Multipath Refinement Fusion Network for Efficient Image Deblurring

Appendix

Convolutional neural networks (CNNs) have achieved great success in image process and computer vision. Image deblurring problems addressed by CNNs can achieve better prediction accuracy than those that use traditional methods. CNNs, especially multi-scale deep CNNs, nonetheless require substantial memory and computational resources to perform image deblurring, thus hindering their deployment in real-time applications. Meanwhile, computation-efficient networks lack the capability to deal with large-scale datasets and thus cannot generate accurate restoration results in many cases. To this end, we propose an efficient and accurate deep learning framework for image deblurring named MRFNet with three main features. 1) The framework is the first of its kind to utilize the multipath refinement fusion (MRF) network for image deblurring. It is intensively reconstructed based on RefineNet. 2) The MRFNet combines lightweight convolution and residual connection as a means of enhancing model performance. 3) The designed scale refinement loss function accelerates the training speed of the MRFNet. Comparative experiments on two popular datasets GOPRO and VisDrone have been conducted to demonstrate the effectiveness of proposed method. The experimental results indicate that MRFNet can achieve state-of-the-art deblurring results at a faster speed than the other deblurring models.

Nowadays deep learning methods shows great potential for computer vision tasks. Convolutional neural networks are used to extract features develops progressively and rapidly for image restoration. Among them, current deblurring methods lack sharp edge reconstruction and meaningful region deblur kernel, which constrains the further processing of image restoration. Therefore, we design the multipath refinement fusion backbone targeting to learning multi-feature information automatically. The multi-path refinement fusion network can extract edge information and sharp features simultaneously, refine the patches iteratively, locate different blurry causes wisely and restore low-quality and extreme-blurry images effectively. In addition, it not only learns edge information but also employs contextual attention modules to search, locate and deblur in a coarse-to-fine manner. At last, several optimal strategies are adopted to enhance the performance and synthesis loss function is proposed to guarantee the sharp and meaningful deblurring.

III. Model Design and Implementation

An MRFNet has been extensively constructed to ensure the balance between accuracy and speed, as these aspects are currently lacking in the existing studies. We first exploit the recurrent and multiscale strategies to train the network. Then, a structure with a branch depth and fusion unit is built on the basis of the lightweight process, the MRF unit, and the remote residual connection. Finally, a scale refinement loss function is used to train the network in a coarse-to-fine manner.

A. Multiscale and Recurrent Learning

Instead of stacking the encoder and decoder processes to directly perform deblurring refinement, the recurrent and multiscale learning strategies were applied in this study. The basic idea of the multiscale learning strategy is for the top network to extract features from the large-coarse scale maps and the bottom network upsampling results. Meanwhile, in the recurrent learning strategy, the bottom layer acquires fusion information from the small refinement maps and the top feedback. In our work, the two strategies are combined by designing four refinement paths to extract features in different scales instead of directly predicting the whole deblurred image. In our method, the top network only needs to focus on learning highly nonlinear residual features, which is effective in restoring deblurred images in a coarse-to-fine manner. The architecture of the proposed MRFNet is shown in Fig. 2.

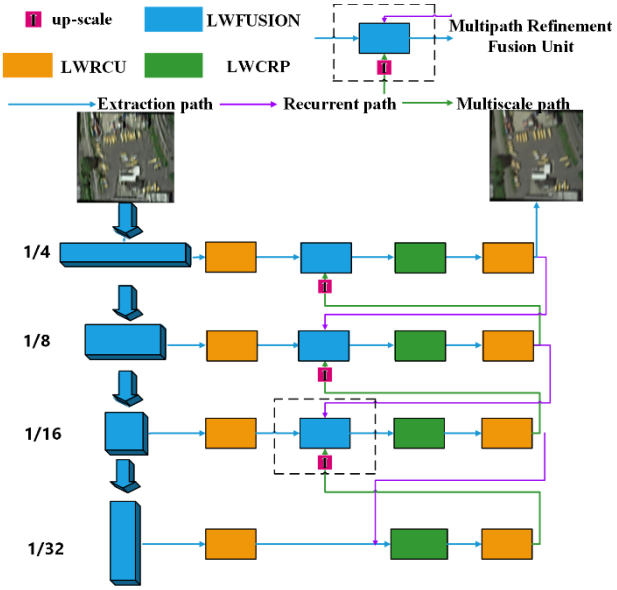


Fig. 2: MRF framework. The image is separated into different scales from top to bottom. Blue line refers the extraction path of extracting features from scales. Then, the blocks of MRF fuse the recurrent last-round results (purple line) and the upsampling feature maps (green line) as a single refinement process. The total four refinement paths finally compute the loss in the scale refinement loss function, then the best deblur results are obtained.

