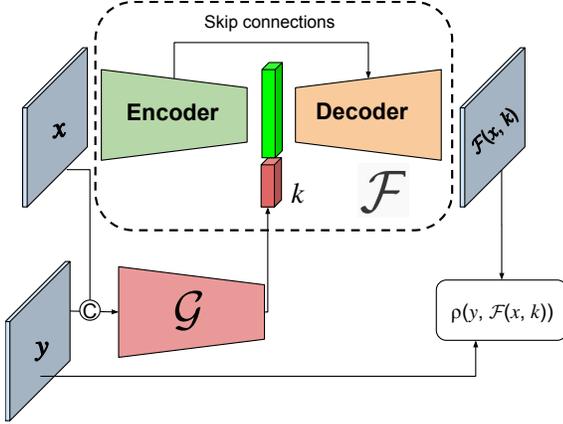
Figure 2. **Roles of the blur operator family $\mathcal{F}$ and the blur kernel extractor $\mathcal{G}$ and their architectures**. $\mathcal{G}$ can be used to extract the blur kernel k, while $\mathcal{F}$ can be used to generate a blurry image given the blur kernel k. $\mathcal{F}$ is an encoder-decoder network with skip connection, while $\mathcal{G}$ is a residual network.

this paper, we implement them by two neural networks, an encoder-decoder with skip connection [30] for $\mathcal{F}$ and a residual network [6] for $\mathcal{G}$. Both $\mathcal{F}$ and $\mathcal{G}$ are fully differentiable, and they can be jointly optimized by minimizing the following loss function:

$$\sum_{i=1}^{n} \rho(y_i, \mathcal{F}(x_i, \mathcal{G}(x_i, y_i))), \qquad (4)$$

where $\rho(\cdot)$ is the Charbonnier loss [16] measuring the distance between the "fake" blurry image $\mathcal{F}(x_i, \mathcal{G}(x_i, y_i))$ and the corresponding real blurry image y_i.

This procedure is illustrated in Fig. 2. First, we sample (x, y) from a dataset of image pairs. Second, we fit the concatenation of these images into $\mathcal{G}$ to generate the corresponding encoded blur kernel vector k. Third, with x and k as the input, we use $\mathcal{F}$ to create the synthesized blurry image. $\mathcal{F}$ encodes x into a bottle-neck embedding vector, concatenates that embedding vector with k, and decodes it to get the synthesized blurry image. Details of the architecture choices and hyper-parameters tuning are given in the supplementary materials.

3.2. Blind image deblurring

Once the blur operator family $\mathcal{F}$ has been learned, we can use it for image deblurring. Given a blurry image y, our task is to recover the sharp image x. We pose it as the optimization problem, where we seek to recover both the sharp image x and the blur kernel k to minimize $\rho(y, \mathcal{F}(x, k))$. To optimize $\rho(y, \mathcal{F}(x, k))$, we propose an iterative optimization procedure that alternates between the following two steps: (A) fix the blur kernel k and optimize the latent sharp image x, and (B) fix x and optimize for k.

Algorithm 1 Blind image deblurring
_____
Input: blurry image y
Output: sharp image x
1: Sample $z_x \sim \mathcal{N}(0, I)$
2: Randomly initialize θ_x of $G^x_{\theta_x}$
3: **while** θ_x has not converged **do**
4:      Sample $z_k \sim \mathcal{N}(0, I)$
5:      Randomly initialize θ_k of $G^k_{\theta_k}$
6:      **while** θ_k has not converged **do**
7:          $g_k \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_k$
8:          $\theta_k \leftarrow \theta_k + \alpha * ADAM(\theta_k, g_k)$
9:      **end while**
10:     $g_x \leftarrow \partial \mathcal{L}(\theta_x, \theta_k)/\partial \theta_x$
11:     $\theta_x \leftarrow \theta_x + \alpha * ADAM(\theta_x, g_x)$
12: **end while**
13: $x = G^x_{\theta_x}(z_x)$
_____

To stablize the optimization process and to obtain better deblurring results, we propose to add a couple of regularization terms into the objective function and reparameterize both x and k with Deep Image Prior (DIP) [37] as follows. First, we propose to add a regularization term on the L_2 norm of the kernel k to stablize the optimization process and avoid the trivial solution. Second, we propose to use the Hyper-Laplacian prior [12] on the image gradients of $\mathbf{x}$ to encourage the sparsity of the gradients, reducing noise and creating more natural looking image x. This corresponds to adding the regularization term: $(g_u^2(x) + g_v^2(x))^{\alpha/2}$ into the objective function, where g_u and g_v are the horizontal and vertical derivative operators respectively. Adding the regularization terms leads to the updated objective:

$$\rho(y, \mathcal{F}(x, k)) + \lambda||k||_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}, \qquad (5)$$

where λ, γ, α are tunable hyper-parameters.

