Structured Kernel Estimation for Photon-Limited Deconvolution

Yash Sanghvi, Zhiyuan Mao, Stanley H. Chan
School of Electrical and Computer Engineering, Purdue University
{ysanghvi, mao114, stanchan}@purdue.edu

Abstract

Images taken in a low light condition with the presence of camera shake suffer from motion blur and photon shot noise. While state-of-the-art image restoration networks show promising results, they are largely limited to well-illuminated scenes and their performance drops significantly when photon shot noise is strong.

In this paper, we propose a new blur estimation technique customized for photon-limited conditions. The proposed method employs a gradient-based backpropagation method to estimate the blur kernel. By modeling the blur kernel using a low-dimensional representation with the key points on the motion trajectory, we significantly reduce the search space and improve the regularity of the kernel estimation problem. When plugged into an iterative framework, our novel low-dimensional representation provides improved kernel estimates and hence significantly better deconvolution performance when compared to end-to-end trained neural networks. The source code and pretrained mdoels are available at https://github.com/sanghviyashiitb/structured-kernel-cvpr23

1. Introduction

Photon-Limited Blind Deconvolution: This paper studies the photon-limited blind deconvolution problem. Blind deconvolution refers to simultaneously recovering both the blur kernel and latent clean image from a blurred image and "photon-limited" refers to presence of photon-shot noise in images taken in low-illumination / short exposure. The corresponding forward model is as follows:

$$\mathbf{y} = \text{Poisson}(\alpha \mathbf{h} \circledast \mathbf{x}). \quad (1)$$

In this equation, $\mathbf{y} \in \mathbb{R}^N$ is the blurred-noisy image, $\mathbf{x} \in \mathbb{R}^N$ is the latent clean image, and $\mathbf{h} \in \mathbb{R}^M$ is the blur kernel. We assume that $\mathbf{x}$ is normalized to $[0, 1]$ and the entries of $\mathbf{h}$ are non-negative and sum up to 1. The constant α represents the average number of photons per pixel and is inversely proportional to the amount of Poisson noise.

Deep Iterative Kernel Estimation: Blind image deconvolution has been studied for decades with many successful algorithms including the latest deep neural networks [8, 24, 34, 42, 43]. Arguably, the adaptation from the traditional Gaussian noise model to the photon-limited Poisson noise model can be done by retraining the existing networks with appropriate data. However, the restoration is not guaranteed to perform well because the end-to-end networks seldom explicitly take the forward image formation model into account.

Recently, people have started to recognize the importance of blur kernel estimation for photon-limited conditions. One of these works is by Sanghvi et. al [30], where they propose an iterative kernel estimation method to backpropagate the gradient of an unsupervised reblurring function, hence to update the *blur kernel*. However, as we can see in Figure 1, their performance is still limited when the photon shot noise is strong.

Structured Kernel Estimation: Inspired by [30], we believe that the iterative kernel estimation process and the unsupervised reblurring loss are useful. However, instead of searching for the kernel directly (which can easily lead to local minima because the search space is too big), we propose to search in a low-dimensional space by imposing structure to the motion blur kernel.

To construct such a low-dimensional space, we frame the blur kernel in terms of trajectory of the camera motion. Motion trajectory is often a continuous but irregular path in the two-dimensional plane. To specify the trajectory, we introduce the concept of *key point estimation* where we identify a set of anchor points of the kernel. By interpolating the path along these anchor points, we can then reproduce the kernel. Since the number of anchor points is significantly lower than the number of pixels in a kernel, we can reduce the dimensionality of the kernel estimation problem.

The key contribution of this paper is as follows: We propose a new kernel estimation method called *Kernel Trajectory Network (KTN)*. KTN models the blur kernel in a low-dimensional and differentiable space by specifying key points of the motion trajectory. Plugging this low-dimensional representation in an iterative framework im-

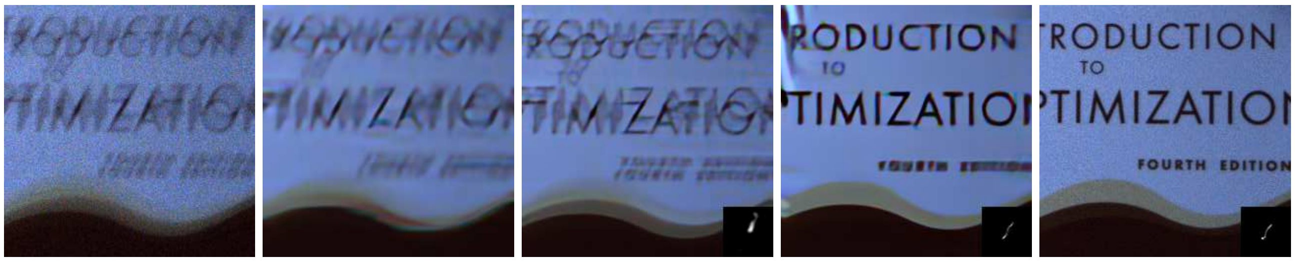


Figure 1. **The proposed Kernel Trajectory Network (KTN) on real noisy blurred image from Photon-Limited Deblurring Dataset (PLDD) [29]** The result corresponding to MPR-Net was generated by retraining the network with GoPro dataset [24] corrupted by Poisson noise. The inset images for "Sanghvi et. al" and "Ours" represent the estimated kernel and the inset image for "Ground-Truth" represents the kernel captured using a point source, as provided in PLDD.

proves the regularity of the kernel estimation problem. This leads to substantially better blur kernel estimates in photon-limited regimes where existing methods fail.

2. Related Work

Traditional Blind Deconvolution: Classical approaches to the (noiseless) blind deconvolution problem [6, 7, 22, 32, 40] use a joint optimization framework in which both the kernel and image are updated in an alternating fashion in order to minimize a cost function with kernel and image priors. For high noise regimes, a combination of ℓ_1+TV prior has been used in [2]. Levin et. al [16] pointed out that this joint optimization framework for the blind deconvolution problem favours the no-blur degenerate solution i.e. $(\mathbf{x}^*, \mathbf{h}^*) = (\mathbf{y}, \mathbf{I})$ where $\mathbf{I}$ is the identity operator. Some methods model the blur kernel in terms of the camera trajectory and then recover both the trajectory and the clean image using optimization [12, 38, 39] and supervised-learning techniques [11, 33, 46].