In the multipath input stream illustrated in Fig. 3(d), the upper MRFNet layer takes the left and right images as the input and processes the deblur datasets into a total of four scales, i.e., k is from 2 to 4.

The four-scale blur feature maps are denoted as b_k, while the refinement results are denoted as l_k. First, the k level of the multipath input stream concatenates the same scale feature maps b_k and the upsampled feature maps l_{k+1} as middle feature maps, which is denoted as c_k.

$$c_k = b_k \oplus l_{k+1} \ (2 \le k \le 4) \qquad (1)$$

Then, the fusion unit adds both c_k and the last-round results l_{k-1} as the final result, which is denoted as l_k. This

process briefly describes how the refinement fusion path works. The whole process can be calculated as

$$l_k = c_k + l_{k-1} \ (2 \le k \le 4) \qquad (2)$$

B. *Lightweight Process and MRF*

A large number of parameters and floating-point operations of the original MRF network originates from the commonly used 3 × 3 convolution. Therefore, we focus on replacing these elements with simpler counterparts without compromising performance.

The original design of an MRF network is to use an encoder–decoder structure equipped with four feature extraction and down-sampling layers. Each path has a fusion unit. The basic block uses 3 × 3 convolution, which we call the fusion unit. Here, the 1 × 1 fusion unit in Fig. 3(a) is replaced with 3 × 3 convolution.A chained residual pool (CRP) is also considered to naturally illustrate why the lightweight process works and how the former three units are reshaped. The lightweight process is applied to the CRP unit by substituting the 5 × 5 and 3 × 3 convolutions with the 5 × 5 and 1 × 1 convolutions in Fig. 3(b).

The refinement path adopts a convolutional layer with a stride of 1 followed by a convolution layer with a stride of 2 such that they consistently shrink the feature map size by half. The two convolution layers act as a residual connection unit (RCU). Two RCUs are installed in the encoder, while three RCUs are installed in the decoder. All of the blocks use 1 × 1, 3 × 3, and 1 × 1 convolutions compared with those in the RCU that use 3 × 3 and 3 × 3 convolution. We call the two convolution layers as the lightweight residual connection unit (LWRCU), as illustrated in Fig. 3(c).

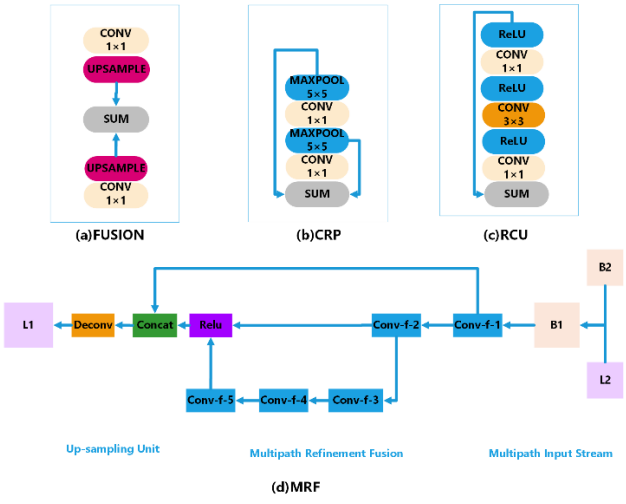


Fig. 3: Different parts of the network. (a) Fusion unit, (b) improved CRP module, (c) lightweight network structure of RCU, and (d) MRF unit.

TABLE I: Specific parameters of the MRFNet.

Network	Kernel	Stride	Padding	Network	Kernel	Stride	Padding
Conv1	5×5×32	1	2	conv_r2_m2	1×1×128	1	1
Conv2	1×1×64	1	1	conv_r2_m3	3×3×128	1	1
Conv3	5×5×128	2	2	conv_r2_m4	1×1×128	1	1
Conv4	1×1×128	1	1	deconv2	4×4×64	1	2
Conv5	3×3×256	1	2	conv_r3_1	3×3×64	1	1
Conv6	1×1×256	1	1	conv_r3_m1	3×3×64	1	1
Conv7	3×3×256	1	2	conv_r3_m2	1×1×64	1	1
Conv8	1×1×256	1	1	conv_r3_m3	3×3×64	1	1
conv_r1_1	3×3×256	1	1	conv_r3_m4	1×1×64	1	1
conv_r1_m1	3×3×256	1	1	deconv3	4×4×32	1	2
conv_r1_m2	3×1×256	1	1	conv_r4_1	3×3×32	1	1
conv_r1_m3	3×3×256	1	1	conv_r4_m1	3×3×32	1	1
conv_r1_m4	3×1×256	1	1	conv_r4_m2	3×3×32	1	1
deconv1	4×4×128	1	2	conv_r4_m3	1×1×32	1	1
conv_r2_1	3×3×128	1	1	conv_r4_m4	3×3×32	1	1