Finally, inspired by the success of Deep Image Prior [37] for zero-shot image restoration [4, 21, 29, 37], we propose to reparameterize both x and k by neural networks. In particular, instead of optimizing x directly, we take x as the stochastic output of a neural network $G^x_{\theta_x}$ and we optimize the parameters θ_x of the network instead. Specifically, we define $x = G^x_{\theta_x}(z_x)$, where z_x is standard normal random vector, i.e., $z_x \sim \mathcal{N}(0, I)$. Similarly, we reparameterize $k = G^k_{\theta_k}(z_k)$. The final objective function for deblurring is:

$$\mathcal{L}(\theta_x, \theta_k) = \rho(y, \mathcal{F}(x, k)) + \lambda||k||_2 + \gamma(g_u^2(x) + g_v^2(x))^{\alpha/2}$$
$$\text{where } x = G^x_{\theta_x}(z_x), z_x \sim \mathcal{N}(0, I), \qquad (6)$$
$$k = G^k_{\theta_k}(z_k), z_k \sim \mathcal{N}(0, I). \qquad (7)$$

This objective function can be optimized using Algorithm 1.

3.3. Approximated manifold of natural images

In Sec. 3.2, we propose a general solution for image deblurring, where little assumption is made about the space of the sharp image x. We use DIP to reparameterize x as the output of a neural network with schotastic input, and we optimize the parameter of the network instead. However, in many situations, the domain of the sharp image x is simpler, e.g., being a face or a car. In this situation, we can have better reparameterization for x, taking into account the learned manifold for the specific domain of x.

In this paper, we also consider the image manifold proposed by Menon et al. [23]. We reparameterize x by $G_{style}(z)$ in which G_{style} is the pretrained StyleGAN [11], z is optimized along the sphere $\sqrt{d}S^{d-1}$ using spherical projected gradient descent [23].

3.4. Blur synthesis using blur transferring

There exist datasets of paired images with "close-to-real" blurs, such as REDS [26], GOPRO [25], or real-world blur [8]. But the collection of these datasets required elaborate setups, expensive hardware (e.g., high-speed camera), and enormous effort. Unfortunately, similar datasets do not exist for many application domains (e.g., faces and scene text), and it is difficult or even impossible to replicate these laboratory setups to collect data for in-the-wild environments (e.g., street scenes).

To this end, a benefit of our approach is the ability to transfer the motion blurs from an existing dataset to a new set of images. In particular, given a dataset with pairs of sharp-blurry images, we can first train $\mathcal{F}$ and $\mathcal{G}$ as described in Sec. 3.1. To transfer the motion blur between the image pair (x, y) to a new image $\hat{x}$, we can simply compute: $\hat{y} := \mathcal{F}(\hat{x}, \mathcal{G}(x, y))$.

4. Experiments

We perform extensive experiments to verify the effectiveness of our blur kernel encoding method. We also provide results for image deblurring and blur synthesis. All the experiments are conducted on a single NVidia V100 GPU. Image deblurring experiments are cross-domain. In particular, all data-driven methods are trained on the REDS dataset [26] and tested on the GOPRO dataset [25].

REDS dataset [26] comprises 300 high-quality videos with various scenes. The videos are captured at 120fps. The corresponding blurry videos are synthesized by upsampling the frame rate and averaging the neighboring frames. We use this dataset to train our kernel extractor as well as deep deblurring models.

GOPRO dataset [25] consists 3142 sharp-blur pair of frames. Those frames are captured at 240fps. The synthesis process is similar to REDS dataset, except for the choice of the camera response function. We use this dataset to test the deblurring methods.

Levin dataset [17] is generated using eight convolution kernels with different sizes. Here we use its kernels to synthesize uniform blur on other datasets.

