Fast Non-blind Deconvolution via Regularized Residual Networks with Long/Short Skip-Connections

Hyeongseok Son
POSTECH
sonhs@postech.ac.kr

Seungyong Lee
POSTECH
leesy@postech.ac.kr

Abstract

This paper proposes a novel framework for non-blind deconvolution using deep convolutional network. To deal with various blur kernels, we reduce the training complexity using Wiener filter as a preprocessing step in our framework. This step generates amplified noise and ringing artifacts, but the artifacts are little correlated with the shapes of blur kernels, making the input of our network independent of the blur kernel shape. Our network is trained to effectively remove those artifacts via a residual network with long/short skip-connections*. We also add a regularization to help our network robustly process untrained and inaccurate blur kernels by suppressing abnormal weights of convolutional layers that may incur overfitting. Our postprocessing step can further improve the deconvolution quality. Experimental results demonstrate that our framework can process images blurred by a variety of blur kernels with faster speed and comparable image quality to the state-of-the-art methods.*

1. Introduction

Image deconvolution has been actively studied in the fields of image processing and computer vision. The degradation model of image deconvolution is usually defined by $y = x * v + n$, where y is the blurred image, x is the latent sharp image, v is the blur kernel, n is additive noise, and $*$ denotes convolution operator. The estimation of the latent image x for a given input y is known as non-blind deconvolution if the blur kernel v is provided. Non-blind deconvolution is an ill-posed problem as multiple solutions may exist due to noise. Previous deconvolution methods [23, 22], which regularize image gradients to be sparse by natural image statistics, work well under the ideal conditions with low noise levels and accurate blur kernels.

These methods, however, cannot restore the latent sharpness if the input blurred image contains strong noise. Suppressing artifacts using general image priors could be effective up to a certain level of noise, but may not work for severe cases. To resolve this limitation, a few methods utilized local priors useful for restoring natural images patches, such as patch statistics [40] and example patches [12]. Denoising based methods were also proposed for non-blind deconvolution [7, 8] and achieved the state-of-the-art performances by removing artifacts from deconvolution operation while preserving the restored image structures. A common drawback of all these methods is slow speed.

Recently deep learning has been used for several image processing problems, such as super-resolution [9, 19] and denoising [4, 35, 2], with improved performance. From the same perspective, we can consider applying deep learning to image deconvolution. However, the conventional approach to train a deep network using image pairs would not be effective, as the blurred images change with blur kernels even though the latent images are the same. Training with such data would be tricky because of the large variances caused by different blur kernels.

To avoid this difficulty, previous methods using deep learning focused on handling a single blur kernel. Similar to the denoising based approach, Schuler *et al.*[31] used a MLP (multi-layer perceptron) to remove the noise remaining in the result of simple deconvolution, where the pairs of patches from the ground-truth and deconvolved images were used for training. A CNN (convolutional neural network) based approach [37] was proposed to directly use the pairs of the ground-truth and blurred images for training without pre-deconvolution. Both methods showed good performance, but they have a common limitation that only fixed kernels can be handled. The network must be retrained to support deconvolution with a different kernel.

In this paper, we propose a deep learning based framework for non-blind image deconvolution that can handle arbitrary blur kernels. Our key observation is that the training complexity can be reduced by including a pre-deconvolution step. This step may produce deconvolution artifacts, such as amplified noise and ringing artifacts, but makes the input of our network to be trained rather independent of the blur kernel shape. The specific artifacts caused by the pre-deconvolution step might change with the

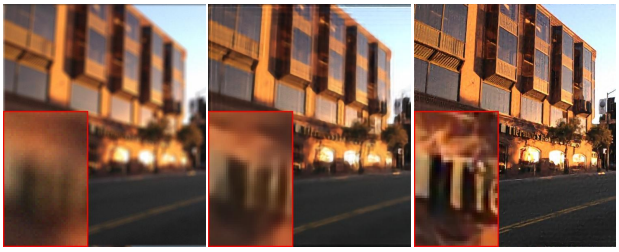
blur kernels, but the variances of the artifacts would be obviously smaller than those of blurred images themselves. We choose the Wiener filter [34] as the pre-deconvolution method, as it is simple and fast while generating images with latent details, although noisy. Our network is then trained with results of the Wiener filter and the ground-truth latent images. The network consists of multiple convolution layers, and can clearly restore image structures after training. Our framework also includes a postprocessing step to improve the deconvolution quality with additional image details.

Training our network is not straightforward as the pre-deconvolved images in the training data contain various artifacts from different blur kernels and image contents. We propose a *residual network with long/short skip-connections* to effectively remove the artifacts, which can combine the benefits of two recently proposed residual networks [14, 19]. We also add a regularization term to the basic Euclidean loss function in the backpropagation step. This regularization term helps us prevent abnormal weights of convolutional layers and makes our network robustly process untrained blur kernels by avoiding overfitting. We demonstrate with experimental results that the network can be successfully trained in our framework and generates deconvolution results with faster speed and comparable quality to other state-of-the-art methods.

2. Related Work

Non-blind deconvolution Non-blind deconvolution methods restore the latent images usually based on probabilistic image priors. Early papers [23, 22], still the state-of-the-art, use natural image statistics and sparsity priors for suppressing artifacts that arise in the deconvolution process. Schmidt *et al.* [30] used field of experts [27] for modeling the priors, and the method was extended by [29, 28] in terms of discriminative learning. In addition to the priors, patch statistics [40] and adaptive prior [11] were considered for deconvolution. These deconvolution methods can successfully restore image structures but have some limitations in restoring image details.

