Efficient Multi-scale Network with Learnable Discrete Wavelet Transform for Blind Motion Deblurring

Xin Gao*1,2, Tianheng Qiu*2,3,4, Xinyu Zhang1,2, Hanlin Bai1, Kang Liu1, Xuan Huang1,4, Hu Wei4, Guoying Zhang1,4, Huaping Liu2

*Equal Contribution †Corresponding Author

1China University of Mining & Technology-Beijing 2Tsinghua University 3University of Science and Technology of China 4Hefei Institutes of Physical Science, Chinese Academy of Sciences

Abstract

Coarse-to-fine schemes are widely used in traditional single-image motion deblur; however, in the context of deep learning, existing multi-scale algorithms not only require the use of complex modules for feature fusion of low-scale RGB images and deep semantics, but also manually generate low-resolution pairs of images that do not have sufficient confidence. In this work, we propose a multi-scale network based on single-input and multiple-outputs (SIMO) for motion deblurring. This simplifies the complexity of algorithms based on a coarse-to-fine scheme. To alleviate restoration defects impacting detail information brought about by using a multi-scale architecture, we combine the characteristics of real-world blurring trajectories with a learnable wavelet transform module to focus on the directional continuity and frequency features of the step-by-step transitions between blurred images to sharp images. In conclusion, we propose a multi-scale network with a learnable discrete wavelet transform (MLWNet), which exhibits state-of-the-art performance on multiple real-world deblurred datasets, in terms of both subjective and objective quality as well as computational efficiency. Our code is available on https://github.com/thqiu0419/MLWNet.

1. Introduction

Most current top-performing single-image blind deblurring algorithms are based on DNNs, which can be structurally categorized into single-scale [2, 10, 12, 28, 29, 42] and multi-scale approaches [3, 9, 13, 27, 37]. Compared to single-scale methods, multi-scale methods are built on the idea of moving from coarse-to-fine, and thus they decompose the challenging single-image blind deblurring problem into easier-to-solve sub-problems to restore the blurred image step-by-step.

Earlier DNNs exploiting the coarse-to-fine concept usually employ estimators of different scales to produce re-
features can be aggregated by simple feature fusion to ultimately obtain competitive results. That is, a DNN itself possesses the ability to learn effective features at different scales. This inspired us to design a multi-scale architecture that retains an original image as input and sequentially restores images at each scale in output stage. This is not only more theoretically sound, but also eliminates the time required to generate pairs of images of different resolutions, as well as the huge complexity increase brought about by fusion modules between RGB domain and deep features.

The coarse-to-fine algorithm has another inherent defect. During progressive restoration, the solution of the upper-level problem takes the solution of the lower-level as initialization [9]. Although this reduces the difficulty of solving the upper-level problem, due to the smaller resolution of lower-level spatial, the features transmitted upwards are semantically precise but spatially ambiguous, thus the restoration ability of multi-scale network in spatial details is limited. Hence, there is an urgent need to improve the quality of restoration of high-frequency detail part. A simple approach is to introduce a frequency domain transform as an alternative to a spatial domain transform. This would provide the algorithm with a choice and direct its attention to different frequencies. In this paper, we consider using the discrete wavelet transform for the following reasons:

1. Many recent state-of-the-art deblurring algorithms [10, 19, 42] have introduced the discrete Fourier transform (DFT) as a frequency prior, which provides information that helps the algorithm to identify and select high- and low-frequency components that need to be preserved during restoration. Compared to DFT, the discrete wavelet transform (DWT) is better suited to deal with images containing more abrupt signals [5].

2. Realistic blur and synthetic blur have significant distributional differences [24]. In the real world, due to the short exposure time of a camera, realistic blur has a specific directionality [29], i.e., the blur trajectory is regionally continuous; conversely, synthetic blur has an unnatural and discontinuous trajectory. To fully utilize the potential defocusing guidance brought by this trajectory continuity, we use 2D-DWT to reveal blur directionality, distinguish changes in the blur signal along different directions, and provide the algorithm with a reliable basis for deblurring through adaptive learning.

