Quad-Pixel Image Defocus Deblurring: A New Benchmark and Model Supplementary Material

Hang Chen[1] Yin Xie[1] Xiaoxiu Peng[1] Lihu Sun[1] Wenkai Su[1] Xiaodong Yang[2] Chengming Liu[2]
[1]OMNIVISION, Wuhan, China [2]OMNIVISION, Santa Clara, CA

In this supplementary material, we provide more details about our method:

- Section 1 QPDD Dataset Details.
- Section 2 Defocus Blur and Disparity of QP Data.
- Section 3 Analysis of Model Complexity.
- Section 4 LAMB Details.
- Section 5 Implementation Details.
- Section 6 More Visual Comparison Results.

Our QPDD dataset and code will be released soon.

1. QPDD Dataset Details

We utilize a 50-megapixel QPD image sensor with a 1.0-micron pixel size and 1/1.5" optical format to capture images. Our QPDD dataset includes 200 indoor scenes, comprising 4,935 pairs of images. Among these pairs, 2,689 pairs have a resolution of 2160 $\times$ 1280 pixels, 2,093 pairs have a resolution of 1280 $\times$ 1792 pixels, and 153 pairs have a resolution of 1080 $\times$ 1280 pixels. Some samples from this dataset are shown in Fig. 1. Additionally, we offer 100 extra multi-depth test images, each with a resolution of 4096 $\times$ 3072 pixels, comprising 50 outdoor, 30 indoor, and 20 laboratory scenes. Examples are presented in Fig. 2. Table 1 shows the overall comparisons between DPDD [1] and our QPDD dataset. Our QPDD dataset provides more defocus and all-in-focus image pairs with different resolutions. Furthermore, we provide 100 extra multi-depth images to verify generalization performance.

Figure 1. Some scenes of the QPDD dataset.

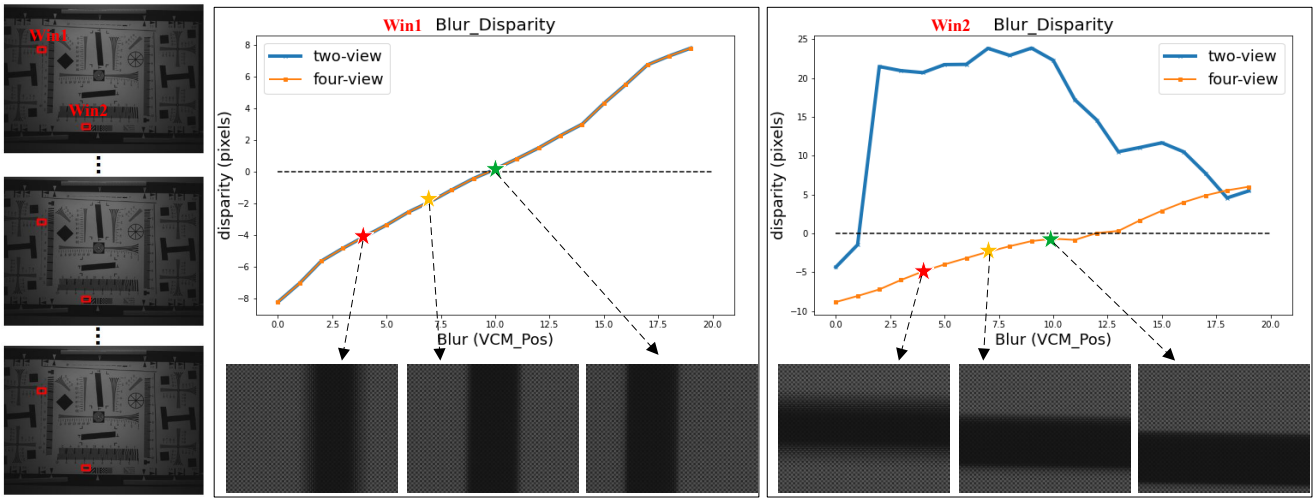
Figure 2. Some scenes from the additional 100 multi-depth test dataset of QPDD.