FFHQ dataset [11] is a human face dataset. This dataset consists of 70,000 high-quality 1024×1024 images with various genders, ethics, background, and accessories. This dataset was used to train the StyleGAN model.

CelebA-HQ dataset [10] is a human face dataset that consists of 30,000 images at 1024×1024 resolution. Its images were selected from the CelebA dataset [22], but the quality was improved using some preprocessing steps such as JPEG removal and $4\times$ super-resolution.

4.1. Blur kernel extractor

This section verifies if our blur kernel extractor can accurately extract and transfer blur from a sharp-blurry image pair to another image. We use the known explicit kernels from the Levin dataset to synthesize blurry images in training and testing for experiments with ground-truth labels. As for experiments on datasets without explicit blur kernels, such as REDs and GOPRO, we check the stability of the deblurring networks trained on internal blur-swapped data.

4.1.1 Testing blur kernel encoding on Levin dataset

Suppose we have a ground-truth blur operator family $\hat{\mathcal{F}}$. We train $\mathcal{F}$ and $\mathcal{G}$ using a sharp-blur pair dataset generated by $\hat{\mathcal{F}}$. Then we can measure the performance of the blur kernel extractor by calculating the distance between $\mathcal{F}(x, \mathcal{G}(x, y))$ and $\hat{\mathcal{F}}(x, h)$ for arbitrary pair (x, h) and $y = \hat{\mathcal{F}}(x, h)$.

In this experiment, we let $\hat{\mathcal{F}}(\cdot, h)$ be a convolutional operator whose kernel is one of the eight used in the Levin dataset [17]. To generate training data, we randomly select 5000 sharp images from the REDS dataset [26] and generate 5000 corresponding blurry images using the mentioned kernels. Then we use these 5000 pairs to learn $\mathcal{F}$ and $\mathcal{G}$. To create testing data, we randomly sample two other disjointed image sets S and T for the source and target sharp images in blur transfer. Each set consists of 500 sharp images from GOPRO dataset [25]. Then for each testing kernel k, we generate the blur images in the source set $y_k = \hat{\mathcal{F}}(x, k) = k * x$, apply blur from (x, y_k) to each $\hat{x} \in T$ via the trained $\mathcal{F}$ and $\mathcal{G}$, and compute the average PSNR score.

$$\frac{\sum_{x \in S, \hat{x} \in T} PSNR(\mathcal{F}(\hat{x}, \mathcal{G}(x, y_k)), \hat{\mathcal{F}}(\hat{x}, k))}{|S| \times |T|}. \quad (8)$$

We report the test results in Table 1. Our method achieves very high PSNR scores, demonstrating its ability to extract and transfer the blur kernels.

Blur	SelfDeblur [29]	DeblurGANv2 [15]	SRN-Deblur [36]	Ours	Sharp
0.489	0.630	0.442	0.448	0.348	
0.630	0.857	0.663	0.633	0.601	
0.717	0.780	0.707	0.694	0.664	
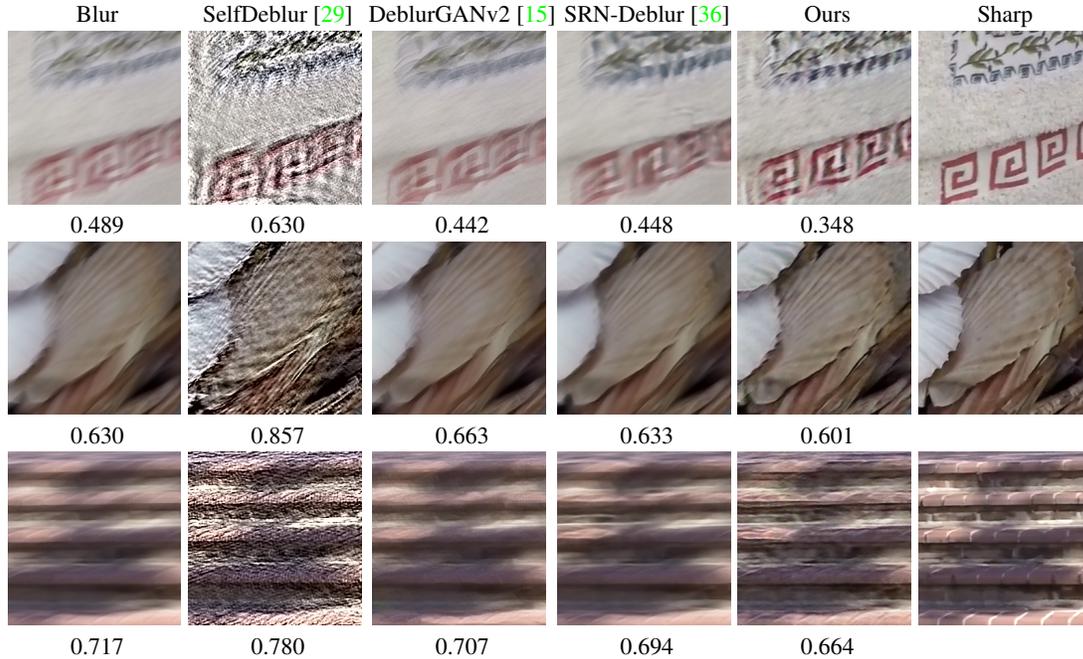
Figure 3. Results of deblurring methods trained on REDS and tested on GOPRO, and their LPIPS score [42] (lower is better).