To make 2D-DWT fit the data distribution and feature layer space more closely, we implemented 2D-DWT with an adaptive data distribution by using group convolution, transferring feature space from the spatial domain into the wavelet domain, generating sub-signals with different frequency features and different directional features, and then constructing wavelet losses for them in order to constrain them by self-supervision. Experimental results demonstrate that the proposed method has advanced performance in terms of accuracy and efficiency (Fig. 1).

In summary, our contributions are as follows:

- We propose a single-input and multiple-output (SIMO) multi-scale baseline for progressive image deblur, which reduces the complexity of existing multi-scale deblurring algorithms and improves the overall efficiency of image restoration networks.
- We construct a learnable discrete wavelet transform node (LWN) to provide reliable frequency and direction selection for the algorithm, which promotes the algorithm’s restoration of the high-frequency components of edges and details.
- For the network to work better, reasonable multi-scale loss is proposed to guide pixel-by-pixel and scale-by-scale restoration. We also created reasonable self-supervised losses for the learnable wavelet transform to limit the learning direction of the wavelet kernel.
- We demonstrate the effectiveness of our algorithm under several motion blur datasets, especially real datasets, and obtain highly competitive results.

2. Related Work

Single-scale Deblurring Algorithm. Image deblurring have developed rapidly in recent years [2, 10, 11, 14, 39, 40]. To preserve richer image details, Kupyn et al. [11] viewed deblurring as a special case of image-to-image conversion, and for the first time utilized a GAN to recover images with richer details. To provide a high-quality and simple baseline, Chen et al. [2] proposed a nonlinear activation-free network that obtained excellent results. Zamir et al. [38] proposed a transformer with an encoder–decoder architecture that can be applied to higher-resolution images and utilizes cross-channel attention. Kong et al. [10] utilized a domain-based self-attention solver to reduce the transformer complexity and suppress artefacts. However, such single-scale algorithms estimate complex restoration problems directly and may require the design of restorers of higher complexity, which is suboptimal in terms of efficiency [9].

Multi-scale Deblurring Algorithm. Multi-scale motion deblur methods aim to achieve progressive recovery using a multi-stage or multi-sub-network approach. The multi-scale algorithm previously used in the field of image deblurring is based on MIMO [3, 13, 20, 21, 27]. Recently, Cho et al. [3] proposed MIMO-UNet, which employs a multi-output single decoder as well as a multi-input single encoder to simulate a multi-scale architecture consisting of stacked networks, thereby greatly simplifying complexity. Zamir et al. [37] designed a multi-stage progressive image recovery network to simplify the information flow between multiple sub-networks. As shown in Sec. 3.1, our proposed multi-scale approach further simplifies the structure of multi-scale networks and better balances the accuracy and complexity.
Figure 2. The overall architecture of the proposed MLWNet, the SEB is a simple module designed with reference [2], the WFB and WHB apply the LWN that implements the learnable 2D-DWT. In training phase, supervised learning is performed using $L_{\text{multi}}$ and self-supervised restraint of the wavelet kernel is performed using $L_{\text{wavelet}}$. In testing phase, only the highest scale restored images is output.

Application of Frequency in Deblurring. Frequency is an important property of images, in recent years frequency domain has been introduced into numerous DNNs to be solved for various tasks [10, 22, 25, 32, 35, 41]. Mao et al. [19] proposed DeepRFT, which introduced a residual module based on the Fourier transform into an advanced algorithm. Kong et al. [10] proposed FFTformer, which introduced the Fourier transform into a feed-forward network to effectively utilize different frequency information. Zou et al. [42] proposed SDWNet, which utilized frequency domain information as a complement to spatial domain information. Different from these methods, we introduce a learnable 2D-DWT capable of adapting to data distribution and feature space, more suitable not only for dealing with digital images rich in mutation signals, but also for dealing with real blurring.