For the non-blind case, i.e., when the blur kernel is assumed to be known, the Poisson deconvolution problem has been studied for decades starting from Richardson-Lucy algorithm [21, 26]. More contemporary methods include Plug-and-Play [28, 29], PURE-LET [18], and MAP-based optimization methods [10, 13].

Deep Learning Methods. Recent years, many deep learning-based methods [5, 31] have been proposed for the blind image deblurring task. The most common strategy is to train a network end-to-end on large-scale datasets, such as the GoPro [24] and the RealBlur [27] datasets. Notably, many recent works [8, 24, 34, 42, 43] improve the performance of deblurring networks by adopting the multi-scale strategies, where the training follows a coarse-to-fine setting that resembles the iterative approach. Generative Adversarial Network (GAN) based deblurring methods [3, 14, 15, 44] are also shown to produce visually appealing images. Zamir et al. [41] and Wang et al. [37] adapt the popular vision transformers to the image restoration problems and demonstrate competitive performance on the deblurring task.

Neural Networks and Iterative Methods: While neural networks have shown state-of-the-art performance on the deblurring task, another class of methods incorporating iterative methods with deep learning have shown promising results. Algorithm unrolling [23], where an iterative method is unrolled for fixed iterations and trained end-to-end has been applied to image deblurring [1, 19]. In SelfDeblur [25], authors use Deep-Image-Prior [35] to represent the image and blur kernel and obtain state-of-the-art blind deconvolution performance.

3. Method

3.1. Kernel as Structured Motion Estimation

Camera motion blur can be modeled as a latent clean image $\mathbf{x}$ convolved with a blur kernel $\mathbf{h}$. If we assume the blur kernel lies in a window of size 32×32, then $\mathbf{h} \in \mathbb{R}^{1024}$. However, in this high dimensional space, only few entries of the blur kernel $\mathbf{h}$ are non-zero. Additionally, the kernel is generated from a two-dimensional trajectory which suggests that a simple sparsity prior is not sufficient. Given the difficulty of the photon-limited deconvolution problem, we need to impose a stronger prior on the kernel. To this end, we propose a *differentiable and low-dimensional representation* of the blur kernel, which we will use as the search space in our kernel estimation algorithm.

We take the two-dimensional trajectory of the camera during the exposure time and divide it into K "key points". Each key point represents either the start, the end or a change in direction of the camera trajectory as seen in Figure 2. Given the K key points as points mapped out in x-y space, we can interpolate them using cubic splines to form a continuous trajectory in 2D. To convert this continuous trajectory to an equivalent blur kernel, we assume a point source image and move it through the given trajectory. The

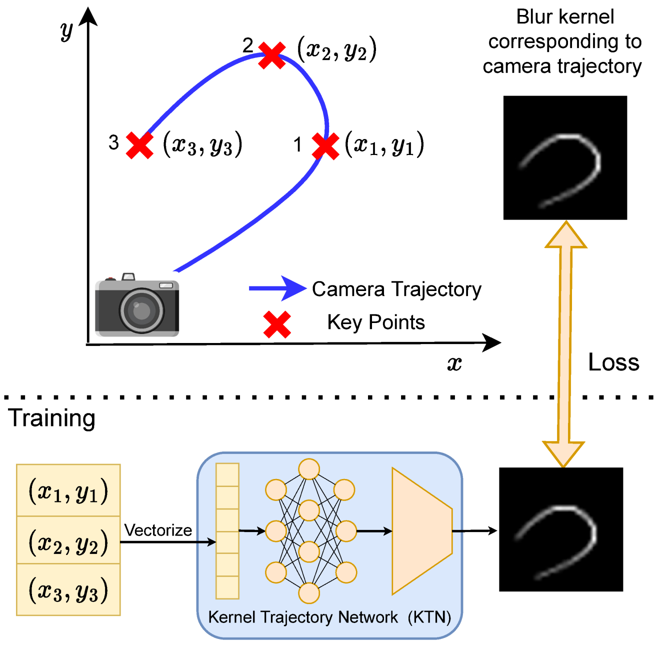


Figure 2. **Blur as Structured Motion Estimation**: In our formulation, we view the blur kernel as the continuous camera trajectory reduced to K key points, as shown in top half of the figure. We learn a differentiable representation from the vectorized K key points to a blur kernel using a neural network. This *lower dimensional and differentiable representation* is leveraged to estimate a better blur kernel and avoiding local minima during inference.

resulting frames are then averaged to give the corresponding blur kernel as shown in Figure 2.

Given the formulation of blur kernel $\mathbf{h}$ in terms of K key points, we now need to put this representation to a differentiable form since we intend to use it in an iterative scheme. To achieve this, we learn the transformations from the key points to the blur kernels using a neural network, which will be referred to as *Kernel-Trajectory Network (KTN)*, and represent it using a differentiable function $T(.)$. Why differentiability is important to us will become clear to the reader in the next subsection.

To train the Kernel-Trajectory Network, we generate training data as follows. First, for a fixed K, we get K key points by starting from $(0, 0)$ and choosing the next $K-1$ points by successively adding a random vector with a uniformly chosen random direction i.e. $U[0, 360]$ and uniformly chosen length from from $U[0, 100/(K-1)]$. Next, the set of key points are converted to a continuous smooth trajectory using bicubic interpolation. Then, we move a point source image through the given trajectory using the `warpPerspective` function in OpenCV, and average the resulting frames.

Using the process defined above, we generate 60,000 blur kernels and their corresponding key point representations. For the Kernel-Trajectory Network $T(.)$, we take a U-Net like network with the first half replaced by 3-fully connected layers and train it with the generated data using ℓ_2-loss. For further architectural details on the KTN, we refer the reader to the supplementary document.

3.2. Proposed Iterative Scheme

We described in the previous subsection how to obtain a low-dimensional and differentiable representation $T(.)$ for the blur kernel and now we are ready to present the full iterative scheme in detail. The proposed iterative scheme can be divided into three stages which are summarized as follows. We first generate an initial estimate of the direction and magnitude of the blur. This is used as initialization for a gradient-based scheme in Stage I which searches the appropriate kernel representation in the latent space $\mathbf{z}$. This is followed by Stage II where we fine-tune the kernel obtained from Stage I using a similar process.