Intuitively, a convolution with a relatively large core size is designed to increase the size of the receiving field (as well as the global context coverage). The 1 × 1 convolution can only locally transform the features of each pixel from one space to another. Here, we empirically prove that the replacement with 1 × 1 convolution would not weaken network performance. Particularly, we replace the 3 × 3 convolution in the CRP and the fusion block with a 1 × 1 counterpart, and we modify the RCU to LWRCU with a bottleneck design, as shown in Fig. 3(c). using this method, we can reduce the number of parameters by more than 50% and the number of triggers by more than 75% (Table I). The convolutions have been shown to reduce considerable computation without sacrificing performance.

We also enhance the MRF unit illustrated in Fig. 3(d). Deep residual networks obtain rich feature information from multi-size inputs. The residual block originally derived in for the image classification tasks is widely used to learn robust features and train deeper networks. Residual blocks can well address vanishing gradient problems. Thus, we replace the connection layer with the MRF unit.

Here, the MRF is specifically designed as a combination of multiple convolution layers (conv-f-1 to conv-f-5), and each convolution layer is followed by a ReLU activation function. Conv-f-2 uses feature maps generated by Conv-f-1 to generate more complex feature maps. Similarly, conv-f-4 and conv-f-5 continue to use the feature map generated by conv-f-3 for further processing. Finally, the feature maps obtained from multiple paths are fused together. The specific calculation expression is as follows:

$$y = f_2(f_1(x)) + f_4\left(f_3\left(f_2(f_1(x))\right)\right) \qquad (3)$$

where *f*, *x* and *y* represent the convolution operation, the characteristic graph of the input, and the characteristic graph of the output, respectively.

We construct a residual connection in each path of the MRFNet. In the process of forward transmission, the remote residual connections transmit low-level features, which are used to refine the visual details of the coarse high-level feature maps. The residual connections allow the gradients to propagate directly to the early convolution layers, thus contributing to effective end-to-end training.

We set the number of paths from 1 to 6 for the multipath process. The operation takes up the least parameters when the paths are equal to 3, whereas the best performance is achieved when the paths are equal to 4. When the number of paths is less than 3, the extracted features are not accurate, and the training loss remains at a high level all of the time. When the number of paths exceeds 4, the deblurring encounters severe performance degradation. To this end, we choose the four-path refinement setting as the final backbone.

3.3.3 edge reconstruction loss

Generating edge information is very important for reconstructing images due to the sharp background is beneficial for the refinement of different blur kernels.

In addition, the ground truth images are preprocessed into grayscale images for further edge feature extraction and sent to discriminator for the comparable benchmark. The generator G produces various generative edge maps for the discriminator D to judge the realness of the generation.

$$\min_{G_e} \max_{D_e} L_{G_e} = \min_{G_e} \left(a_{adv,1} \max_{D_e} \left(L_{adv,1} \right) + a_{FM} L_{AM} \right) \qquad (3)$$

Thanks to the edge maps, we can classify the objects into N categories, each category has a different blur kernels to refine the blur features for specific objects.

The following is the blur class: motion blur and Gaussian blur, which are the two main blurry resources. For instance, the Figure 1 shows the car running fast on the streets which causes the motion blur, and the building is the far from the lens which causes the gaussian blur. This framework could find and locate the general objects and apply different blur kernels through deep learning training process. And then deblurring the specific stuffs into sharp ones aided by the edge generation modules and the contextual attention mapping.

Input blur and ground truth pairs, then the edge generative networks predict the structure of the whole picture, then the pretrained classify networks preprocessed the edge feature information to make sure the location and the class associate with the deblur kernels. This process as the human attention deblurring methods: broad-view, location, classification, deblurring.

The first step broad-view for edge abstract is illustrated above, the next step is classification. The process is simple: For a given image, $g_l(a,b)$ is the spatial information in the l-th layer. Then, G_l represents the sum of the $g_l(a,b)$.Thus, for a specific obkect class, the input $\sum A_l\, G_l$ is the input of the softmax function. A is the weight corresponding to class and predicts the essential level of Gl for the object class. Finally, the Q is the output of the softmax function, it is denoted as $\frac{exp(S)}{\sum_e exp(S)}$.