3. Proposed Method

Our proposed MLWNet(Fig. 2) aims to explore an efficient multi-scale architectural approach to achieve high-quality blind deblurring of a single image. First, we designed a novel and highly scalable SIMO multi-scale baseline to address the performance bottleneck faced by partially multi-scale networks. It takes a single image as an input and gradually generates a series of sharp images from the bottom up. Then, we propose a learnable wavelet transform node (LWN) for image deblurring, and it enhances the proposed algorithm’s ability to restore detail information.

3.1. Multi-scale Baseline

Our multi-scale network maintains an encoder-decoder architecture, but retains the original image with the highest resolution as input, and restores sharp images of various scales sequentially in the output stage. In this way, we eliminate the complexity of the fusion module when dealing RGB images of different scales and with deep semantics.

For ease of understanding, we will introduce here the Encoder phase, multi-scale semantic Fusion phase (hereafter referred to as Fusion phase), and Decoder phase sequentially. As shown in Fig. 2, the Encoder stage consists of several gray Simple Encoder Blocks(SEB), where information flows top-down for feature extraction; the Fusion stage consists of several green Wavelet Fusion Blocks(WFB), where information flows bottom-up for semantic fusion at different scales; and the Decoder stage consists of several purple Wavelet Head Blocks(WHB), and the bottom-up flow of information enables gradual restoration. Compared with WHB, SEB replaces LWN with a $1 \times 1$ conv for channel scaling, and a $3 \times 3$ depth-conv for feature extraction.

The Encoder stage is responsible for full feature extraction, downsampling feature maps once after each block, and then passing the feature maps to the Fusion stage or the Decoder stage as needed, whose outputs $E_{i_{\text{out}}}$ to block of the $i-th$ layer can be denoted as:

$$E_{i_{\text{out}}} = \begin{cases} \phi_i (\text{embed}(x)), & i = i_{\text{max}} \\ \phi_i (E_{i'}_{\text{out}}), & \text{otherwise} \end{cases}$$

(1)

where $\phi_i$ represent the block of corresponding layer of Encoder. The Fusion stage adapts and fuses information from different scales of semantics produced in Encoder to generate intermediate representations with deep semantics and shallow details. The Fusion stage’s outputs $F_{i_{\text{out}}}$ to the block of $i-th$ layer is expressed as in Eq. 2, where $\delta_i$
represents the block of the corresponding layer of Fusion.

\[
F_{\text{out}}^i = \begin{cases} 
\delta_i (E_{\text{out}}^i + E_{\text{out}}^{i-1}), & \text{if } i = i_{\min} \\
\delta_i (E_{\text{out}}^i + E_{\text{out}}^{i-1}), & \text{otherwise}
\end{cases} 
\] (2)

The Decoder stage utilizes the information passed by both the Encoder and Fusion stages to progressively upsample the feature maps and generating pre-output feature maps at each scale separately. Output \( D_{\text{out}}^i \), to the block of the \( i \)-th layer is denoted as in Eq. 3. Here \( \xi_i \) represents the block of the corresponding layer of the Decoder.

\[
D_{\text{out}}^i = \begin{cases} 
\xi_i (E_{\text{out}}^i), & \text{if } i = i_{\min} \\
\xi_i (D_{\text{out}}^{i-1} + E_{\text{out}}^i), & i > i_{\min} \text{ and } i < i_{\max} \\
\xi_i (D_{\text{out}}^{i-1} + E_{\text{out}}^{i-1} + E_{\text{out}}^i), & i = i_{\max}
\end{cases}
\]

Each block in the Decoder is followed by a 3 \( \times \) 3 convolution to generate a restored image at each scale. To improve inference efficiency, all but the highest-scale outputs are generated in the training phase to compute the multi-scale loss. We note that this design also have a similar role with auxiliary head [36] in facilitating the model’s learning of the recovered uniform patterns.