Initialization Before starting the iterative scheme, we need a light-weight initialization method. This is important because of multiple local minima in the kernel estimation process.

We choose to initialize the method with a rectilinear motion kernel, parameterized by length ρ and orientation θ. To determine the length and orientation of the kernel, we use a minor variation of the kernel estimation in PolyBlur [9]. In this variation, the "blur-only image" $G(\mathbf{y})$ is used as the input and ρ, θ for the initial kernel are estimated using the minimum of the directional gradients. We refer the reader to Section II in the supplementary document for further details on the initialization. Explanation on the "blur-only image" is provided when we describe Stage I of the scheme.

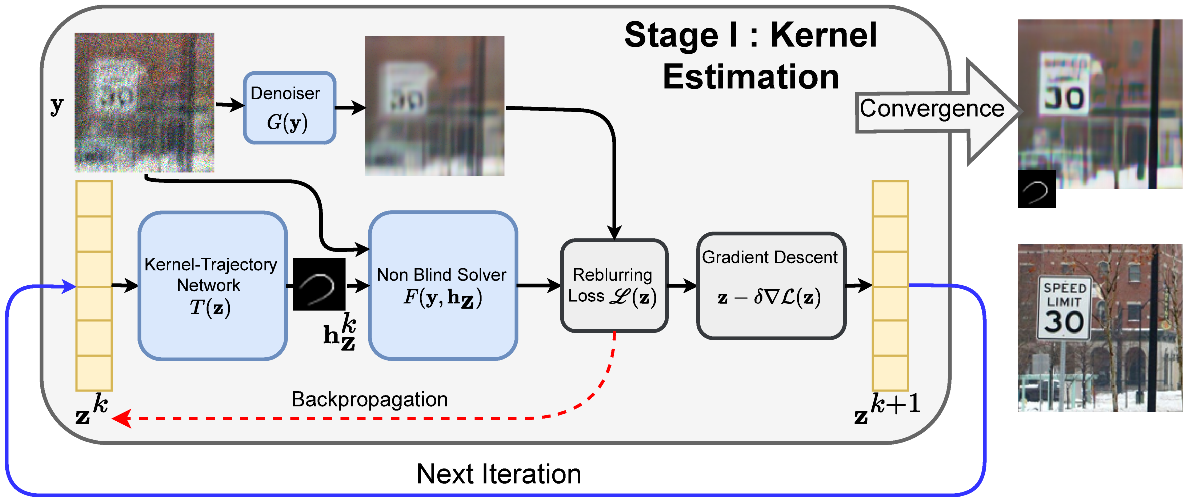
Stage I: Kernel Estimation in Latent Space Given an initial kernel, we choose initial latent $\mathbf{z}^0$ by dividing the rectilinear kernel into K key points. Following the framework in [30], we run a gradient descent based scheme which optimizes the following cost function:

$$\mathcal{L}(\mathbf{z}) \overset{\text{def}}{=} \underbrace{\|G(\mathbf{y}) - \mathbf{h}_\mathbf{z} \circledast F(\mathbf{y}, \mathbf{h}_\mathbf{z}))\|_2^2}_{\text{Reblurring Loss}}, \tag{2}$$

where $\mathbf{h}_\mathbf{z} \overset{\text{def}}{=} T(\mathbf{z})$ represents the kernel output from Kernel-Trajectory network $T(.)$ given the vectorized key points representation $\mathbf{z}$. $F(.)$ represents the Poisson non-blind deconvolution solver which takes both noisy-blurred image and a blur kernel as the input. $G(\mathbf{y})$ represents a denoiser which is trained to remove *only the noise from noisy-blurred image*. The overall cost function represents reblurring loss i.e. how well the kernel estimate and corresponding image estimate $\mathbf{h}_\mathbf{z} \circledast F(\mathbf{y}, \mathbf{h}_\mathbf{z})$ match the blur-only image $G(\mathbf{y})$.

To minimize the cost function in (2), we use a simple gradient descent based iterative update for $\mathbf{z}$ as follows:

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \delta \underbrace{\nabla_\mathbf{z} \mathcal{L}(\mathbf{z}^k)}_{\text{backpropagation}} \tag{3}$$



Stage I : Kernel Estimation
$\mathbf{y}$
Denoiser $G(\mathbf{y})$
Convergence
Kernel-Trajectory Network $T(\mathbf{z})$
$\mathbf{h}_{\mathbf{z}}^{k}$
Non Blind Solver $F(\mathbf{y},\mathbf{h}_{\mathbf{z}})$
Reblurring Loss $\mathscr{L}(\mathbf{z})$
Gradient Descent $\mathbf{z}-\delta\nabla\mathcal{L}(\mathbf{z})$
$\mathbf{z}^{k}$
Backpropagation
$\mathbf{z}^{k+1}$
Next Iteration

Figure 3. Flowchart describing first stage of the proposed scheme. We estimate the motion kernel of the blurry noisy image in lower dimensional latent space $\mathbf{z}$ where the blur kernel is represented by $T(\mathbf{z})$ and by minimizing the reblurring loss $\mathcal{L}$ as defined in equation 2

where $\delta > 0$ is the step size and $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}^k)$ represents the gradient of the cost function $\mathcal{L}$ with respect to $\mathbf{z}$ evaluated $\mathbf{z}^k$. It should be noted that the cost function is evaluated using the non-blind solver $F(.)$ and Kernel-Trajectory Network $T(.)$ - two neural network computations. Therefore, we can compute the gradient $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}^k)$ using auto-differentiation tools provided in `PyTorch` by backpropagating the gradients through $F(.)$ and then $T(.)$

Stage II: Kernel Fine-tuning In the second stage, using the kernel estimate of Stage I, we fine-tune the kernel by "opening up" the search space to the entire kernel vector instead of parametrizing by $T(.)$. Specifically, we optimize the following loss function

$$\mathcal{L}(\mathbf{h}) \overset{\text{def}}{=} \|G(\mathbf{y}) - \mathbf{h} \circledast F(\mathbf{y},\mathbf{h}))\|_2^2 + \gamma\|\mathbf{h}\|_1. \quad (4)$$