Dataset	Total Nums	Distributions		Training	Testing	Validation
		Nums	Resolutions			
DPDD[1]	500	500	1680×1120	350	76	74
QPDD	4935	2689 2093 153	2160 × 1280 1280 × 1792 1080 × 1280	3735	600 + 100	600

Table 1. Comparisons between DPDD and QPDD datasets.

2. Defocus Blur and Disparity of QP Data

To demonstrate that the disparity provided by the four-view (L, R, T, B) inputs of QP data is more robust than that provided by the two-view (L, R) inputs, we conduct the following experiments. First, we adjust the parameters of the VCMs with a fixed step size to obtain a sequence of images with gradually varying blur levels. We then calculate the disparity of the selected windows for different views in each frame using the normalization cross-correlation (NCC) [6] matching algorithm. As shown in Fig. 3 (b), both two-view (L, R) inputs and four-view (L, R, T, B) inputs exhibit a good linear relationship between disparity and defocus blur in vertical textures. However, in horizontal textures (see Fig. 3 (c)), the disparity results from two-view (L, R) inputs show significant fluctuations, while the four-view (L, R, T, B) inputs still maintain a good linear relationship between disparity and defocus blur. Disparity is correlated to the amount of defocus blur. Four-view inputs provide more robust disparity estimation compared to two-view inputs, enabling better guidance for defocus deblurring.



(a) (b) (c)

Figure 3. Analysis on defocus blur and disparity. (a) Adjust the VCMs position in fixed steps to capture a sequence of images with gradually varying blur levels, and select two regions, win1 and win2, for disparity calculation. (b) Disparity results of win1. (c) Disparity results of win2. The x-axis represents the VCM positions corresponding to the images, which have a linear relationship with defocus blur. The y-axis represents the disparity of the images. The 'two-view' indicates that the disparity is calculated using the L and R views. The 'four-view' indicates that the disparity is calculated using the L, R, T and B views.

3. Analysis of Model Complexity

Furthermore, we provide a comprehensive complexity analysis of our method (see Tab. 2), presenting the number of parameters, and measuring the computational cost using FLOPs (with image size at 1120 $\times$ 1680). We also provide the number of parameters and computational costs of other methods, including DPDNet [1], RDPD [2], IFAN [8], DRBNet [12], BaMBNet [9], Restormer [16], LaKDNet [13], and K3DN [14], for reference. The PSNR values obtained from testing on the DPDD dataset are also provided as a reference. As shown in Tab. 2, compared to CNNs-based methods (RDPD [2], IFAN [8], DRBNet [12], and BaMBNet [9]), our computational cost has significantly increased, but the performance improvement is also substantial. However, compared to the Transformer-based method Restormer [16], the large kernel-based method LaKDNet [13], and the disparity-based method K3DN [14], our computational cost is relatively lower, and our performance is better.

For DP data, the L and R views are input into the two-branch encoder of LMNet with shared parameters separately. For QP data, the L and R views are concatenated, as well as the T and B views. Each concatenated pair is then fed separately into the two-branch encoder without shared parameters. Due to the different weight training methods used in the two-branch encoder, the QP-based LMNet has 5.68M more parameters compared to the DP-based LMNet, but both models have almost the same computational cost.

Method	PSNR$_\uparrow$	Params (M)$_\downarrow$	FLOPs (G)$_\downarrow$
DPDNet$_D$ [1]	25.13	31.03	3150
RDPD$_D$ [2]	25.39	24.28	901
IFAN$_D$ [8]	25.99	10.48	**794**
DRBNet$_D$ [12]	26.33	11.69	1273
BaMBNet$_D$ [9]	26.40	**4.50**	1804
Restormer$_D$ [16]	26.66	26.13	4458
LaKDNet$_D$ [13]	26.89	38.42	5168
K3DN$_D$ [14]	27.06	15.90	3352
Ours$_D$	**27.25**	18.18	3227
Ours$_Q$	-	23.86	3231

Table 2. Comparisons of model complexity. The suffix D indicates two-view (L, R) inputs, and the suffix Q indicates four-view (L, R, T, B) inputs.