For ease of understanding, the 1D-DWT case is given here first. For the input discrete signal \( t \), given the wavelet function \( \psi_{j,k} (t) = 2^j \psi(2^j t - k) \) for scaling factor \( j \), time factor \( k \), and scale function \( \phi_{j,k} (t) = 2^j \phi(2^j t - k) \), the decomposition of the input signal \( t \) at \( j_0 \) is given by:

\[
f(t) = \sum_{j>j_0} \sum_k d_{j,k} \psi_{j,k}(t) + \sum_k c_{j_0,k} \phi_{j_0,k}(t)
\] (3)

In this process, \( d_{j,k} = \langle f(t), \psi_{j,k}(t) \rangle \) represents the detail coefficients (i.e., high-frequency components), \( c_{j,k} = \langle f(t), \phi_{j,k}(t) \rangle \) represents the approximation coefficients (i.e., low-frequency components) so that the input signal can be disassembled step by step into a rich wavelet domain signal. In order to use wavelet transform in DNN, we introduce the analysis vectors \( \tilde{a}_0[k] = \langle \frac{1}{\sqrt{2}} \psi(\frac{1}{2}), \phi(t - k) \rangle \), \( \tilde{a}_1[k] = \langle \frac{1}{\sqrt{2}} \psi(\frac{1}{2}), \phi(t - k) \rangle \) to represent the high and low frequency filters respectively, then:

\[
c_{j+1,p} = \sum_k \tilde{a}_0[k - 2p] c_{j,k} \\
d_{j+1,p} = \sum_k \tilde{a}_1[k - 2p] c_{j,k}
\] (4)

According to Eq. 4, the decomposition of the original signal over the wavelet basis can be viewed as a recursive convolution of the original signal with a specific filter using a step size of 2. Similarly, the inverse wavelet transform can be realized by transposed convolution of two synthetic vectors \( \tilde{s}_0 \) and \( \tilde{s}_1 \). We extend the above situation to 2D by efficiently extracting axial high-frequency information using 2D discrete wavelet transform (2D-DWT), and designing learnable methods to make it adaptive to the data distribution and feature layers. Fig. 3(b) presents the construction process of the forward wavelet convolution kernel in 2D, which can be expressed as:

\[
\mathcal{F}_{ll} = \tilde{a}_0 \times \tilde{a}_0^T, \mathcal{F}_{lh} = \tilde{a}_0 \times \tilde{a}_1^T \\
\mathcal{F}_{hl} = \tilde{a}_1 \times \tilde{a}_0^T, \mathcal{F}_{hh} = \tilde{a}_1 \times \tilde{a}_1^T
\]

\[
\mathcal{K}_w = \text{cat} (\mathcal{F}_{ll}, \mathcal{F}_{lh}, \mathcal{F}_{hl}, \mathcal{F}_{hh})
\]

We set \( \tilde{a}_0 \) and \( \tilde{a}_1 \) as learnable filters; \( \mathcal{F}_{ll}, \mathcal{F}_{lh}, \mathcal{F}_{hl}, \) and \( \mathcal{F}_{hh} \) are low-frequency, horizontal high-frequency, vertical high-frequency, and diagonal high-frequency convolution operators obtained from vector outer products; The wavelet kernel \( \mathcal{K}_w \) is spliced from above convolution operators.

Then we implement learnable wavelet transform as a group convolution like Fig. 3(a), given a set of input feature maps \( X_{\text{in}} \in \{C, H, W\} \), its projection in wavelet domain \( X_{\text{out}} \in \{4C, H, W\} \) (low-frequency, horizontal high-frequency, vertical high-frequency, and diagonal high-frequency component) is generated through the wavelet
convolution. Similarly, the construction method of the inverse kernel and the forward propagation can be easily deduced, since they are inverse processes of each other.

Then, a natural question can then be posed: how to ensure the correctness of the wavelet kernel learning and ensure that it does not degrade into a general group convolution and cause performance degradation?