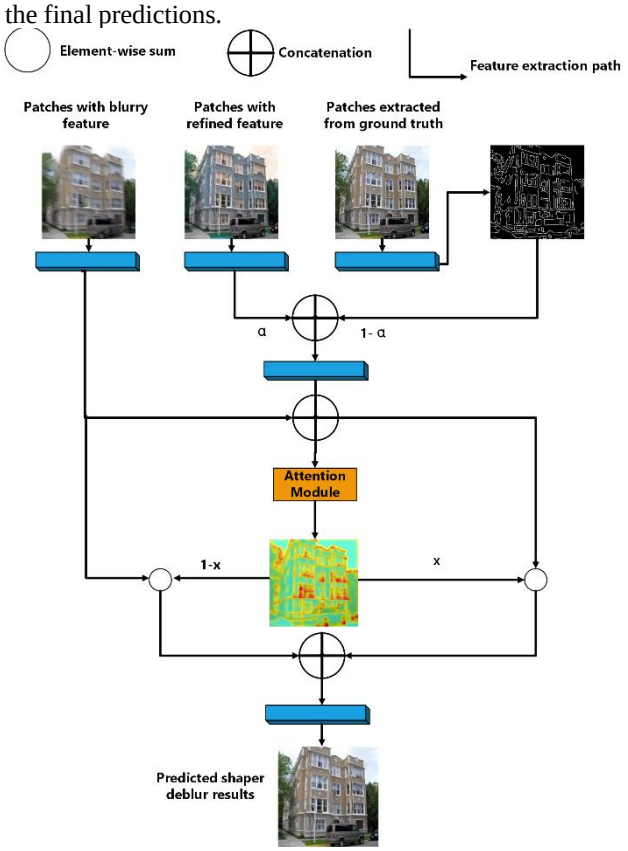
The score S is dines as follows:

$$S = \frac{\sum A \sum g_l(a,b)}{\sum(a,b) \sum A_l \sum g_l(a,b)} \qquad (4)$$

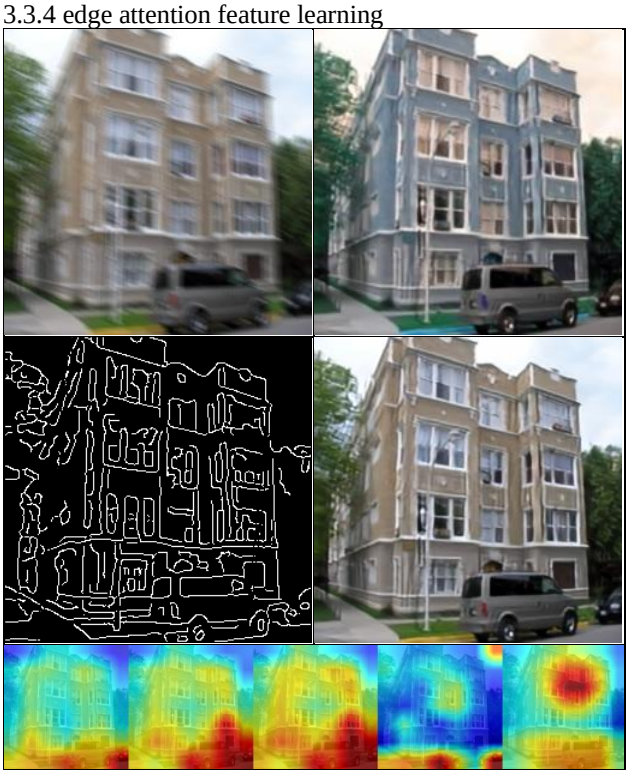
The score of the global average pooling predicts the importance of the location of (a,b) leading to the classification of an blurry object in the image.From the figure 18 wecould conclude that changing the receptive field could result in different contextual attention results. When the receptive field is large, the whole object is perceptive as a whole. When the receptive field is small, we could find that each part of the object is concerned and the texutre is located in detail.