There is another approach to deconvolution based on image denoising [7, 31]. In this approach, a simple deconvolution method is first applied to the given blurry image, and then the artifacts in the deconvolution result are removed by a denoising method. Advanced denoising methods based on BM3D algorithm [6], such as DEB-BM3D [7] and IDD-BM3D [8], can be used for removing deconvolution artifacts, but they are commonly slow. Recently a MLP method [31] was proposed to learn the denoising operation for image deconvolution, and showed better performance than previous denoising methods. However, this method uses a single kernel for training, and the trained model cannot handle deconvolution for other kernels.



(a) input (b) Xu *et al.* [37] (c) our method

Figure 1. Existing deep learning based approach [37] cannot handle an untrained blur kernel. Input image (a) was blurred by a disk kernel with radius 8. In (b), the network of [37] trained by a disk kernel with radius 7 was used for deconvolution. PSNRs are 21.38 and 26.59dB for (b) and (c), respectively. The original image was taken from [37].

Deep learning for image deblurring Xu *et al.* [37] proposed a non-blind deconvolution method based on CNN using 1D-separable kernels to cover large inverse blur kernels. However, their network can handle only one kernel at a time, and needs to be trained separately for different kernels. Sun *et al.* [33] addressed the non-uniform image deblurring problem using deep learning, but their method cannot handle general kernel shapes because it trained a classification network using 70 fixed blur kernels. Hradi *et al.* [16] considered text deblurring, and used deep learning to directly remove non-uniform blurs through a network. Text documents have more specific characteristics than natural images, and by exploiting the characteristics, general kernels with different shapes could be handled.

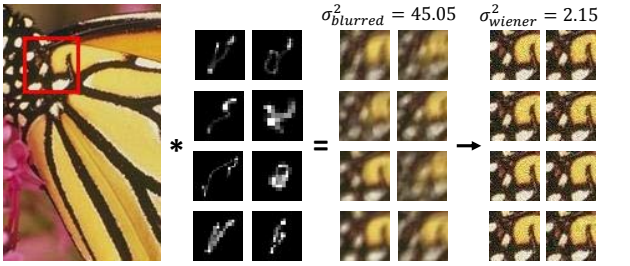
Schuler *et al.* [32] proposed a deep learning based framework for image deblurring. However, their kernel estimation module is not appropriate for estimating big blur kernels and simple L_2 deconvolution is used for the image restoration layer. The accuracy is not yet comparable to the state-of-the-art methods that do not use deep learning.

Deep learning for image restoration Deep learning has been used for several image restoration problems, such as denoising [4, 35, 2], inpainting [35, 26], deraining [10], and filtering [38]. Especially, image super-resolution is closely related to non-blind deconvolution, as it is a kind of image deconvolution with small Gaussian blur kernels. Dong *et al.* [9] proposed a basic network for super-resolution, which uses only three layers. Kim *et al.* [19] addressed multi-scale super-resolution using a very deep network.

3. Non-blind Deconvolution Using a CNN

3.1. Basic idea

In the previous deep learning based method [37], the network is trained for a single kernel and cannot handle decon-



(a) input (b) blur kernels (c) blurred (d) Wiener filtered

Figure 2. Training complexity can be reduced a lot by applying pre-deconvolution. The variations in the Wiener filtered images are much less than those in the images blurred with different kernels.

volution with other kernels (Fig. 1). Extending the approach of [37] simply by using images blurred with many different kernels for training would not work due to high variations in the blurred images. Our main idea is to reduce the complexity of the network input by applying pre-deconvolution to the blurred images. Pre-deconvolution may introduce some artifacts, but training a network for removing the artifacts is easier than training for the image deconvolution operation itself, as the variations in the deconvolution artifacts are smaller than those in the blurred images (Fig. 2).

Our approach is similar to denoising based approaches [7, 8] in that both contain a deconvolution step followed by noise removal. However, after network training, our deconvolution runs much faster and achieves better image quality (Sec. 5). The method of Schuler *et al.* [31] also applies simple L_2 deconvolution and removes the noise using a neural network. However, their network handles a single kernel and cannot be used for deconvolution with multiple kernels. We demonstrate our method achieves better quality and higher speed than [31] in Sec. 5.

3.2. Overall framework

The overall process of our framework contains three steps: (1) Wiener filtering, (2) artifact removal by a network, and (3) postprocessing (Fig. 3). We apply Wiener filter to an input blurred image, producing a sharp image probably with amplified noise and ringing artifacts. Our network is then trained to remove the deconvolution artifacts while preserving the sharpness of image structures. A postprocessing can be applied to the output of the network if further improvements of image quality are needed

3.3. Network

The role of our network is to remove the artifacts from the previous simple deconvolution step. We take a convolutional network for the task, with a similar overall structure to [38], and use convolution layers and Parametric Rectified Linear Units (PReLU) [13] alternatingly. Our network contains total 11 convolution layers, of which 10 layers have $3\times3\times32$ kernels and the final layer has $1\times1\times3$ kernels for reconstructing an image (Fig. 3). We can use a deeper network, but from preliminary experiments described in Sec. 4.2, it turned out that our problem is not so complex as to need a very deep network. So we chose a simple network model to increase the speeds of training and running the network in our framework. The inference time of our network for a 800×600 image is less than 0.1 second.