A common practice is to introduce the principle of perfect reconstruction [16, 34] to constrain adaptive wavelets, for a complex \( z \in \mathbb{C} \), given a filter \( \tilde{x} \) that applies to the \( z \)-transform, its \( z \)-transform can be expressed as \( X(z) = \sum_{n \in \mathbb{Z}} \tilde{x}(n)z^{-n} \). Then \( \tilde{a}_0, \tilde{d}_1, \tilde{s}_0, \tilde{s}_1 \) can be obtained as their corresponding \( z \)-transforms \( A_0, A_1, S_0, S_1 \), which must then be satisfied if perfect reconstruction is desired:

\[
A_0(z)S_0(z) + A_1(z)S_1(z) = 0 \tag{6}
\]

\[
A_0(z)S_0(z) + A_1(z)S_1(z) = 2 \tag{7}
\]

Eq. 6 is known as aliasing cancellation condition, which is used to remove aliasing arising from the downsampling. We will give a detailed derivation of Eq. 5 and Eq. 6 in the Appendix.

After wavelet convolution, to achieve frequency restoration in the wavelet domain, the input \( X \) wavelet domain component was separated into a separate dim, we use depth-wise convolution with an expansion factor of \( r \) for wavelet domain feature extraction and transformation after the wavelet positive transform, and \( 1 \times 1 \) convolution for channel expansion and scaling, and finally a learnable wavelet inverse transform is used to reduce the wavelet domain feature maps to the spatial domain for output, which constitutes the LWN. We follow the rules of [2] and design structurally similar Wavelet Head Block (WHB) and Wavelet Fusion Block (WFB), both with LWN as an important base module, but used for semantic aggregation and multiscale output at different scales, respectively; thus, WHB adds the recovery convolution branch after WFB.

### 3.3. Loss

**Multi-scale Loss.** Follow PSNR loss [2], we propose multi-scale loss function as the main loss of the algorithm for calculating the pixel difference between restored image and GT at each scale:

\[
L_{multis}(x, y) = \sum_{k=1}^{K} w_k \times L_{psnr}(x_k, y_k) \tag{8}
\]

where \( x, y \) denote the output and GT, respectively. \( k \) denotes the downsampling level, \( w_k \) represents the weights under the corresponding scale. Our design of \( w_k \) is based on the simple finding that we cannot produce absolutely accurate downsampled clear images, and that the accuracy gap is progressively enlarged with decreasing scales. To ensure that lower scale output layers do not negatively affect the final output, we empirically set the loss weight \( w_k \) to \( \frac{1}{k} \) for the corresponding k-scale.

**Wavelet Loss.** Both Eq. 6 and Eq. 7 can be easily converted losses optimized toward a minimum of 0, here we use the mean square error. Then, based on the convolvable nature of the \( z \) transform, it is possible to construct convolutionally equivalent substitutions for polynomial multiplications for the perfect reconstruction condition:

\[
L_{wavelet}(\theta_i) = \left( \sum_{k}^{N-1} \left( \langle a_0, s_0 \rangle_k + \langle a_1, s_1 \rangle_k - \hat{V} \left( \frac{z^{-k}}{2} \right) \right)^2 \right) \tag{9}
\]

where \( k \) denotes the position of the filter when it performs the convolution, \( \hat{V} \) is a vector with a center position value of two, and \( \theta_i \) denotes the constructed filter.

**Overall Loss.** Based on the multi-scale loss and wavelet loss, the loss function used in this study is:

\[
L_{total}(x, y) = L_{wavelet}(\theta_i) + L_{multis}(x, y) \tag{10}
\]

### 4. Experimental Results

#### 4.1. Datasets and Implementation Details

We evaluate our method on the realistic datasets RealBlur [23] and RSBlur [24], as well as the synthetic dataset GoPro [20], they are all composed of blurry-sharp image pairs. During training we use AdamW optimizer (\( \alpha = 0.9 \) and \( \beta = 0.9 \)) for a total of 600K iterations. The initial value of the learning rate is \( 10^{-3} \) and is updated with cosine annealing schedule. The patch size is set to 256 x 256 pixels and we followed the training strategy of [28, 38]. For data augmentation, we only use random flips and rotations. More experimental results can be found in the Appendix.

#### 4.2. Comparison with State-of-the-Art Methods

We have compared our method with several advanced deblurring methods and use the PSNR and SSIM to evaluate the quality of restored images.