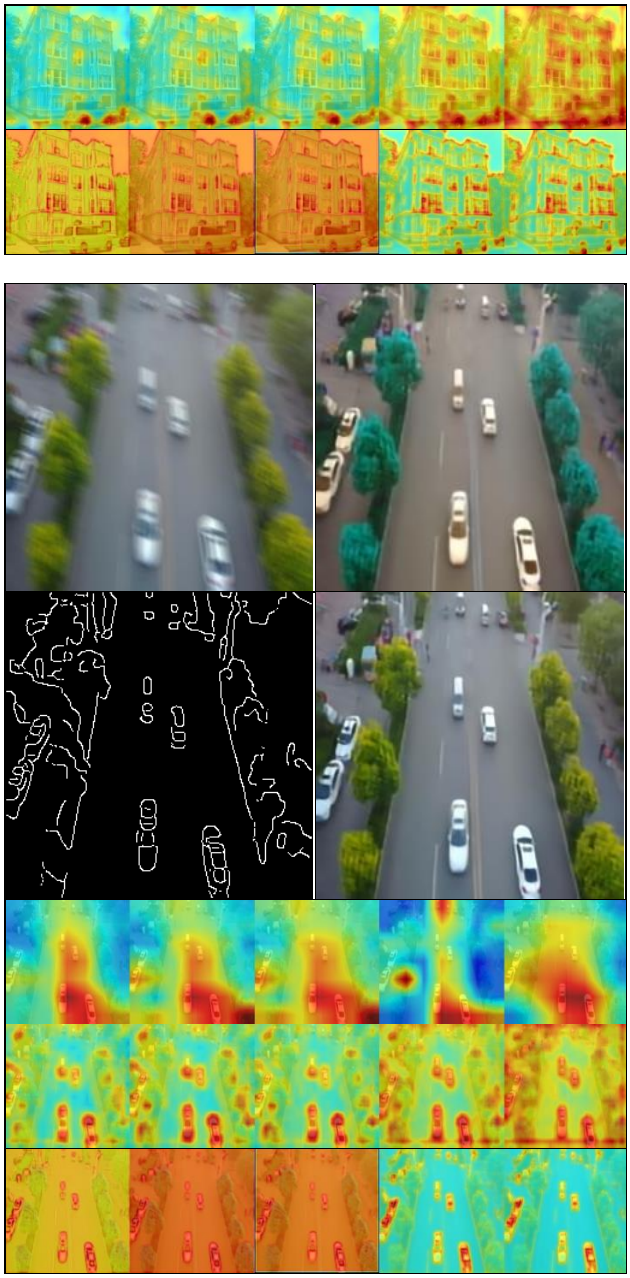
Finally, the data flow from the edge feature extraction and locate the contextual attention specifically, we know the structure information, the predicted object and blurry potential class. Then we use the deblurring feature prior network to deblur the images and produce sharper ones. Through this way, we could restore the image for applying different blur strategies in different images regions, so the target is more specific, hence the performance is excellent and the reconstruction of the object structure is meaningful and vivid. The whole feature extraction and fusion illustration could be abstracted in Figure 20.

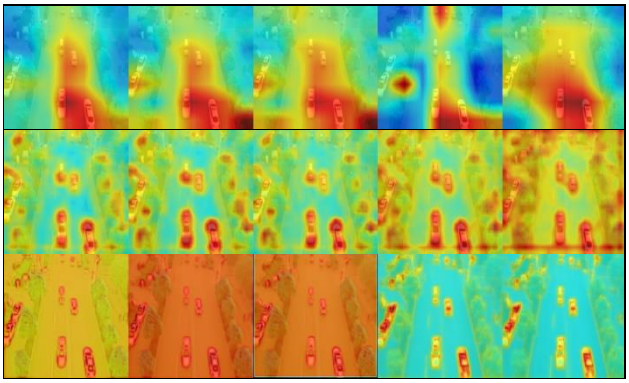
In the last stage, the task is to fuse the deblurring features and the edge feature from outputs, to make the final restored frame. During the training process, the process accepts three inputs: patches with blurry feature extracted from initial encoder, patches with refined feature extracted from initial decoder, and patches extracted from the ground truth. In the progressive weighted training process, first, the edge feature is extracted from the patches with refined feature and patches from ground truth, the a is a hyper parameter which is set to 0 initially to control the proportion of the refined resource. Second, the blurry patches and the mixed edge feature patches are combined the send to the contextual attention module, the contextual attention module using the softmax function to predict the foreground and generate the preliminary activated heatmaps. At last, the α is set to 1 and the deblur refined feature patches are sent to attention module at the middle process of training process and predicted by the attention module once again. The results are compared with the synthesis loss function between predicted deblurring results and the patches with sharp feature. So at the beginning of the training deblurring feature refines the input of blurry images and benefits the edge feature extraction, then, in the middle of the training process, deblurring feature and edge feature make a fusion by controlling parameter a. and each path which contains different scale of double feature patches are refined and matched by multipath context attention module with the activated heatmaps to inference the final predictions.