Blur	SelfDeblur [29]	[15] REDS	[15] imgaug	[36] REDS	[36] imgaug	Ours
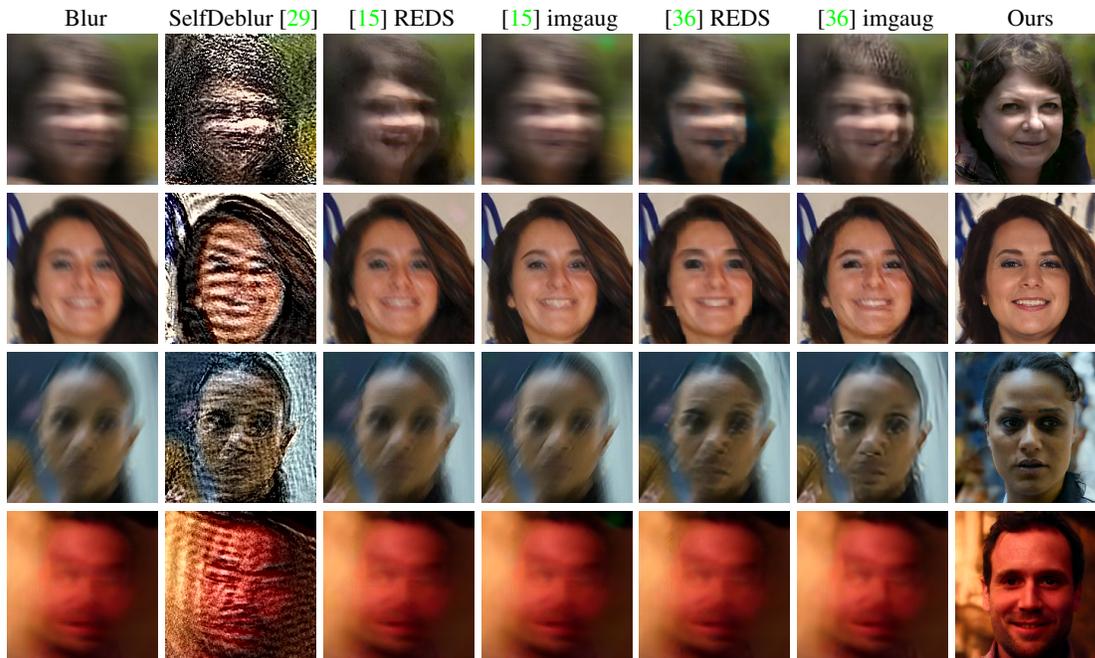
Figure 4. Qualitative results of deblurring methods. Here DeblurGANv2 REDS is the model trained with face dataset using REDS kernel, while DeblurGANv2 imgaug is the model trained with face dataset using imgaug. The blurry image in the first and second rows are synthesized using blur transferring technique in Sec. 7 and *imgaug* [9] respectively. The last two rows are in-the-wild blurry images that we randomly collect on the Internet.