Note the presence of the second term which acts as an ℓ_1-norm sparsity prior. Also the kernel vector $\mathbf{h}$ is being optimized instead of the latent key point vector $\mathbf{z}$. Using variable splitting as used in Half-Quadratic Splitting (HQS), we convert the optimization problem in (4) to as follows:

$$\mathcal{L}(\mathbf{h},\mathbf{v}) = \|G(\mathbf{y}) - \mathbf{h} \circledast F(\mathbf{y},\mathbf{h}))\|_2^2 + \gamma\|\mathbf{h}\|_1 + \frac{\mu}{2}\|\mathbf{h}-\mathbf{v}\|_2^2 \quad (5)$$

for some hyperparameter $\mu > 0$. This leads us to the following iterative updates

$$\mathbf{h}^{k+1} = \mathbf{h}^k - \delta \cdot \left\{\nabla_{\mathbf{h}}\mathcal{L}(\mathbf{h}^k) + \mu(\mathbf{h}^k - \mathbf{v}^k)\right\}, \quad (6)$$

$$\mathbf{v}^{k+1} = \max\left(\left|\mathbf{h}^{k+1}\right| - \gamma/\mu, 0\right) \cdot \text{sign}(\mathbf{h}^{k+1}) \overset{\text{def}}{=} \mathcal{S}_{\gamma/\mu}(\mathbf{h}^{k+1}). \quad (7)$$

Algorithm 1 Iterative Poisson Deconvolution Scheme

```
1:  Input: Noisy-blurry $\mathbf{y}$, Photon-Level $\alpha$, denoiser $G(\cdot)$, non-blind solver $F(\cdot)$, Kernel-Trajectory-Network $T(\cdot)$.
2:  Initialize $\mathbf{z}^0$ using method described in Algorithm 1 in supplementary
3:  for $k = 0, 1, 2, \cdots$ do % Stage I begins here
4:      $\mathbf{h}_{\mathbf{z}}^k \leftarrow T(\mathbf{z}^k)$
5:      $\mathcal{L}(\mathbf{z}) \leftarrow \|G(\mathbf{y}) - \mathbf{h}_{\mathbf{z}}^k \circledast F(\mathbf{y}, \mathbf{h}_{\mathbf{z}}^k)\|_2^2$
6:      Calculate $\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}^k)$ using automatic differentiation
7:      $\mathbf{z}^{k+1} \leftarrow \mathbf{z}^k - \delta\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{z}^k)$
8:  end for
9:  $\mathbf{h}^0, \mathbf{v}^0 \leftarrow T(\mathbf{z}^\infty)$, $\mu \leftarrow 2.0$, $\gamma \leftarrow 10^{-4}$
10: for $k = 0, 1, 2, \cdots$ do % Stage II begins here
11:     $\mathcal{L}(\mathbf{h}) \leftarrow \|G(\mathbf{y}) - \mathbf{h} \circledast F(\mathbf{y}, \mathbf{h})\|_2^2$
12:     Calculate $\nabla_{\mathbf{h}}\mathcal{L}(\mathbf{h}^k)$ using automatic differentiation
13:     $\mathbf{h}^{k+1} \leftarrow \mathbf{h}^k - \delta\left(\nabla_{\mathbf{h}}\mathcal{L}(\mathbf{h}^k) + \mu(\mathbf{h}^k - \mathbf{v}^k)\right)$
14:     $\mathbf{v}^{k+1} \leftarrow \mathcal{S}_{\gamma/\mu}(\mathbf{h}^{k+1})$
15:     $\mu \leftarrow 1.01\mu$
16: end for
17: return $\mathbf{h}^{(\infty)}$ and $\mathbf{x}^{(\infty)} = F(\mathbf{y}, \mathbf{h}^{(\infty)})$
```

4. Experiments

4.1. Training

While our overall method is not end-to-end trained, it contains pre-trained components - the non-blind solver $F(.)$ and denoiser $G(.)$. The architectures of $F(.)$ and $G(.)$ are inherited from PhD-Net [29] which takes as input a noisy-blurred image and kernel. However, for the denoiser $G(.)$, we fix the kernel input to identity operator since it is trained to remove only the noise from noisy-blurred image $\mathbf{y}$.

$F(.)$ and $G(.)$ are trained using synthetic data as follows. We take clean images from Flickr2K dataset [20] and the blur kernels from the code in Boracchi and Foi [4]. The blurred images are also corrupted using Poisson shot noise with photon levels α uniformly sampled from $[1, 60]$. The non-blind solver $F(.)$ is trained using kernel and noisy blurred image as input, and clean image as the target. The denoiser $G(.)$ is trained with similar procedure but with blurred-noisy image as the only input blur-only images as the target. The training processes, along with other experiments described in this paper are implemented using `PyTorch` on a NVIDIA Titan Xp GPU.

For quantitative comparison of the method presented, we retrain the following state-of-the-art networks for Poisson Noise: *Scale Recurrent Network (SRN)* [34], *Deep-Deblur* [24], *DHMPN* [43], *MPR-Net* [42], and *MIMO-UNet+* [8]. We perform this retraining in the following two different ways. First, we use synthetic data training as described for $F(.)$ and $G(.)$. Second, for testing on realistic blur, we retrain the networks using the GoPro dataset [24] as it is often used to train neural networks in contemporary deblurring literature. We add the Poisson noise with the same distribution as the synthetic datasets to the blurred images. While retraining the networks, we use the respective loss functions from the original papers for sake of a fair comparison.