**Evaluations on the RealBlur dataset.** Since our proposed method is driven by real-world deblurring, we first conduct comparisons on RealBlur [23]. The quantitative analysis results are shown in Tab. 1, the PSNR value of our method is 0.91dB and 0.49dB higher than state-of-the-art GRL on the RealBlur-J and RealBlur-R datasets respectively. Compared to other multi-scale architectures, our method has fewer model computational and running time while the performance is better. Fig. 4 shows visual comparison on RealBlur-J dataset, our method obtains restored text with...
The proposed method generates an image with clearer characters.

### Table 1. Quantitative evaluations on the RealBlur dataset [23]. The experimental results were trained under the corresponding datasets respectively, and average runtime is tested on 256 × 256 patches.

<table>
<thead>
<tr>
<th>Method</th>
<th>RealBlur-J PSNR</th>
<th>SSIM</th>
<th>RealBlur-R PSNR</th>
<th>SSIM</th>
<th>Avg. runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeblurGAN-v2 [12]</td>
<td>29.69</td>
<td>0.870</td>
<td>36.44</td>
<td>0.935</td>
<td>0.04s</td>
</tr>
<tr>
<td>SRN [27]</td>
<td>31.38</td>
<td>0.909</td>
<td>38.65</td>
<td>0.965</td>
<td>0.07s</td>
</tr>
<tr>
<td>MPRNet [37]</td>
<td>31.76</td>
<td>0.922</td>
<td>39.31</td>
<td>0.972</td>
<td>0.09s</td>
</tr>
<tr>
<td>SDWNet [42]</td>
<td>30.73</td>
<td>0.896</td>
<td>38.21</td>
<td>0.963</td>
<td>0.04s</td>
</tr>
<tr>
<td>MIMO-UNet+ [3]</td>
<td>31.92</td>
<td>0.919</td>
<td></td>
<td>-</td>
<td>0.02s</td>
</tr>
<tr>
<td>MIMO-UNet++ [3]</td>
<td>32.05</td>
<td>0.921</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DeepRFT+ [19]</td>
<td>32.19</td>
<td>0.931</td>
<td>39.84</td>
<td>0.972</td>
<td>0.09s</td>
</tr>
<tr>
<td>BANet [29]</td>
<td>32.00</td>
<td>0.923</td>
<td>39.55</td>
<td>0.971</td>
<td>0.06s</td>
</tr>
<tr>
<td>BANet+ [29]</td>
<td>32.42</td>
<td>0.929</td>
<td>39.90</td>
<td>0.972</td>
<td>0.12s</td>
</tr>
<tr>
<td>Stripformer [28]</td>
<td>32.48</td>
<td>0.929</td>
<td>39.84</td>
<td>0.974</td>
<td>0.04s</td>
</tr>
<tr>
<td>MSSNet [9]</td>
<td>32.10</td>
<td>0.928</td>
<td>39.76</td>
<td>0.972</td>
<td>0.06s</td>
</tr>
<tr>
<td>MSDI-Net [13]</td>
<td>32.35</td>
<td>0.923</td>
<td></td>
<td>-</td>
<td>0.06s</td>
</tr>
<tr>
<td>MAXIM-3S [30]</td>
<td>32.84</td>
<td>0.935</td>
<td>39.45</td>
<td>0.962</td>
<td>-</td>
</tr>
<tr>
<td>FFTformer [10]</td>
<td>32.62</td>
<td>0.933</td>
<td>40.11</td>
<td>0.973</td>
<td>0.13s</td>
</tr>
<tr>
<td>GRL-B [14]</td>
<td>32.82</td>
<td>0.932</td>
<td>40.20</td>
<td>0.974</td>
<td>1.28s</td>
</tr>
<tr>
<td>MLWNet-S</td>
<td>33.02</td>
<td>0.933</td>
<td></td>
<td>-</td>
<td>0.04s</td>
</tr>
<tr>
<td>MLWNet-B</td>
<td><strong>33.84</strong></td>
<td><strong>0.941</strong></td>
<td><strong>40.69</strong></td>
<td><strong>0.976</strong></td>
<td>0.05s</td>
</tr>
</tbody>
</table>

Table 2. Quantitative evaluations trained on the RSBlur dataset [24], the RealBlur-J dataset was used for testing only.