Residual network with long/short skip-connections Our network architecture is originally inspired by [19], which proposed a residual network for super-resolution. The residual network uses a single skip-connection to link the first data layer and the final reconstruction layer, and works better than conventional networks [9], which simply use a stack of convolution layers, due to a long-term memory effect. However, residual learning is also known to be effective for handling small changes between layers [14], which is a desired property in our image restoration task. A single long skip-connection in [19] would not be enough for capturing small changes between consecutive convolution layers. So we add a short skip-connection to every convolution layer, and our network finally contains both long and short skip-connections (Fig. 3). This long/short skip-connections can take both advantages of capturing small changes [14] and the long-term memory effect [19], as analyzed in Sec. 4.2. We also use the pre-activation technique [15], which places a PReLU layer before each convolution layer. The residual layers are implemented by simply combining existing layers with element-wise summation.

3.4. Image gradient regularization for training

Existing image processing methods [37, 9, 19] usually use the Euclidean (L_2 distance) loss function between the ground-truth and the result of the network for the training. In an one-to-many problem, however, it may converge to a local optimum or incur overfitting. To control the direction of convergence, we adopt a regularization term in the loss function.

We add the sparse prior [23] into the loss layer as a regularization term. In a discrete image, our regularization term is calculated by

$$R(f) = \alpha \sum_{i,j} w_{i,j}(|f_{i,j} - f_{i-1,j}|^p + |f_{i,j} - f_{i,j-1}|^p), \quad (1)$$

where $p = 0.8$ and $w_{i,j} = e^{-10|\nabla x_{i,j}|^2}$. f and x denote the network output and the label image respectively. A weight α is a weight constant of the regularization term. We append an adjustment weight w for preventing the regularization term from affecting edges. The weight w is controlled

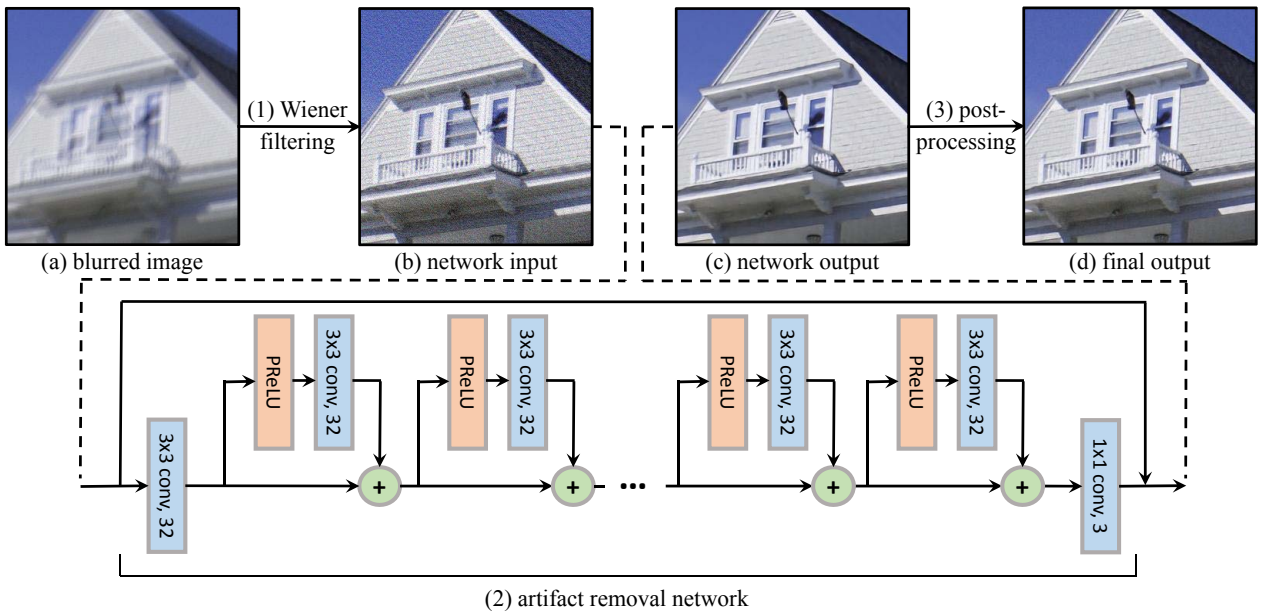


Figure 3. Overall framework with our network architecture

by gradients of label images, making smooth image regions in label images affect weights of the network more. Calculating the gradient of the regularization term for backpropagation is explained in the supplemental material. As the regularization term is included in the loss layer, it only affects the training time and has no influence on the inference speed.

3.5. Training

Data preparation We used images taken from BSDS500 dataset [3] and augmented this data by flipping and rotating images. We trained the network with images synthetically blurred by various kernels. For simplicity, we used linear blur kernels that have various lengths ($11 \sim 31$ pixels) and orientations. We tested other kinds of motion blur kernels as well, but the final deconvolution results were of similar quality. We added random Gaussian noise ($\sigma = 0.35\% \sim 3.5\%$) to the blurred images. We then applied Wiener filter to the images, where the noise-to-signal ratio (NSR) of Wiener filter was automatically estimated by our estimation method described in the supplemental material. We used patches of size 61×61 for training, cropped from the images with stride 31.