4. Local-gate assisted Mamba Block Details

Inspired by the continuous linear time-invariant (LTI) systems, the state space models (SSMs), such as structured state space sequence model (S4) [4] and Mamba [3], map a 1-dimensional function or sequence $x(t) \in \mathbb{R} \rightarrow y(t) \in \mathbb{R}$ through a hidden state $h(t) \in \mathbb{R}^N$. These models can be formulated as linear ordinary differential equations (ODEs) as follows:

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}h(t). \end{aligned} \tag{1}$$

where N is the state size, $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the evolution parameter, $\mathbf{B} \in \mathbb{R}^{N\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times N}$ are the projection pa-

rameters.

To integrate SSMs into deep learning algorithms, the continuous differential equations are converted to discrete version. Denote Δ as the timescale parameter, the continuous parameters $\mathbf{A}$ and $\mathbf{B}$ are transformed into discrete parameters $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ using the zero-order hold (ZOH) rule, which are defined as follows:

$$\begin{aligned} \overline{\mathbf{A}} &= exp(\Delta\mathbf{A}), \\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}. \end{aligned} \tag{2}$$

After the discretization, the discretized version of Eq. (2) using a step size Δ can be rewritten as:

$$\begin{aligned} h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{3}$$

Furthermore, the iterative process delineated in Eq. (3) can be expedited through parallel computation, employing a global convolution operation:

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^{k-1}\overline{\mathbf{B}}), \\ \mathbf{y} &= \mathbf{x} \circledast \overline{\mathbf{K}}. \end{aligned} \tag{4}$$

where k is the length of the input sequence $\mathbf{x}$, $\circledast$ denotes convolution operation, and $\overline{\mathbf{K}} \in \mathbb{R}^k$ is a structured convolutional kernel.

Despite the efficiency brought by discretization, parameters (Δ, $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, $\mathbf{C}$) in SSMs are data-independent and time-invariant, limiting the expressiveness of the hidden state to compress seen context. Recently, Mamba [3] introduces a selection mechanism into SSM, making the parameters data-dependent, thus allowing for a dynamic feature representation. In addition, the parallel scanning allows Mamba to facilitate efficient training. The architecture of Mamba is shown in Fig. 4.

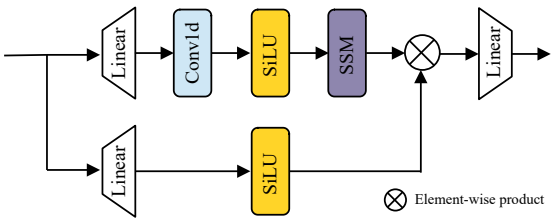


Figure 4. The structure of Mamba block. Based on traditional SSM, Mamba adds a SiLU activation similar to the Gated MLP [10], which allows the model to fuse and select information across tokens. At the same time, the Linear and Conv layers allow the model to learn data-dependent parameters.

The Mamba block achieves linear scalability in sequence length and delivers competitive performance in language modeling tasks. To apply Mamba for visual tasks, some methods [5, 7, 11] adopt a 2D Selective Scan strategies with four or more directions to flatten 2D images into 1D sequences. The flattening strategy causes some spatially close pixels in the 2D feature map to become widely separated in the 1D token sequence, leading to local pixel forgetting. Furthermore, Mamba introduces a larger number of hidden states to capture long-range dependencies, which can lead to channel redundancy. The motivation behind our Local-gate assisted Mamba Block (LAMB) design is to explore Mamba to capture long-range dependencies with linear complexity, and to design a Local Gate Module (LGM) that reduces local pixel forgetting and channel redundancy within Mamba. By integrating the features from a bidirectional SSM [17] and LGM using learnable weights, the LAMB enables a more enhanced capture of global and local dependencies.

5. Implementation Details

Our LMNet is implemented in PyTorch framework, and all training and experimental procedures are conducted on two NVIDIA RTX 3090 GPUs. The configurations of our model are in Tab. 3.