<table>
<thead>
<tr>
<th>Method</th>
<th>RSBlur PSNR</th>
<th>SSIM</th>
<th>RealBlur-J PSNR</th>
<th>SSIM</th>
<th>MACs(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRN [27]</td>
<td>32.53</td>
<td>0.840</td>
<td>29.86</td>
<td>0.886</td>
<td></td>
</tr>
<tr>
<td>MIMO-Unet [3]</td>
<td>32.73</td>
<td>0.846</td>
<td>29.53</td>
<td>0.876</td>
<td></td>
</tr>
<tr>
<td>MIMO-Unet+ [3]</td>
<td>33.37</td>
<td>0.856</td>
<td>29.99</td>
<td>0.889</td>
<td></td>
</tr>
<tr>
<td>MPRNet [37]</td>
<td>33.61</td>
<td>0.861</td>
<td>30.46</td>
<td>0.946</td>
<td></td>
</tr>
<tr>
<td>Restormer [38]</td>
<td>33.69</td>
<td>0.863</td>
<td>30.48</td>
<td>0.891</td>
<td></td>
</tr>
<tr>
<td>Uformer-B [33]</td>
<td>33.98</td>
<td>0.866</td>
<td>30.37</td>
<td>0.899</td>
<td></td>
</tr>
<tr>
<td>SFNet [4]</td>
<td>34.35</td>
<td>0.872</td>
<td>30.26</td>
<td>0.897</td>
<td></td>
</tr>
<tr>
<td>MLWNet-B</td>
<td><strong>34.94</strong></td>
<td><strong>0.880</strong></td>
<td><strong>30.53</strong></td>
<td><strong>0.905</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Quantitative evaluation for generalizability shows the results of models trained on the RealBlur-J dataset and tested on the RSBlur dataset, MACs are measured on 256 × 256 patches.

The proposed method makes textures and structures clearer.

**Evaluations on the GoPro dataset.** We conduct an exen-
The deblurring performance of the proposed method in low-light is impressive, and the recovery of characters and texture structures far exceeds other advanced methods.

Figure 6. Visual comparisons on the GoPro dataset [20]. Our method better preserves texture information without sharpening.

Table 4. Quantitative evaluations trained and tested on the GoPro dataset [20]. Our proposed MLWNet obtains competitive results with a combination of time efficiency and accuracy.

Single-scale vs. Multi-scale. Given that each node of the baseline network is composed of SEB, we have deformed the multi-scale and single-scale according to the architecture. Tab. 5 shows that the multi-scale architecture improves the PSNR and SSIM values to varying degrees, proving that

5. Analysis and Discussion

In this section, we provide a more in-depth analysis and show the contribution of each component of the proposed method. We perform 20w iterations on the RealBlur-J dataset for ablation with a batch size of 8 to train our method using a version of the model with a width of 32, and the results are shown in Tab. 6.

Table 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>GOPRO Avg. runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
</tr>
<tr>
<td>DeblurGAN-v2 [12]</td>
<td>29.55</td>
</tr>
<tr>
<td>SRN [27]</td>
<td>30.26</td>
</tr>
<tr>
<td>DMPHN [39]</td>
<td>31.20</td>
</tr>
<tr>
<td>SDWNNet [42]</td>
<td>31.26</td>
</tr>
<tr>
<td>MPRNet [37]</td>
<td>32.66</td>
</tr>
<tr>
<td>MIMO-UNet+ [3]</td>
<td>32.45</td>
</tr>
<tr>
<td>DeepRFT+ [19]</td>
<td>33.23</td>
</tr>
<tr>
<td>MAXIM-3S [30]</td>
<td>32.86</td>
</tr>
<tr>
<td>Stripformer [28]</td>
<td>33.08</td>
</tr>
<tr>
<td>MSDI-net [13]</td>
<td>33.28</td>
</tr>
<tr>
<td>Restorer [38]</td>
<td>33.57</td>
</tr>
<tr>
<td>NAFNet [2]</td>
<td>33.69</td>
</tr>
<tr>
<td>FFTFormer [10]</td>
<td>34.21</td>
</tr>
<tr>
<td>GRL-B [14]</td>
<td>33.93</td>
</tr>
<tr>
<td>MLWNet-B</td>
<td>33.83</td>
</tr>
</tbody>
</table>
the SIMO strategy we employ helps to eliminate motion blur to some extent. As shown in Sec. 3.1, we use SIMO for coarse-to-fine restoration, and only use multiple outputs to calculate multi-scale losses during training.