Gradient vanishing/explosion prevention We used stochastic gradient descent (SGD) optimizer and adjustable gradient clipping [19] for training our network. In image recognition, batch normalization [17] is widely used for preventing the gradient vanishing problem. We tested batch normalization in our network and discovered that it introduces unnecessary shifts of feature distributions that may hinder image reconstruction. So we used adjustable gradient clipping instead, which suppresses abnormally big gradients with high learning rates and preserves big gradients with low learning rates, helping the network converge faster while preventing gradient exploding. With this gradient clipping (the clipping threshold is 0.05 in our training), our network could converge within six hours with the learning rate 0.1. We tested other optimizers such as Adam [20], but the combination of SGD and adjustable gradient clipping was most effective for our training.

3.6. Postprocessing

Our network removes artifacts while preserving the sharpness of structural edges, but it may not effectively preserve very small details. The problem may be relieved by using more layers, but it needs larger physical memory and longer training/test time. To resolve this limitation without increasing the number of layers, we can perform a simple optimization as a postprocessing step for preventing the loss of details using

$$\|y_1 - v * x\|^2 + \lambda \|y_2 - x\|^2, \tag{2}$$

where y_1, y_2 and v are the input blurred image, the output of our network, and the blur kernel, respectively. To estimate the latent image x, as mentioned in [5, 31], we can simply compute the solution of Eq. (2) in the Fourier domain by

$$X = \frac{\overline{V} \odot Y_1 + \lambda Y_2}{|V|^2 + \lambda}, \tag{3}$$

where X, V, Y_1, Y_2 are the Fourier representations of x, v, y_1, y_2, respectively, and $\overline{V}$ is the complex conjugate of V. $\odot$

(a) blurred image (b) $L2$ deconv (c) Wiener filter

(d) ground-truth (e) Network by (b) (f) Network by (c)

Figure 4. Deconvolution results of two networks trained by different pre-deconvolution methods for a noisy blurred image. PSNRs of (b, c, e, f) are 21.92, 20.76, 25.87, and 26.47dB, respectively.

denotes element-wise multiplication. This step brings more details to the final output without complex regularization while preserving the quality of the network output.

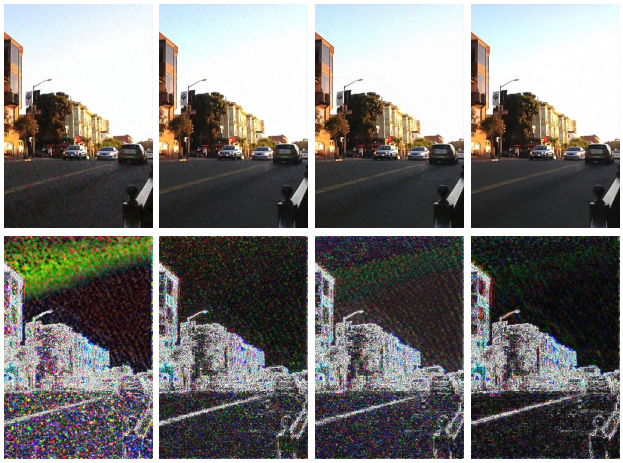
4. Analysis

4.1. Pre-deconvolution method

Required properties Pre-deconvolution methods should satisfy two properties in our framework. Its result should include the information of inherent details of the latent image as our network would not be able to restore already damaged details. It should also generate consistent deconvolution results in terms of artifacts so that our network can be well trained for effectively removing the artifacts.

Wiener filter Based on the required properties, we use a Wiener filter as pre-deconvolution in our framework. Although the Wiener filter may produce quantitatively worse deconvolution results than other methods, its results usually contain more details while other methods may sacrifice the details for suppressing artifacts (Fig. 4). In other words, the direct result of the Wiener filter would be of low quality, but its potential for quality improvement could be high. We can also expect that the Wiener filter produces similar patterns of artifacts when used with optimal parameters that can be computed from the variances of the latent image and noise. As we use a synthetic dataset in the training step, we could calculate the optimal parameters from the ground-truth image and the added noise, and generate a consistent dataset appropriate for training our network. In the test step, we used parameters estimated from the input blurry images, as mentioned in Sec. 3.5.

For comparison, we tested L_2 regularized deconvolution, which is simple and fast, and was used in [31]. We found that L_2 deconvolution fails to restore small details when the blur kernel is large, and in that case, the final deconvolution results from the network in our framework are of lower quality with L_2 deconvolution than with the Wiener filter (Fig. 4). In other cases with relatively small blur kernels, the final deconvolution results with both deconvolution methods were similar.



(a) naïve (b) [19] (c) [14] (d) ours

Figure 5. Deconvolution results and visualization of the differences from the ground-truth ($20\times$ brighter). In the network, a long skip-connection is useful for preserving the overall color in a smooth region (b), while many short skip-connections are useful for restoring sharp details (c). PSNRs are 26.86, 29.18, 29.18, 29.94dB from (a) to (d). The original image is the same as the one used in Fig. 1.