3.3.4 edge attention feature learning







C. Loss Function Design

Given a pair of sharp and blurred images, MRFNet takes these images as the input and produces four groups of feature maps in different scales. Assume that the input image size is H × W. The four scales of the feature maps are H/4 × W/4， H/8 × W/8, H/16 × W/16, and H/32 × W/32. To train the MRFNet in an end-to-end manner, we adopt the L2 loss between the predicted deblurring result map and the ground truth as follows:

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N} ||x_s^i - F(x_l^i)||^2 \qquad (4)$$

where θ is the parameter set, x_s is the ground truth patch, and F is the mapping function that generates the restored image from the N-interpolated LR training patches x_l. Here, the patch size is defined at different levels.

The scale refinement loss function is useful in learning the features in a coarse-to-fine manner. Each refinement path has a loss function that can be used to evaluate the training process. When others adopt a single final L2 loss function, our scale refinement loss function computes the results in different scales, which leads to a much faster convergence speed and an even higher inference precision.

$$L_{final} = \frac{1}{2K}\sum_{k=1}^{K} \frac{1}{c_k w_k h_k} ||L_k - S_k||^2 \qquad (5)$$

where L_k represents the model output of the scale level k, and S_k is the k-scale sharp maps.

The loss at each scale is normalized by the number of channels C_s, width W_s, and height H_s.

IV. Performance Evaluation

In this section, we compare MRFNet to the recently adopted methods of DeblurGAN, DMPHN, and SIUN in terms of accuracy and time efficiency.

A. Experimental Setup

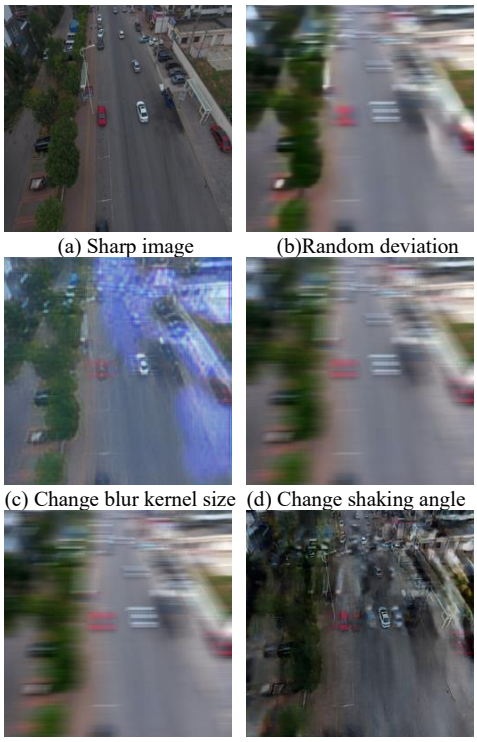
We implement our MRFNet by using Caffe. The model is trained with Adam (β_1 = 0.9, β_2 = 0.999). In the training process, the images are randomly cropped to 256 × 256. The batch size of 16 is used for the training in four NVIDIA RTX2080Ti GPUs. At the beginning of each epoch, the learning rate is initialized as 10^{-4} and subsequently decayed by half every 10 epochs. We train 70 epochs for VisDrone and 50 epochs for GOPRO.

In terms of time efficiency, we evaluate the inference time of the existing state-of-the-art CNNs on RTX2080Ti GPUs with 11 GB of memory.

B. Dataset

We use two public datasets to train and evaluate the performance of MRFNet. The first dataset is VisDrone,

which provides synthetic blur techniques. The second one is GOPRO, which is captured in real-world scenarios.



(a) Sharp image (b)Random deviation (c) Change blur kernel size (d) Change shaking angle (e) Change shaking length (f)Motion blur

Fig. 4: Overview of dataset augmentation by changing the parameters. The first column shows a sharp image, an image blurred by changing the blur kernel size, and an image blurred by altering the blur shaking length. The second column shows the blurred image added with random standard deviation, in which the image is blurred by changing the shaking angle. The last image is generated by real motion blur.

The benchmark dataset of VisDrone consists of 288 video clips formed by 261,908 frames and 10,209 static images. The dataset, which was captured by various drone-mounted cameras, covers a wide range of elements, including location (taken from 14 different cities separated by thousands of kilometers in China), environment (urban and country), objects (pedestrian, vehicles, or bicycles.), and density (sparse and crowded scenes). The dataset was collected using various drone platforms (i.e., drones of different models) in different scenarios under various weather and lighting conditions.

We produce the VisDrone blur dataset by using different methods, including Gaussian blur. Several data enhancement techniques can be utilized to prevent our network from overfitting. In terms of geometric transformation, the images are flipped horizontally, vertically, and rotated at random angles. For the colors, the RGB channels are replaced randomly. For the image color degradation, the saturation in the HSV color space is multiplied by a random number in [0,5]. In addition, Gaussian blur is added to the blurred image. Finally, to ensure that our network is robust to noise of different intensities, the standard deviation of noise is also randomly sampled from the Gaussian distribution N (0-1).