**WFB and WHB.** We default to using WFB and WHB via $L_{\text{wavelet}}$ statute in this section, we also proved the effectiveness of LWN, because LWN is a subpart of the first two. After applying them, the PSNR index improved by 0.25dB compared to the multi-scale baseline, which shows that LWN and its extension modules can easily help multi-scale algorithms improve detail recovery capabilities. Fig. 7 shows the four types of feature maps generated after forward learnable wavelet convolution. We can see that the high and low frequency information of the input feature map are mixed together, and after wavelet group convolution, the network exerts different attention on the feature information in different directions or frequencies.

**Effectiveness of $L_{\text{wavelet}}$.** By adding unreduced WFB and WHB to the baseline without using wavelet loss, the performance improvement of the baseline is minimal and does not break the bottleneck of multiscale detail recovery, which suggests that the wavelet convolution degenerates into a general group convolution that only serves to deepen the network, proving that merely deepening the network is ineffective for the restoration of detail information.

**Table 5. Comparison in various input and output modes.**

<table>
<thead>
<tr>
<th>Method</th>
<th>SISO</th>
<th>MIMO</th>
<th>SIMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR/SSIM</td>
<td>32.29/0.924</td>
<td>32.19/0.928</td>
<td>32.37/0.929</td>
</tr>
<tr>
<td>MACs(G)</td>
<td>19.24</td>
<td>21.83</td>
<td>19.29</td>
</tr>
</tbody>
</table>

Table 6. Ablation study on components of the proposed MLWNet. We set the baseline network to use SEB in its entirety, and models that do not use SIMO will represent single scales using SISO.

**Table 7. Performance comparison at different noise difference levels, where L3 contains the noise difference mean.**

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>L5</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoPro</td>
<td>34.81</td>
<td>34.66</td>
<td>33.76</td>
<td>33.19</td>
<td>32.63</td>
</tr>
<tr>
<td>RealBlur-J</td>
<td>33.92</td>
<td>33.81</td>
<td>33.97</td>
<td>33.93</td>
<td>33.54</td>
</tr>
</tbody>
</table>

Finally, there is a large difference in noise levels between synthetic blur and real blur. We follow [1] to calculate the noise level of each testset, and divided the noise difference between clear images and blurred images of GoPro and RealBlur-J into 5 levels for separate testing. As shown in Tab. 7, we note that as the noise difference increases, our method maintains good performance on RealBlur-J, while the performance drops fast on GoPro. This shows that our method is robust to the noise of real blur, and that there are differences in the noise between synthetic blur and real blur.

**6. Conclusion**

In this paper, we propose a SIMO-based multi-scale architecture to achieve efficient motion deblurring. We have developed a learnable discrete wavelet transform module, which not only improves the algorithm’s ability to recover details, but is also more applicable to the real world. In addition, we construct a reasonable multi-scale loss to guide the recovery of blurred images pixel by pixel and scale by scale, and constrain the learning direction of the wavelet kernel with self-supervised loss to achieve better image deblurring. Through extensive experiments on multiple real and synthetic datasets, we demonstrate that it outperforms existing state-of-the-art methods in terms of quality and efficiency of image restoration, especially with optimal deblurring performance and generalization in real scenes.
References


conference on applications of computer vision, pages 2149–2159, 2022. 3


[37] 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, pages 14821–14831, 2021. 1, 2, 6, 7