4.2. Analysis on network architecture

When we used a deep convolutional network without a residual structure in our framework, the deconvolution results showed imperfect restoration of sharp details and global color differences from the ground-truth images (Fig. 5a). Our residual network resolves these problems by adapting two kinds of residual structures in the network. A residual structure with a long single skip-connection between the first data layer and the final reconstruction layer, proposed in [19], is effective for preserving the global color by transferring the original input to the output (Fig. 5b). The other residual structure with many short skip-connections between adjacent layers, proposed in [14], helps recovering sharp details as a small residual block is useful for training small perturbations of features (Fig. 5c). By combining these two residual structures, we can improve the image quality even more than [19] and [14] (Table 1).

In our network, the number of layers is also an important factor for determining the performance. The accuracy slightly increases as the network includes more layers (Table 2). However, the quality improvement is relatively small compared to the increased computational and memory over-

network architecture	PSNR
network w/o skip-connections	27.26
network w/ single long skip-connection	29.58
network w/ many short skip-connections	29.42
our residual network	**29.85**

Table 1. Accuracy (in dB) of various network architectures. All networks use 10 convolution layers and 32 channels.

# of layer	w/o PP	w/ PP	# of channels	w/o PP	w/ PP
5	30.07	30.40	32	30.07	30.40
10	30.39	30.69	64	30.25	30.57
15	30.45	30.72	96	30.28	30.59
20	30.43	30.71	128	30.28	30.58

Table 2. Accuracy (in dB) with different numbers of layers and channels used in our network. We fixed the number of channel as 32 when changing the number of layers. The number of layers was fixed as 5 for channel number changes. PP denotes the post-processing step in Sec. 3.6.

test set	α			
	0	0.001	0.01	0.05
accurate kernels	29.51	29.53	29.62	27.71
inaccurate kernels	23.72	24.51	25.93	25.96

Table 3. Accuracy (in dB) with different regularization weights α. The test for inaccurate kernels used an inaccurate blur kernel degraded by Gaussian smoothing.

heads. In our experiments, our network almost reached the performance limit at 15 layers, where the Euclidean loss could not be reduced anymore. Based on this observation, we use a small number of layers, which enables fast inference speed. Further quality improvements can be achieved by adopting the post-processing step in Sec. 3.6, instead of increasing the number of layers. Table 2 shows that the performance gaps between different numbers of layers become smaller after postprocessing.

4.3. Effects of regularization term

In many applications of non-blind deconvolution, the kernel may not be expected to be accurate. For example, in image deblurring, the kernel used for deconvolution is an estimated one and usually not very accurate. Consequently, our deconvolution method should be able to produce reasonable results even for inaccurate kernels. In addition, although we train our network with only linear blur kernels, it should produce plausible deconvolution results for arbitrary kernels. Image gradient regularization in Sec. 3.4 helps our network properly handle inaccurate and untrained kernels.

The regularization term in Eq. (1) suppresses convolutional kernel weights in the network to avoid overfitting

(a) $\alpha = 0$ (b) $\alpha = 0.001$ (c) $\alpha = 0.01$ (d) $\alpha = 0.05$

Figure 6. Visual changes with different regularization weights α. We used an inaccurate blur kernel degraded by Gaussian smoothing. PSNRs are 22.80, 23.58, 25.02, 25.62dB from the left.

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

Figure 7. Blur kernels used for Table 5. The sizes of the kernels are (a) 19×19, (b) 17×17, (c) 15×15, (d) 27×27, and (e) 23×23.

and prevents the network from unexpectedly responding to the data patterns not included in the training set. Table 3 shows that our network trained without the regularization term ($\alpha = 0$) works well for accurate kernels but the performance is degraded for inaccurate kernels. When we add the regularization with a proper weight ($0.001 \leq \alpha \leq 0.01$), the quality for accurate kernels remains the same but inaccurate kernels can be handled much better. Fig. 6 visualizes these effects. Deconvolution examples in Sec. 5 also show that our method can effectively handle arbitrary kernels.

5. Experiment Results

5.1. Non-blind deconvolution

We implemented our framework and tested with various images on Intel Core i7 CPU and NVIDIA Titan X GPU. We compared our method with several other methods; sparse prior [23], HL [22], IDD-BM3D [8], MLP [31], adaptive prior [11], and CSF [28]. For all other methods, we used MATLAB implementations provided by the authors. For training our network, we used Caffe library [18]. We ran 100 iterations for the sparse prior method [23], and 50 iterations for IDD-BM3D [8] to reduce the running times.

After training, our deconvolution method needs only a few FFT and convolution operations and runs faster than other methods (Table 4), where [23, 22, 8] use CPU only and [31, 11, 28] and our method use GPU as well. In fact, the Wiener filter is the slowest step in our framework. For a 800×600 image, our framework spends 0.14s in the Wiener filter, 0.09s in our network, and 0.06s in the postprocessing.