4.1.2 Training on synthetic datasets

For a sharp-blur dataset without explicit blur kernels, we can randomly swap the blur operator between its pairs using our method. To be more specific, for each sharp-blur pair (x, y) and a random sharp image $\hat{x}$ from this dataset, we generate the blurry image $\hat{y}$ using the blur kernel extracted from (x, y). Then we use this synthetic dataset to train a deep deblurring model and compare its performance to the one trained on the original dataset. In this experiment, we

choose SRN-Deblur [36], a typical deep image deblurring method. The testing datasets are REDS and GOPRO.

The performance of deblurring networks, measured by the average PSNR score on test sets, is reported in Table 2. PSNR scores when training on blur-swapped datasets are comparable to the ones obtained when training on the original dataset.

4.2. General blind image deblurring

4.2.1 Qualitative results

We now evaluate our blind image deblurring method, described in Sec. 3.2, and compare it to other methods in a cross domain setting. We use the state-of-the-art deep-learning-based methods, including DeblurGANv2 [15], SRN-Deblur [36], and a recent kernel-based algorithm called SelfDeblur [29]. We train all the methods using REDS dataset [26] and test them on GOPRO dataset [25].

Some visualization results and their corresponding LPIPS scores [42] are shown in Fig. 3. The methods based on deep neural networks [15, 36] produce results that are very similar to the input. On the other hand, the predicted images of SelfDeblur [29] are noisy with many artifacts. Our method consistently generates sharp and visually pleasing results.

4.2.2 Retrieving unseen kernel

Our algorithm is based on the assumption that an unseen blur operator can be well approximated using the encoded blur kernel space. Here we conduct an experiment to verify this assumption. We use $\mathcal{F}$ and $\mathcal{G}$ that are trained on one dataset, either REDS or GOPRO, to retrieve unseen blur operator of each sharp-blur image pair in the testing subset of the same or different dataset using step (B) in Sec. 3.2. To evaluate the accuracy of that extracted blur, we compute PSNR score between the reconstructed and original blurry images. The average PSNR score for each configuration is reported in Table 3. As can be seen, the quality of kernels extracted in cross-domain setting is similar to the ones in same-domain configuration. It shows that our method is effective in handling unseen blur.

Fig. 5 visualizes some results when training on REDS and testing on GOPRO. Our reconstructed blurry images are close to the original ones, indicating the high quality of the extracted kernels.

	kernel 1	kernel 2	kernel 3	kernel 4
PSNR (db)	49.48	51.93	52.06	53.74
	kernel 5	kernel 6	kernel 7	kernel 8
PSNR (db)	49.91	49.49	51.43	50.38

Table 1. Results of our blur kernel extraction on Levin dataset

Training data	Dataset	
	REDS	GOPRO
Original	30.70	30.20
Blur-swapped	29.43	28.49

Table 2. Results of SRN-Deblur trained [36] on the original and blur-swapped datasets.

sharp   original blur   retrieved blur

Figure 5. Retrieving unseen kernel. The first column shows the sharp images from the GOPRO dataset, the second column shows their corresponding blurry images. In the last row, we approximate the blur operators using the kernels from REDS dataset and apply it to the sharp images.

Tranining set	Test set	
	REDS4	GOPRO
REDS	34.35	30.67
GOPRO	31.38	35.13

Table 3. Results of our method in retrieving unseen blur kernel with same and cross-domain configs.

4.3. Using an approximated natural image manifold

4.3.1 Qualitative results

As discussed in Sec. 3.3, we can incorporate a GAN-based image manifold as the sharp image prior to attain realistic deblurring results. Following [23], we conduct face deblurring experiments using the StyleGAN model pretrained on the FFHQ dataset to approximate the natural facial image manifold. We use both synthesized and in-the-wild blurry images for testing. As for synthetic data, we use images from CelebHQ dataset [10]. The blur synthesis techniques include motion-blur augmentation from the *imgaug* (the second row in Fig. 4) tool [9] and the blur transferred from the GOPRO dataset (the first row in Fig. 4). As for in-the-wild images, we search for blurry faces from the Internet (the last two rows in Fig. 4). Each deep model is trained using FFHQ dataset [11] with blur operators are synthesized by *imgaug* or blur kernels transferred from GOPRO dataset [25]. As for our method, we use the blur extractor trained

on REDS dataset in Sec. 4.2.2. All the test blurs, therefore, are unseen to our method.