4.2. Quantitative Comparison

We quantitatively evaluate the proposed method on three different datasets, and compare it with state-of-the-art deblurring methods. In addition to the end-to-end trained methods described previously, we also compare our approach to the following Poisson deblurring methods: Poisson-Plug-and-Play [28], and PURE-LET [18]. Even though these methods assume the blur kernel to be known, we include them in the quantitative comparison since they are specifically designed for Poisson noise. For all of the methods described above, we compare the restored image's quality using PSNR, SSIM, and Learned Perceptual Image Patch Similarity (LPIPS-Alex, LPIPS-VGG) [45]. We include the latter as another metric in our evaluation since failure of MSE/SSIM to assess image quality has been well documented in [36, 45]

BSD100: First, we evaluate our method on synthetic blur as follows. We collect 100 random images from the BSD-500 dataset, blur them synthetically with motion kernels from the Levin dataset [17] followed by adding Poisson noise at photon-levels $\alpha = 10, 20$, and 40. The results of the quantitative evaluation are provided in Table 1. Since the blur is synthetic, ground-truth kernel is known and hence, can be used to simultaneously evaluate Poisson non-blind deblurring methods i.e, Poisson Plug-and-Play, PURE-LET, and PhD-Net. The last method is the non-blind solver $F(.)$ and serves as an upper bound on the deconvolution performance.

Levin Dataset: Next, we evaluate our method on the Levin dataset [17] which contains 32 real blurred images along with the ground truth kernels, as measured through a point source. We evaluate our method on this dataset with addition of Poisson noise at photon levels $\alpha = 10, 20$ and 40 and the results are shown in Table 2. For a fair comparison, end-to-end trained methods are retrained using synthetically blurred data (as described in Section IV-A) for evaluation on BSD100 and Levin dataset.

RealBlur-J [27]: To demonstrate that our method is able to handle realistic blur, we evaluate our performance on randomly selected 50 patches of size 256×256 from the Real-Blur-J [27] dataset. Note that we reduce the size of the tested image because our method is based on a single-blur convolutional model. Such model may not be applicable for a large image with spatially varying blur and local motion of objects. However, for a smaller patch of a larger image, the single-blur-kernel model of deconvolution is a much more valid assumption.

To ensure a fair comparison, we evaluate end-to-end networks by retraining on both the synthetic and GoPro dataset. As shown in Table 3, we find that end-to-end networks perform consistently better on the RealBlur dataset when trained using the GoPro dataset instead of synthetic blur. This can be explained by the fact both GoPro and RealBlur have realistic blur which is not necessarily captured by a single blur convolutional model.

4.3. Qualitative Comparison

Color Reconstruction We show reconstructions on examples from the real-blur dataset in Figure 4. While our method is grayscale, we perform colour reconstruction by estimating the kernel from the luminance-channel. Given the estimated kernel, we deblur each channel of the image using the non-blind solver and then combine the different channels into a single RGB-image. *Note that all qualitative examples in this paper for end-to-end trained networks are trained using the GoPro dataset*, since they provide the better visual result.

Photon-Limited Deblurring Dataset We also show qualitative examples from photon-limited deblurring dataset [29] which contains 30 images' raw sensor data, blurred by camera shake and taken in extremely low-illumination. For reconstructing these images, we take the average of the R, G, B channels of the Bayer patter image, average it and then reconstruct it using the given method. The qualitative results for this dataset can be found in Figure 5. We also show the estimated kernels, along with estimated kernels from [30, 40], in Figure 6.

However, instead of using the reblurring loss directly, we find the scheme is more numerically stable if we take the gradients of the image first and then estimate the reblurring

Method Photon Level, Metric		SRN [34]	DHMPN [43]	Deep-Deblur [24]	MIMO-UNet+ [8]	MPRNet [42]	Ours	P4IP [28]	PURE-LET [18]	PhD-Net [29]
$\alpha = 10$	PSNR ↑	20.71	20.89	21.17	21.04	21.09	**21.57**	19.26	22.49	23.00
	SSIM ↑	0.386	0.391	0.401	0.356	0.393	**0.471**	0.348	0.485	0.500
	LPIPS-Alex ↓	0.681	0.702	0.656	0.733	0.678	**0.560**	0.733	0.588	0.544
	LPIPS-VGG ↓	0.646	0.652	0.627	0.683	0.641	**0.587**	0.674	0.607	0.567
$\alpha = 20$	PSNR ↑	20.79	21.03	21.30	21.36	21.25	**21.93**	19.45	22.94	23.63
	SSIM ↑	0.392	0.401	0.410	0.396	0.405	**0.483**	0.353	0.516	0.540
	LPIPS-Alex ↓	0.683	0.688	0.666	0.660	0.667	**0.542**	0.726	0.526	0.500
	LPIPS-VGG ↓	0.639	0.640	0.621	0.663	0.631	**0.578**	0.668	0.584	0.539
$\alpha = 40$	PSNR ↑	20.89	21.15	21.43	**21.63**	21.41	21.62	20.18	23.48	24.38
	SSIM ↑	0.409	0.418	0.425	0.441	0.428	**0.527**	0.372	0.561	0.593
	LPIPS-Alex ↓	0.677	0.673	0.673	0.586	0.647	**0.488**	0.706	0.467	0.446
	LPIPS-VGG ↓	0.629	0.626	0.612	0.639	0.613	**0.549**	0.660	0.557	0.503
Blind?		✓	✓	✓	✓	✓	✓	×	×	×
End-To-End Trained?		✓	✓	✓	✓	✓	×	×	×	✓

Table 1. **Performance on BSD100 Dataset with Synthetic Blur**. ↑ represents metrics where higher means better and vice versa for ↓. LPIPS-Alex and LPIPS-VGG represent the perceptual measures from [45]. The best performing blind deconvolution method for each metric and photon level is shown in **bold**. The non-blind deconvolution methods are shown for reference in grey columns.