We used synthetically blurred images for the evaluation. We took 24 images from the Kodak Lossless True Color Image Suite [1], and 5 blur kernels from [24] (Fig. 7). We used

Image size	Levin [23]	HL [22]	IDD-BM3D [8]	MLP [31]	Fortunato [11]	CSF [28]	Ours
400×300	20.29	0.60	120.38	1.25	0.13	0.88	**0.12**
800×600	87.35	0.99	481.50	4.79	0.46	1.09	**0.29**
1600×1200	324.44	2.65	1940.43	19.01	2.17	2.75	**0.95**

Table 4. Running times (in seconds) on color images. The blur kernel used in this experiment is in Fig. 7d.

kernel type	noise level	Levin [23]	HL [22]	IDD-BM3D [8]	MLP [31]	Fortunato [11]	CSF [28]	$\alpha = 0$ w/o PP	$\alpha = 0$ w/ PP	$\alpha = 0.01$ w/o PP	$\alpha = 0.01$ w/ PP
(a)	1%	30.06	30.24	31.83	30.46	30.63	30.42	32.39	32.60	32.73	**32.78**
	3%	27.17	26.95	28.02	26.17	27.29	26.93	28.86	28.93	28.93	**28.94**
(b)	1%	29.89	30.03	31.49	30.46	30.33	30.06	32.44	32.61	32.64	**32.70**
	3%	26.88	26.88	27.93	26.12	27.13	26.89	28.79	**28.88**	28.87	**28.88**
(c)	1%	30.52	30.56	31.69	30.88	30.74	30.41	32.35	32.67	33.00	**33.13**
	3%	27.49	27.60	28.54	27.02	27.75	27.52	29.44	**29.59**	29.57	**29.59**
(d)	1%	29.46	29.68	31.34	29.59	29.41	29.27	31.85	31.98	32.10	**32.14**
	3%	26.48	26.54	27.55	25.79	26.50	26.38	28.19	**28.28**	28.23	28.25
(e)	1%	29.89	30.10	31.23	31.27	29.96	29.80	32.00	32.19	32.23	**32.34**
	3%	27.03	27.07	28.00	26.71	27.13	27.01	28.73	**28.81**	28.68	28.69

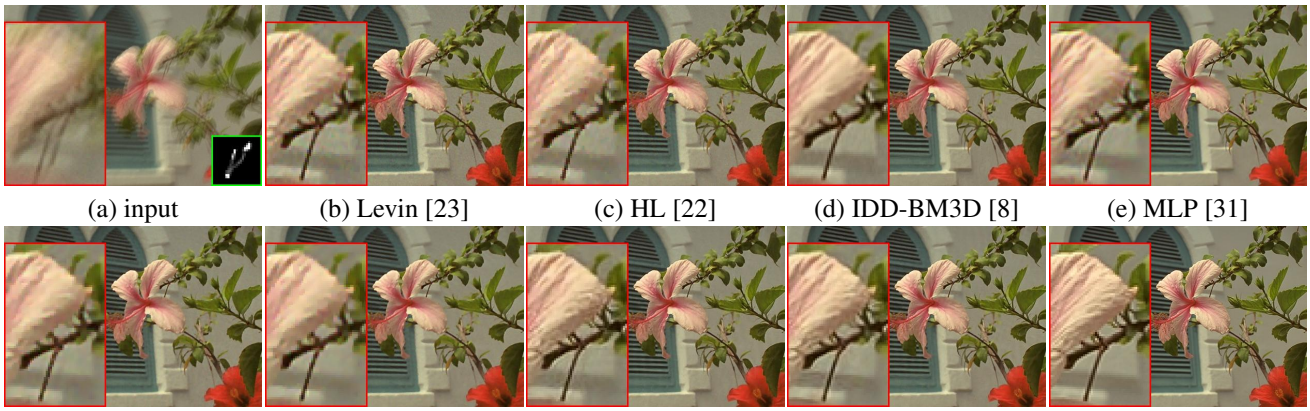
Table 5. Quantitative comparison using the Kodak Lossless True Color Image Suite [1] (values in dB). A blur kernel used in MLP [31] to train its model was used in this experiment (kernel (e) and 1% noise level). PP denotes postprocessing.

circular convolution to make blurred images, but excluded the boundary when calculating the accuracy as [31] did. Our method can handle blurry images made by a cropped convolution as described in the supplemental material. We tested two networks which are trained with and without the regularization term ($\alpha = 0.01$ and $\alpha = 0$). For each degradation model with a different blur kernel and noise level, we found the best parameters for each method using a few images and fixed the parameters for testing with 24 images. Specifically, we retrained the model of CSF [28] to handle different noise levels. Table 5 shows our method outperforms prior-based methods [23, 22, 11, 28]. Our method also achieves better quantitative results than IDD-BM3D [8] and MLP [31] with much faster speed. In the last row of Table 5, our method shows better performance than [31] even in the condition that the method was trained for.

Figs. 8 and 9 show qualitative results on synthetically blurred images used in Table 5. Our trained network can successfully handle various cases. Fig. 8 is a relatively easy example, where the testing condition is the same as in [31]. In that case, all methods show visually pleasing results, while ours are the sharpest. Fig. 9 is a difficult case which contains strong noise. Our results visually outperform other state-of-the-art methods in this case. Prior-based methods cannot restore sharp details in order to suppress artifacts, and IDD-BM3D also cannot preserve structural edges under the high noise level. MLP does not work well in the condition that the network has not been trained for. Both networks trained with and without the regularization show similar performance in this test, and both networks outperform other methods. More experimental results on extremely noisy images are included in the supplemental material.