We compare our deblurring results and different baseline methods in Fig. 4. As can be seen, the deep deblurring models [15, 36] fail to produce sharp outcomes, particularly on blur. The state-of-the-art MAP-based algorithm [29] generates unrealistic and noisy images. In contrast, our method can successfully approximate realistic sharp face outputs in all test cases.

4.3.2 Loss convergence

One may think that the good deblurring results in the previous experiment are purely due to restricting the sharp image solution space to a GAN manifold. Yes, but the blur kernel prior is equally important; without a good blur kernel prior, the method would fail to converge to desirable results. To prove it, we analyze the optimization processes on a specific deblurring example with different blur kernel manifolds: (1) the traditional convolution kernels with DIP used in SelfDeblur [29], (2) the bicubic downsampling kernel used in PULSE [23], and (3) our encoded kernel. The results are shown in Fig. 6. The first two methods failed to converge since the real blur operator is neither linear nor uniform. In contrast, the method using our kernel method quickly converges to a realistic face.
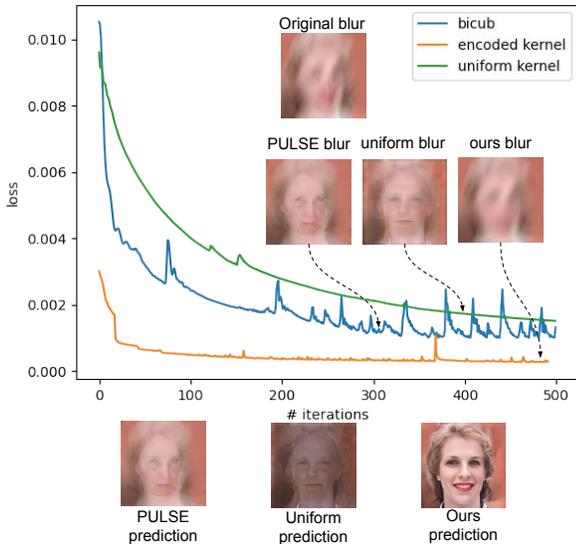
Figure 6. Loss convergence of the method in Sec. 3.3 when using different kernel priors.

4.4. Blur synthesis

Our blur transfer method is effective in synthesizing new blurry images. In Fig. 7, we transfer the blur operator from the source sharp-blur pair (x, y) (the two middle columns) to the target sharp image $\hat{x}$ (the first column) to synthesize its corresponding blurry image $\hat{y}$. We see that the content
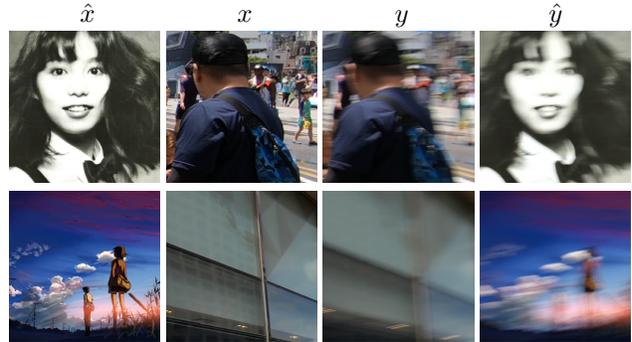
Figure 7. Transferring blur kernel from the source pair x, y to the target sharp $\hat{x}$ to generate the target blurry image $\hat{y}$.

of $\hat{x}$ is fully preserved in $\hat{y}$, and the blur in $\hat{y}$ looks similar to the blur in y. Our method can also work with any type of images, such as grayscale images (the first row) or animation images (the second row).

One application of this blur synthesis is data augmentation. We experiment with the use of this augmentation technique to improve image deblurring. In particular, we use FFHQ dataset [11] to synthesize three sharp-blur datasets with different types of blur kernels: (1) common motion-blur kernels generated by imgaug tool [9], (2) our encoded REDS kernels, and (3) our encoded GOPRO kernels. The first dataset is the traditional deblurring dataset. The second dataset can be considered as data augmentation, and the last dataset is used for unseen blur testing. We train SRN-Deblur models [36] in two scenarios: using only the first dataset or using the combination of the first two datasets. Testing results are reported in Table 4. The network trained on the combined data is more stable and performs better in the unseen blur scenario.

| Tranining kernels | Test kernels | | |
	imgaug	REDS	GOPRO
imgaug	**28.64**	24.22	22.96
comb.	28.30	**28.37**	**23.92**

Table 4. Effect of blur augmentation in improving SRN-Deblur [36] model, tested on the synthetic FFHQ datasets.