Method Photon Level, Metric		SRN [34]	DHMPN [43]	Deep-Deblur [24]	MIMO-UNet+ [8]	MPRNet [42]	Ours	P4IP [28]	PURE-LET [18]	PhD-Net [29]
$\alpha = 10$	PSNR ↑	20.26	20.50	20.93	21.25	21.04	**22.01**	19.92	21.63	22.41
	SSIM ↑	0.510	0.509	0.524	0.516	0.533	**0.611**	0.463	0.590	0.638
	LPIPS-Alex ↓	0.507	0.521	0.496	0.594	0.479	**0.340**	0.546	0.371	0.341
	LPIPS-VGG ↓	0.531	0.526	0.518	0.661	0.511	**0.477**	0.555	0.522	0.466
$\alpha = 20$	PSNR ↑	20.49	20.39	21.11	21.64	21.33	**22.72**	19.53	21.79	22.78
	SSIM ↑	0.523	0.521	0.536	0.554	0.551	**0.641**	0.442	0.607	0.667
	LPIPS-Alex ↓	0.496	0.502	0.492	0.485	0.459	**0.304**	0.533	0.339	0.304
	LPIPS-VGG ↓	0.515	0.514	0.501	0.610	0.493	**0.448**	0.554	0.510	0.447
$\alpha = 40$	PSNR ↑	20.59	20.50	21.20	21.88	21.54	**22.32**	17.32	21.78	22.96
	SSIM ↑	0.535	0.532	0.545	0.583	0.567	**0.647**	0.362	0.614	0.687
	LPIPS-Alex ↓	0.491	0.494	0.494	0.428	0.447	**0.273**	0.487	0.324	0.263
	LPIPS-VGG ↓	0.506	0.506	0.493	0.557	0.479	**0.444**	0.560	0.507	0.432

Table 2. **Performance on Levin dataset with realistic camera shake blur** [16]. The best performing blind deconvolution method for each metric and photon level is shown in **bold** and non-blind deconvolution methods are shown for reference in grey columns.

loss. This can be explained by the fact that unlike simulated data, the photon level is not known exactly and is estimated using the sensor data itself by a simple heuristic. For further details on how to use the sensor data, we refer the reader to [29].

4.4. Ablation Study

In Table 4, we provide an ablation study by running the scheme for different number of key points i.e. $K = 4, 6$, and 8 and without KTN ($K = 0$) on RealBlur dataset. Through this study, we demonstrate the effect of the Kernel Trajectory Network has on the iterative scheme. As expected, changing the search space for kernel estimation improves the performance significantly across all metrics. Increasing the number of key points used for representing kernels also steadily improves the performance of the scheme, which can be explained by the fact there are larger degrees of freedom.

5. Conclusion

In this paper, we use an iterative framework for the photon-limited blind deconvolution problem. More specifically, we use a non-blind solver which can deconvolve Poisson corrupted and blurred images given a blur kernel. To mitigate ill-posedness of the kernel estimation in such

Method →		SRN [34]		DHMPN [43]		Deep-Deblur [24]		MIMO-UNet+ [8]		MPRNet [42]		**Ours**
Training → Photon lvl, Metric		Synth.	GoPro	Synth.	GoPro	Synth.	GoPro	Synth.	GoPro	Synth.	GoPro	Synth.
$\alpha = 10$	PSNR ↑	25.72	27.64	25.72	27.58	25.98	27.57	26.20	26.78	26.26	**28.16**	26.61
	SSIM ↑	0.612	0.706	0.603	0.696	0.577	0.719	0.531	0.571	0.641	0.729	**0.738**
	LPIPS-Alex ↓	0.438	0.310	0.454	0.329	0.441	0.297	0.484	0.396	0.401	0.288	**0.277**
	LPIPS-VGG ↓	0.508	0.454	0.509	0.472	0.496	0.440	0.549	0.508	0.496	0.427	**0.416**
$\alpha = 20$	PSNR ↑	25.37	27.91	25.46	28.02	25.95	27.81	26.69	27.53	26.51	**28.29**	27.23
	SSIM ↑	0.658	0.775	0.655	0.764	0.636	0.778	0.630	0.678	0.715	**0.793**	**0.793**
	LPIPS-Alex ↓	0.426	0.275	0.429	0.288	0.427	0.265	0.401	0.313	0.360	0.256	**0.241**
	LPIPS-VGG ↓	0.492	0.421	0.496	0.437	0.485	0.410	0.495	0.446	0.466	0.402	**0.382**
$\alpha = 40$	PSNR ↑	25.67	28.34	25.72	28.27	26.22	28.13	27.24	28.14	26.85	**28.72**	27.11
	SSIM ↑	0.665	0.768	0.653	0.760	0.626	0.771	0.675	0.712	0.716	**0.788**	0.782
	LPIPS-Alex ↓	0.415	0.268	0.418	0.268	0.418	0.258	0.347	0.267	0.343	0.245	**0.221**
	LPIPS-VGG ↓	0.482	0.405	0.481	0.413	0.470	0.396	0.457	0.404	0.444	0.386	**0.360**

Table 3. **Performance on RealBlur-J Dataset with realistic blur** [27]: **Bold** and underline refer to overall best performing method and best synthetic performance method. It should be noted that methods that are not trained end-to-end are usually at disadvantage when comparing on metrics like PSNR. However, it can be seen that our reconstruction is generally preferred by other perceptual metrics.