The GOPRO dataset contains images with changeable motion blurs. GOPRO cameras can record sets of sharp information that are integrated over time to generate blurred images rather than using a preset kernel to blur the generated images. When the camera sensor of GOPRO receives light during exposure, it accumulates a clear image stimulus each time, resulting in a blurred image.

C. *Comparative Experiments*

We conduct comparative experiments with DeepDeblur，DeblurGAN,DeblurGANv2, DeepDeblur, DMPHN, and SIUN to verify the performance of our model. The proposed MRFNet can achieve state-of-the-art performance compared with SIUN on VisDrone. The values of PSNR and SSIM are much higher than DeblurGAN and DMPHN, suggesting the advantage of our method in handling Gaussian blurs. Moreover, our method performs better than SIUN and DMPHN and even much better than DeblurGAN in dealing with the motion blurs of GOPRO. The trends in Table II prove the superiority of the MRFNet framework on the PSNR and SSIM values. Due to the Visdrone dataset adding extreme blurry and distorted texture augmentation, other methods shows great weakness in SSIM, which means they lack capacity in restoring severe structure information missing and extreme blurry image deblurring.

TABLE II. Testing results of the blurred image datasets and their PSNR and SSIM values.

Method	GOPRO		VisDrone	
	PSNR	***SSIM***	***PSNR***	***SSIM***
DeepDeblur	29.4237	0.761372	27.14940	0.539367
deblurGAN	28.22642	0.747912	28.29447	0.609642
deblurGANv2	32.19638	0.87114	28.43967	0.614876
DMPHN	34.21846	0.898285	28.54136	0.526301
SIUN	34.46135	0.900913	28.28039	0.543417
Our model	**34.63429**	**0.907881**	**29.40845**	**0.862474**

D. *Ablation Experiments*

The original MRF network used as the benchmark is denoted as RefineNet. From this original RefineNet, we add the lightweight process to the benchmark and denote it as LWRefineNet. Then, we add the remote residual connection to the refinement path and residual connection to fusion unit and denote it as RCRefineNet. Finally, we combine lightweight and residual strategy to the benchmark network and define it as MRFNet.

TABLE III. Memory consumption of graphics cards by different methods (Model= MRFNet).

Method	GOPRO	VisDrone
	Network(MB)+Batch(8)	***Network(MB)+Batch(16)***
DeepDeblur	**6311**	7930
deblur GAN	**4538**	6012
deblurGANv2	**6861**	8107
DMPHN	6541	7329
SIUN	8399	8561
Our model	5452	**5898**

As shown in Table III, DeblurGAN requires the least GPU memory usage at 4538 MB, while our method requires slightly higher GPU memory usage than DeblurGAN in GOPRO. For the VisDrone dataset, our network consumes the least GPU memory for the batch size of 16. The lightweight process can reduce the parameters of the model, contributing to the low-memory requirement.

TABLE IV: Average time of inferring images.

Method	GOPRO	VisDrone
	InferTime(s)	***InferTime(s)***
DeepDeblur	2.427	2.362
Deblur GAN	2.346	2.144
deblurGANv2	2.528	2.663
DMPHN	1.886	0.764
SIUN	0.684	0.357
LWRefineNet	**0.494**	**0.319**

As shown in Table IV, LWRefineNet is the fastest method in terms of the time of loading the network model and the inferences. The inference is run on GTX1650 4G GPU. The image size from GOPRO is 1280 × 768, while that from VisDrone is 256 × 256.

TABLE V: Quantitative numerical results on PSNR and SSIM.

Method	GOPRO		VisDrone	
	PSNR	***SSIM***	***PSNR***	***SSIM***
RefineNet	34.17826	0.894369	28.73991	0.854758
LWRefineNet	34.21445	0.906998	29.24461	0.860164
RCRefineNet	34.39430	0.903012	29.03971	0.858601
MRFNet	**34.63429**	**0.907881**	**29.40845**	**0.862474**

As shown in Table V, the LWRefineNet and RCRefineNet perform slightly better than RefineNet. MRFNet shows the best numerical results.