5. Conclusion

In this paper, we have proposed a method to encode the blur kernel space of an arbitrary dataset of sharp-blur image pairs and leverage this encoded space to solve some specific tasks such as image deblurring and blur synthesis. For image deblurring, we have shown that our method can handle unseen blur operators. For blur synthesis, our method can transfer blurs from a given dataset of sharp-blur image pairs into any domain of interest, including domains of facial, grayscale, and animated images.

References

[1] Ayan Chakrabarti. A neural approach to blind motion deblurring. In *Proceedings of the European Conference on Computer Vision*, 2016.

[2] Tony F Chan and Chiu-Kwong Wong. Total variation blind deconvolution. *IEEE transactions on Image Processing*, 7 (3):370–375, 1998.

[3] Sunghyun Cho, Yasuyuki Matsushita, and Seungyong Lee. Removing non-uniform motion blur from images. In *Proceedings of the International Conference on Computer Vision*, 2007.

[4] Yossi Gandelsman, Assaf Shocher, and Michal Irani. double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[6] 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*, 2016.

[7] Michal Hradivs, Jan Kotera, Pavel Zemcik, and Filip vSroubek. Convolutional neural networks for direct text deblurring. In *Proceedings of the British Machine Vision Conference*, 2015.

[8] Jucheol Won Jaesung Rim, Haeyun Lee and Sunghyun Cho. Real-world blur dataset for learning and benchmarking deblurring algorithms. In *Proceedings of the European Conference on Computer Vision*, 2020.

[9] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, Franccois-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. Imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[11] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[12] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *Advances in Neural Information Processing Systems*, 2009.

[13] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[14] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jivr´i Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[15] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the International Conference on Computer Vision*, 2019.

[16] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[17] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[18] Songnan Lin, Jiawei Zhang, Jinshan Pan, Yicun Liu, Yongtian Wang, Jing SJ Chen, and Jimmy Ren. Learning to deblur face images via sketch synthesis. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2020.

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[20] Guangcan Liu, Shiyu Chang, and Yi Ma. Blind image deblurring using spectral properties of convolution operators. *IEEE Transactions on image processing*, 23(12):5047–5056, 2014.

[21] Jiaming Liu, Yu Sun, Xiaojian Xu, and Ulugbek S Kamilov. Image restoration using total variation regularized deep image prior. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the International Conference on Computer Vision*, 2015.

[23] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[24] James G Nagy and Dianne P O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998.

[25] 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*, 2017.

[26] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[27] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[28] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep genera-

tive prior for versatile image restoration and manipulation. *arXiv preprint arXiv:2003.13659*, 2020.

[29] Dongwei Ren, Kai Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural blind deconvolution using deep priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[31] 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.

[32] Qi Shan, Wei Xiong, and Jiaya Jia. Rotational motion deblurring of a rigid object from a single image. In *Proceedings of the International Conference on Computer Vision*, 2007.

[33] Ziyi Shen, Wei-Sheng Lai, Tingfa Xu, Jan Kautz, and Ming-Hsuan Yang. Deep semantic face deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[34] Yibing Song, Jiawei Zhang, Lijun Gong, Shengfeng He, Linchao Bao, Jinshan Pan, Qingxiong Yang, and Ming-Hsuan Yang. Joint face hallucination and deblurring via structure generation and detail enhancement. *International Journal of Computer Vision*, 127(6-7):785–800, 2019.

[35] 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*, 2017.

[36] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[38] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[39] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. *International journal of computer vision*, 98(2):168–186, 2012.

[40] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. Learning to super-resolve blurry face and text images. In *Proceedings of the International Conference on Computer Vision*, 2017.

[41] Rajeev Yasarla, Federico Perazzi, and Vishal M Patel. Deblurring face images using uncertainty guided multi-stream semantic networks. *IEEE Transactions on Image Processing*, 29:6251–6263, 2020.

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[43] Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Haozhe Xie, Wangmeng Zuo, and Jimmy Ren. Spatio-temporal filter adaptive network for video deblurring. In *Proceedings of the International Conference on Computer Vision*, 2019.