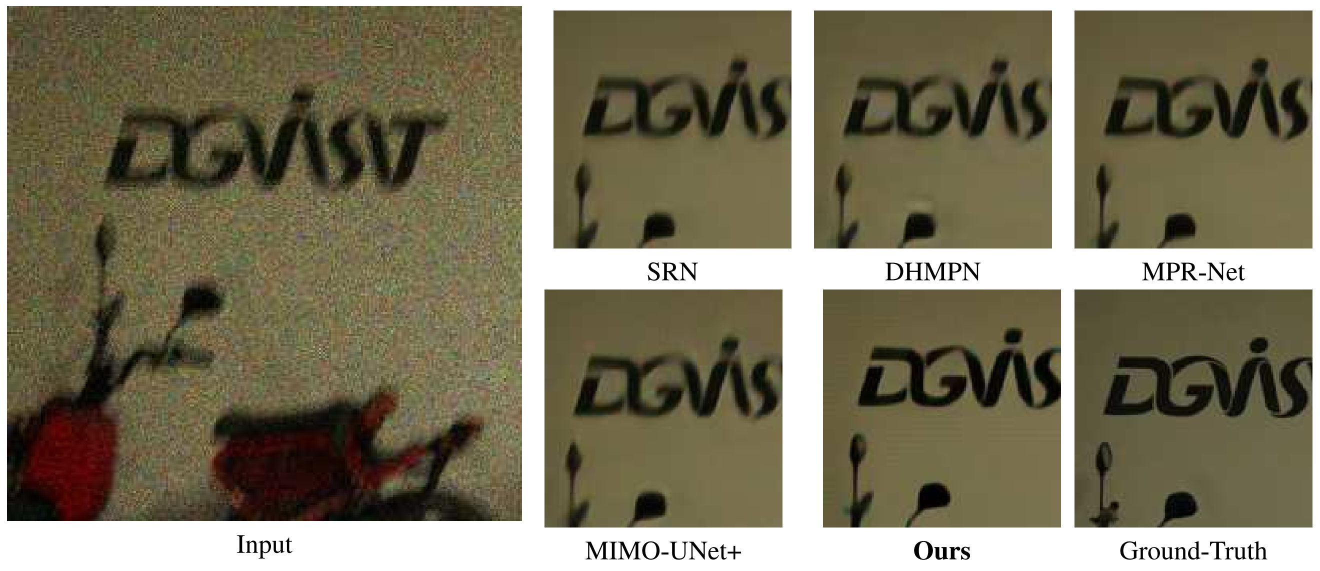


Figure 4. **Qualitative example on the Real-Blur Dataset**: For a more extensive set of results, we refer the reader to the supplementary document.

high noise regimes, we propose a novel low-dimensional representation to represent the motion blur. By using this novel low-dimensional and differentiable representation as a search space, we show state-of-the-art deconvolution performance and outperform end-to-end trained image restoration networks by a significant margin.

We believe this is a promising direction of research for both deblurring and general *blind inverse problems* i.e., inverse problems where the forward operator is not fully known. Future work could involve a supervised version of this scheme which does not involve backpropagation through a network as it would greatly reduce the computational cost. Another research direction could be to apply this framework to the problem of spatially varying blur.

Acknowledgement

The work is supported, in part, by the United States National Science Foundation under the grants IIS-2133032 and ECCS-2030570.

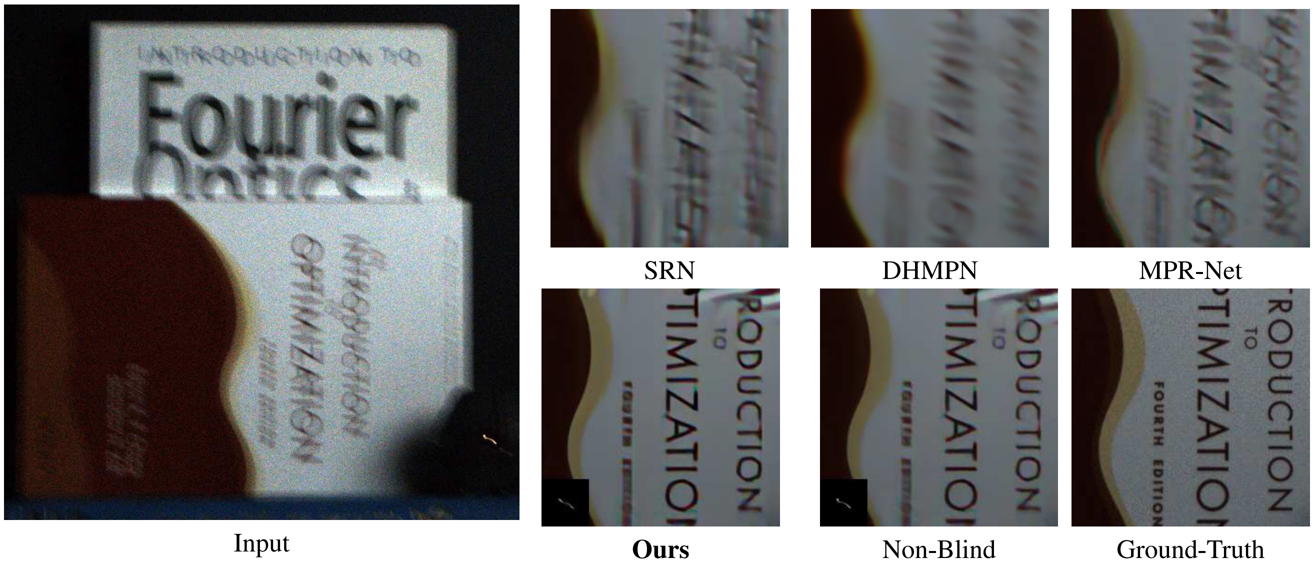


Figure 5. **Visual comparisons on Photon-Limited Deblurring Dataset**. Qualitative results on realistic blurred and photon-limited images from the Photon-Limited Deblurring dataset [29].The inset image for "Ours" and "Non-Blind" represent the estimated and ground-truth kernel respectively. For a more extensive set of qualitative results, we refer the reader to the supplementary document.

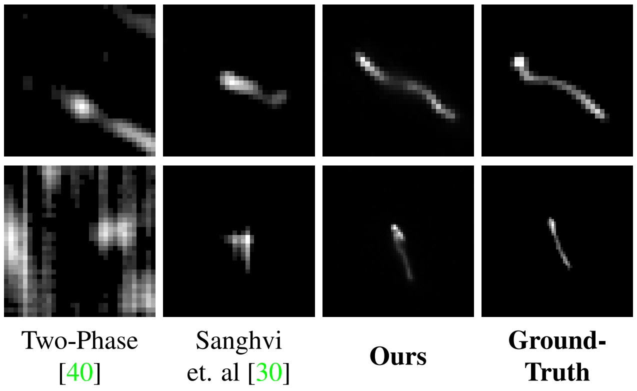


Figure 6. **Estimated Kernels for different methods**: We show the estimated kernels from two examples from the PLDD dataset. Two-Phase [40] uses blur-only image $G(\mathbf{y})$ as input, and ground-truth kernel is estimated using a point-source.

Key Points ↓ Photon Level ↓ Metric →		PSNR	SSIM	LPIPS-Alex	LPIPS-VGG
$\alpha = 10$	$K = 0$	25.44	0.719	0.295	0.434
	$K = 4$	25.90	0.730	0.286	0.423
	$K = 6$	26.38	0.735	0.279	0.422
	$K = 8$	**26.73**	**0.740**	**0.272**	**0.414**
$\alpha = 20$	$K = 0$	25.61	0.767	0.267	0.407
	$K = 4$	26.93	0.785	0.242	0.385
	$K = 6$	26.78	0.784	0.246	0.392
	$K = 8$	**27.26**	**0.795**	**0.240**	**0.384**
$\alpha = 40$	$K = 0$	25.82	0.759	0.248	0.390
	$K = 4$	26.88	0.771	0.228	0.374
	$K = 6$	**27.37**	0.780	0.216	0.366
	$K = 8$	27.29	**0.785**	**0.214**	**0.361**

Table 4. **Ablation Study for effect of Kernel Trajectory Network** $T(.)$: Reconstruction metrics for different number of key points. $K = 0$ represents the variant of the scheme which does not use the Kernel-Trajectory Network and estimates the kernel directly from Stage II.