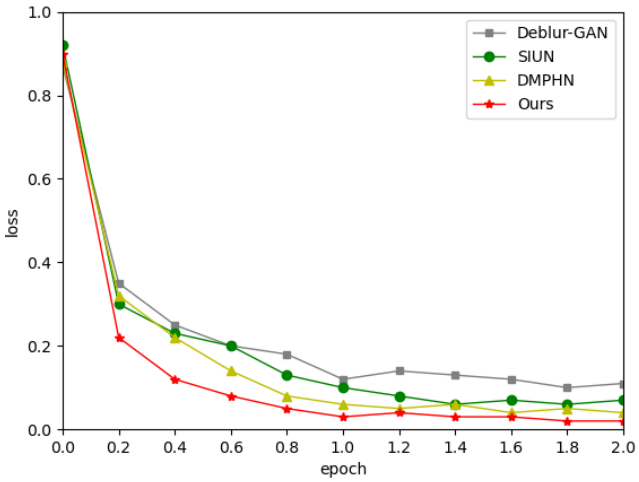


Fig. 5: Training loss of the four methods. Only the first two epochs are shown.

As shown in Fig. 5, our multi-scale refinement loss function takes each sub-task as an independent component within a single unified task, allowing the training process to converge more rapidly and perform better than those of the other methods. The training losses of the other approaches decrease remarkably at the first round and consistently stay at 6% with a smooth trend in the following training courses. Our method, aided by the loss weight scheduling technique, exhibits a dramatic downward trend and remains at approximately 4%. The model accuracy improvements (approximately 10% to 21%) resulting from the multiple rounds of training for the four loss weight groups verify the good convergence and advantages of our method's training strategy.

(a) Input image

(b) DeblurGan

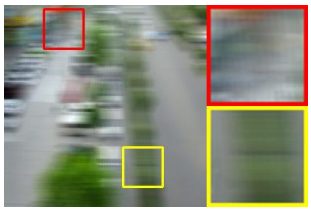
(c) DMPHN

(d) SIUN

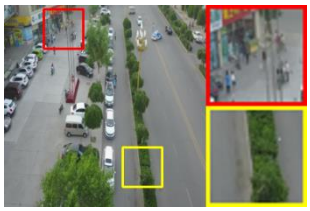
(e) Our method

Fig. 6: Visual effects of different methods. From top to bottom: blurred image, results of DeblurGAN, DMPHN, SIUN and ours. The left images are global deblur results, while local restoration details are shown on the right. Our results show clear object boundaries without artifacts.

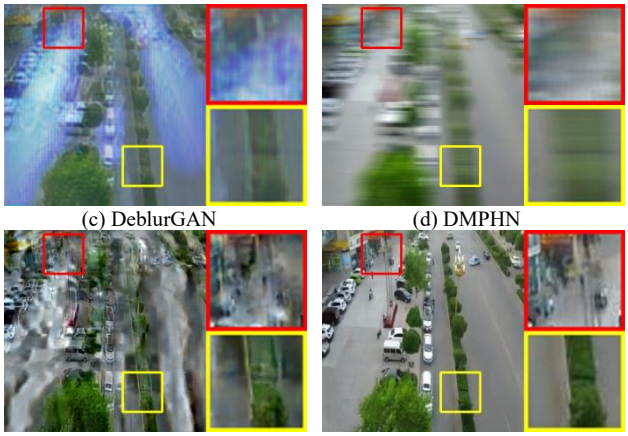
The experimental results indicate that MRFNet can achieve considerable precision. Furthermore, MRFNet runs much faster than the other deblurring models, such as SIUN and DMPHN. Compared with DeblurGAN, the proposed MRFNet model performs well both in terms of speed and deblurring quality. Owing to the added lightweight process, the GPU memory's occupation remains at a low level. Our method can also recover more details and achieve relatively high SSIM and PSNR values. Figs. 6 and 7 show that results of the other models whose results contain artifacts and color distortions, whereas the MRFNet performs image deblurring in a stable manner. For instance, the handwriting and flags in Fig.6 and the details in the streets in Fig. 7 are processed to better state by MRFNet.



(a) Input image



(b) Ground truth



(c) DeblurGAN (d) DMPHN

(e) SIUN (f) Ours

Fig. 7: Results of comparative experiments. Our restored images show vivid colors and sharp details.

V. Conclusion and Future Work

In this paper, we propose an efficient and accurate framework called the MRFNet to .... The proposed network exploits the MRF, lightweight process, remote residual connection, and scale refinement loss function to enable the model to handle motion and Gaussian blur scenarios while preserving fast inference speed.

In the future, we will develop fast deblurring inference of MRFNet on edge devices. As computational capability will likely be much lower than that of GPUs used in our experiments, the techniques of model compression, including pruning, quantization, and so on, will also be explored. We will also adapt this model to video deblurring or the deblurring of inpainting results at the post-processing stage.