Fig. 10 shows qualitative comparisons with extremely noisy examples. Our method is robust to very high noise levels, even though the Wiener filter in our framework might be vulnerable to noise. Although our network had been trained with lower noise levels ($0.35 \sim 3.5\%$) as described in Sec. 3.5, our method still showed better accuracy than HL [22] and CSF [28] as shown in the top row of Fig. 10. For better handling of higher noise levels, our network can be re-trained. We trained the network with noise levels ($0.35 \sim 10\%$), and our method again showed a better result than HL and CSF at a very high noise level (10%) as shown in the bottom row of Fig. 10, although all methods showed low visual quality in that case. HL with weak and strong regularizations made images too smooth and too noisy, respectively. CSF also did not show particularly superior performance than HL. In contrast, our method reproduced sharper structures in the images than other methods.

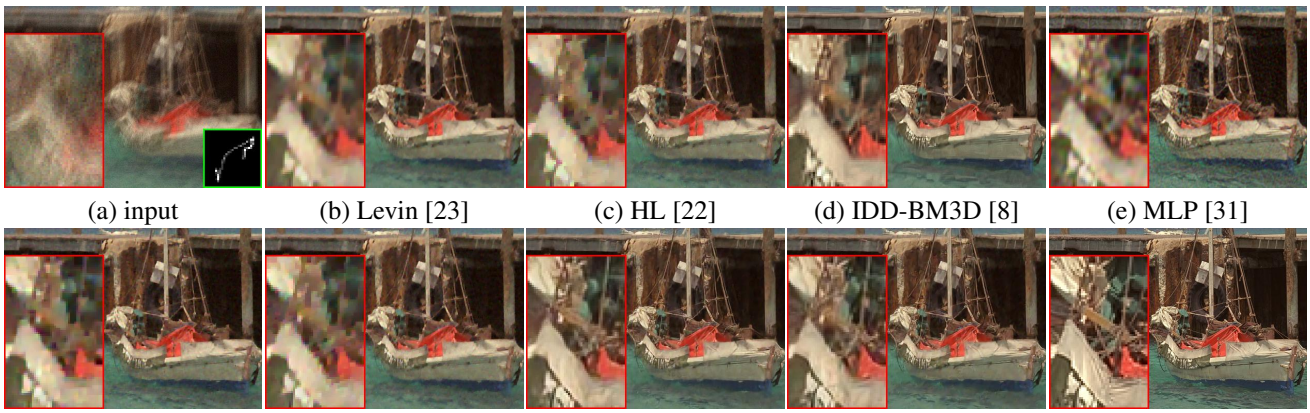
Fig. 11 shows qualitative comparisons with real examples. We tuned the parameters of all methods to obtain the best results. In the examples, our network without the regularization term could sharpen noise but shows comparable quality with other methods. On the other hand, our network with the regularization term removes artifacts more aggressively than other methods, while still nicely restoring the fine structures in the images.



(a) input (b) Levin [23] (c) HL [22] (d) IDD-BM3D [8] (e) MLP [31]

(f) Fortunato [11] (g) CSF [28] (h) ours ($\alpha = 0$) (i) ours ($\alpha = 0.01$) (j) ground-truth

Figure 8. Qualitative comparison with the training case of [31]. The blurred image was blurred by kernel (e) in Fig. 7 and weak noise (1%), for which the network in [31] was trained. PSNRs of the top row are 31.57, 31.74, 33.88, 33.71 dB from (b) to (e). PSNRs of the bottom row are 31.96, 31.75, 34.59, **34.79**dB from (f) to (i).



(a) input (b) Levin [23] (c) HL [22] (d) IDD-BM3D [8] (e) MLP [31]

(f) Fortunato [11] (g) CSF [28] (h) ours ($\alpha = 0$) (i) ours ($\alpha = 0.01$) (j) ground-truth

Figure 9. Qualitative comparison with more difficult case. We changed the blur kernel to kernel (d) in Fig. 7 and made the noise stronger (3%). PSNRs of the top row are 26.01, 26.11, 26.92, 25.46 dB from (b) to (e). PSNRs of the bottom row are 25.97, 25.83, **27.85**, 27.84dB from (f) to (i).

5.2. Blind deconvolution

Our non-blind deconvolution method can be used for image deblurring. Our method is so fast that it can be effectively used in the iterative process of kernel estimation. We replaced the latent image estimation step by our method in an existing image deblurring method [39], whose implementation was released by the authors of [25]. We then evaluated our image deblurring framework using two image deblurring benchmarks [21, 24]. Our method produced results comparable to the state-of-the-art deblurring methods, and these results are included in the supplemental material.