References

[1] Chirag Agarwal, Shahin Khobahi, Arindam Bose, Mojtaba Soltanalian, and Dan Schonfeld. Deep-url: A model-aware approach to blind deconvolution based on deep unfolded Richardson-Lucy network. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3299–3303, 2020. 2

[2] Jérémy Anger, Mauricio Delbracio, and Gabriele Facciolo. Efficient blind deblurring under high noise levels. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 123–128. IEEE, 2019. 2

[3] Muhammad Asim, Fahad Shamshad, and Ali Ahmed. Blind image deconvolution using deep generative priors. *IEEE Transactions on Computational Imaging*, 6:1493–1506, 2020. 2

[4] Giacomo Boracchi and Alessandro Foi. Modeling the performance of image restoration from motion blur. *IEEE Transactions on Image Processing*, 21(8):3502–3517, 2012. 5

[5] Ayan Chakrabarti. A neural approach to blind motion deblurring. In *European Conference on Computer Vision (ECCV)*,

pages 221–235. Springer, 2016. 2

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

[7] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM Transactions on Graphics*, pages 1–8, 2009. 2

[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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4641–4650, 2021. 1, 2, 5, 6, 7

[9] Mauricio Delbracio, Ignacio Garcia-Dorado, Sungjoon Choi, Damien Kelly, and Peyman Milanfar. Polyblur: Removing mild blur by polynomial reblurring. *IEEE Transactions on Computational Imaging*, 7:837–848, 2021. 3

[10] Mario AT Figueiredo and Jose M Bioucas-Dias. Deconvolution of Poissonian images using variable splitting and augmented Lagrangian optimization. In *Proceedings of the IEEE/SP Workshop on Statistical Signal Processing*, pages 733–736, 2009. 2

[11] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2319–2328, 2017. 2

[12] Ankit Gupta, Neel Joshi, C Lawrence Zitnick, Michael Cohen, and Brian Curless. Single image deblurring using motion density functions. In *European Conference on Computer Vision (ECCV)*, pages 171–184. Springer, 2010. 2

[13] Zachary T Harmany, Roummel F Marcia, and Rebecca M Willett. This is SPIRAL-TAP: Sparse Poisson intensity reconstruction algorithms— theory and practice. *IEEE Transactions on Image Processing*, 21(3):1084–1096, 2011. 2

[14] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. DeblurGAN: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8183–8192, 2018. 2

[15] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better. In *The IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019. 2

[16] Anat Levin. Blind motion deblurring using image statistics. *Advances in Neural Information Processing Systems (NeurIPS)*, 19, 2006. 2, 6

[17] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971, 2009. 5

[18] Jizhou Li, Florian Luisier, and Thierry Blu. PURE-LET image deconvolution. *IEEE Transactions on Image Processing*, 27(1):92–105, 2017. 2, 5, 6

[19] Yuelong Li, Mohammad Tofighi, Vishal Monga, and Yonina C Eldar. An algorithm unrolling approach to deep image deblurring. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7675–7679. IEEE, 2019. 2

[20] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 5

[21] Leon B Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745, 1974. 2

[22] Zhiyuan Mao, Nicholas Chimitt, and Stanley H. Chan. Image reconstruction of static and dynamic scenes through anisoplanatic turbulence. *IEEE Transactions on Computational Imaging*, 6:1415–1428, 2020. 2

[23] Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021. 2

[24] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2, 5, 6, 7

[25] Dongwei Ren, Kai Zhang, Qilong Wang, Qinghua Hu, and Wangmeng Zuo. Neural blind deconvolution using deep priors. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3338–3347, 2020. 2

[26] William Hadley Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972. 2

[27] Jaesung Rim, Haeyun Lee, Jucheol Won, and Sunghyun Cho. Real-world blur dataset for learning and benchmarking deblurring algorithms. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–201. Springer, 2020. 2, 5, 7

[28] Arie Rond, Raja Giryes, and Michael Elad. Poisson inverse problems by the plug-and-play scheme. *Journal of Visual Communication and Image Representation*, 41:96–108, 2016. 2, 5, 6

[29] Yash Sanghvi, Abhiram Gnanasambandam, and Stanley H Chan. Photon limited non-blind deblurring using algorithm unrolling. *IEEE Transactions on Computational Imaging (TCI)*, 8:851–864, 2022. 2, 4, 5, 6, 8

[30] Yash Sanghvi, Abhiram Gnanasambandam, Zhiyuan Mao, and Stanley H. Chan. Photon-limited blind deconvolution using unsupervised iterative kernel estimation. *IEEE Transactions on Computational Imaging*, 8:1051–1062, 2022. 1, 2, 3, 5, 8

[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. 2

[32] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics*, 27(3):1–10, 2008. 2

[33] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion

blur removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 769–777, 2015. 2

[34] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8174–8182, 2018. 1, 2, 5, 6, 7

[35] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9446–9454, 2018. 2

[36] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009. 5

[37] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general U-shaped transformer for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17683–17693, June 2022. 2

[38] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Deblurring shaken and partially saturated images. *International Journal of Computer Vision*, 110(2):185–201, 2014. 2

[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. 2

[40] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 157–170. Springer, 2010. 2, 5, 8

[41] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[42] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14821–14831, 2021. 1, 2, 5, 6, 7

[43] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5978–5986, 2019. 1, 2, 5, 6, 7

[44] Kaihao Zhang, Wenhan Luo, Yiran Zhong, Lin Ma, Bjorn Stenger, Wei Liu, and Hongdong Li. Deblurring by realistic blurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2737–2746, 2020. 2

[45] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 5, 6

[46] 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. 2