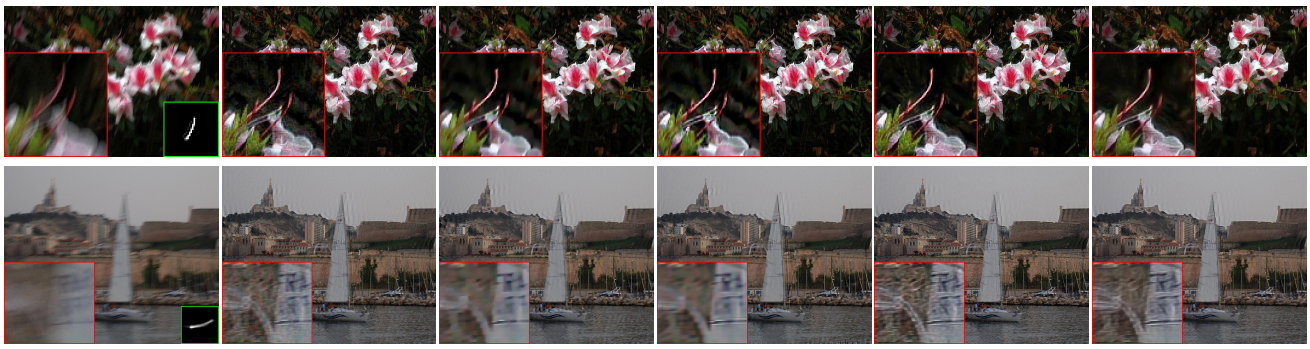
6. Conclusion

In this paper, we have proposed a novel framework for uniform non-blind deconvolution using a pre-processing step of simple deconvolution. Although the framework consists of simple functions and network architecture, our results are comparable to existing state-of-the-art results with faster speed. Experimental results show that our approach can produce high quality deconvolution results for the images blurred by various blur kernels with different noise levels using only a single trained network. Since we focus on increasing the range of blur kernels which our network can handle to solve the general deconvolution problem, dedicated approaches could show better performance than ours in some specific settings, such as blurred images with saturations. Nevertheless our approach has benefits for handling image deconvolution in term of generality and speed, with comparable quality.

Since our framework uses the Wiener filter, it cannot handle non-uniform blur. Additional process, such as ap-

(a) input (b) HL (weak) [22] (c) HL (strong) [22] (d) CSF [28] (e) ours (f) ground-truth

Figure 10. Qualitative comparison with extremely noisy images. Blurred images in the top and bottom rows of (a) have very high noise levels (5% and 10%, respectively). PSNRs of the top row are 27.16, 26.44, 26.73, **28.28** dB from (b) to (e). PSNRs of the bottom row are 22.93, 22.66, 22.72, **24.07** dB from (b) to (e).



(a) input (b) Wiener filter (c) HL [22] (d) IDD-BM3D [8] (e) ours ($\alpha = 0$) (f) ours ($\alpha = 0.01$)

Figure 11. Qualitative comparison with real photographs. We took the input images and the estimated blur kernels from [25, 36].

plying adaptive deconvolution using non-uniform blur kernel, would be effective for solving this problem. Developing a fast pre-deconvolution method handling outliers and saturation and combining the pre-deconvolution and post-processing steps into our network would also be interesting future work.

Acknowledgements

This work was supported by Institute for Information and Communications Technology Promotion (IITP) Grant (R0126-16-1078) and the National Research Foundation of Korea (NRF) Grant (NRF-2014R1A2A1A11052779) both funded by the Korea government (MSIP).

References

[1] Kodak lossless true color image suite. `http://r0k.us/graphics/kodak/`. Accessed: 2016-10-01.

[2] F. Agostinelli and M. R. Anderson. Adaptive multi-column deep neural networks with application to robust image denoising. In *Proc. NIPS*, 2013.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. PAMI*, 33(5):898–916, 2011.

[4] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Proc. CVPR*, 2012.

[5] S. Cho and S. Lee. Fast motion deblurring. *ACM Trans. Graph*, 28(5):1, 2009.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3D filtering. In *Proc. SPIE Electronic Imaging*, 2006.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3D transform-domain collaborative filtering. In *Proc. SPIE Electronic Imaging*, 2008.

[8] A. Danielyan, V. Katkovnik, and K. Egiazarian. BM3D frames and variational image deblurring. *IEEE Trans. Image Process*, 21(4):1715–1728, 2012.

[9] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *Proc. ECCV*, 2014.

[10] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *Proc. ICCV*, 2013.

[11] H. E. Fortunato and M. M. Oliveira. Fast high-quality non-blind deconvolution using sparse adaptive priors. *The Visual Computer*, 30(6-8):661–671, 2014.

[12] Y. Hacohen, E. Shechtman, and D. Lischinski. Deblurring by example using dense correspondence. In *Proc. ICCV*, 2013.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. ICCV*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016.

[16] M. Hradi. Convolutional neural networks for direct text deblurring. In *Proc. BMVC*, 2015.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM MM*, 2014.

[19] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *Proc. CVPR*, 2016.

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[21] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *Proc. ECCV*, 2012.

[22] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *Proc. NIPS*, 2009.

[23] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph*, 26(3):70, 2007.

[24] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proc. CVPR*, 2009.

[25] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *Proc. CVPR*, 2016.

[26] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *Proc. NIPS*, 2015.

[27] S. Roth and M. J. Black. Fields of Experts: A framework for learning image priors. In *Proc. CVPR*, 2005.

[28] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proc. CVPR*, 2014.

[29] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and R. Stefan. Discriminative non-blind deblurring. In *Proc. CVPR*, 2013.

[30] U. Schmidt, K. Schelten, and S. Roth. Bayesian deblurring with integrated noise estimation. In *Proc. CVPR*, 2011.

[31] C. J. Schuler, H. C. Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *Proc. CVPR*, 2013.

[32] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf. Learning to deblur. *IEEE Trans. PAMI*, 38(7):1439–1451, 2016.

[33] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. *Proc. CVPR*, 2015.

[34] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[35] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Proc. NIPS*, 2012.

[36] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *Proc. ECCV*, 2010.

[37] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Proc. NIPS*, 2014.

[38] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *Proc. ICML*, 2015.

[39] L. Xu, S. Zheng, and J. Jia. Unnatural L0 sparse representation for natural image deblurring. In *Proc. CVPR*, 2013.

[40] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. *Proc. ICCV*, 2011.