Stage Level	LAMB Nums	Channel Nums	Spatial Dimension
1	$N_1 = 4$	48	$H \times W$
2	$N_2 = 6$	96	$H/2 \times W/2$
3	$N_3 = 6$	192	$H/4 \times W/4$
Enhancement	$N_4 = 8$	192	$H/4 \times W/4$

Table 3. Configurations of LMNet. Each level of encoder and decoder is composed of N LAMBs, where $N \in \{N_1, N_2, N_3\}$. At the third level, following the third SFM, there is a feature enhancement module consisting of N_4 LAMBs. The number of feature channels in the first level is $M = 48$. H and W are the image height and width.

6. More Visual Comparison Results

We provide more visual comparisons on DPDD dataset [1] (see Fig. 5 and Fig. 6) and our proposed QPDD dataset (see Fig. 7, Fig. 8, and Fig. 9). All the presented results are based on experimental methods that are consistent with the descriptions in the main text of the paper. For the DPDD dataset, we compare our LMNet with several recent state-of-the-art dual-pixel (DP) based defocus deblurring methods, including RDPD [2], IFAN [8], MIRNetV2 [15], Restormer [16], and LaKDNet [13]. Note that we use their publicly available checkpoints to generate the all-in-focus restorations. From Fig. 5 and Fig. 6, we can see that our method is effective for removing spatially varying defocus blur. For the QPDD dataset, two representative classic defocus deblurring methods, Restormer [16] and LaKDNet [13], are used as references. To ensure a fair comparison, we retrain these two methods using the two-view (L, R) sRGB

images in the QPDD dataset following their own setup. For our LMNet, we provide two sets of results: one trained with two-view (L, R) inputs and the other trained with four-view (L, R, T, B) inputs. As shown in Fig. 7, Fig. 8, and Fig. 9, our method with four-view inputs recovers more details of the textures and edges compared to other methods.

References

[1] Abdullah Abuolaim and Michael S. Brown. Defocus deblurring using dual-pixel data. In *European Conference on Computer Vision*, pages 111–126. Springer, 2020. 1, 2, 3

[2] Abdullah Abuolaim, Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Learning to reduce defocus blur by realistically modeling dual-pixel data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2289–2298, 2021. 2, 3, 5, 6

[3] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 2, 3

[4] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 2

[5] Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. Mambair: A simple baseline for image restoration with state-space model. *arXiv preprint arXiv:2402.15648*, 2024. 3

[6] Heiko Hirschmuller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE transactions on pattern analysis and machine intelligence*, 31(9):1582–1599, 2008. 1

[7] Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. *arXiv preprint arXiv:2403.09338*, 2024. 3

[8] Junyong Lee, Hyeongseok Son, Jaesung Rim, Sunghyun Cho, and Seungyong Lee. Iterative filter adaptive network for single image defocus deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2034–2042. IEEE, 2021. 2, 3, 5, 6

[9] Pengwei Liang, Junjun Jiang, Xianming Liu, and Jiayi Ma. Bambnet: A blur-aware multi-branch network for dual-pixel defocus deblurring. *IEEE/CAA Journal of Automatica Sinica*, 9(5):878–892, 2022. 2

[10] Hanxiao Liu, David R. So Zihang Dai, and Quoc V. Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021. 3

[11] Yang Liu, Jiahua Xiao, Yu Guo, Peilin Jiang, Haiwei Yang, and Fei Wang. Hsidmamba: Exploring bidirectional state-space models for hyperspectral denoising. *arXiv preprint arXiv:2404.09697*, 2024. 3

[12] Lingyan Ruan, Bin Chen, Jizhou Li, and Miu-Ling Lam. Learning to deblur using light field generated and real defocus images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16283–16292. IEEE, 2022. 2

[13] Lingyan Ruan, Mojtaba Bemana, Hans-peter Seidel, Karol Myszkowski, and Bin Chen. Revisiting image deblurring with an efficient convnet. *arXiv preprint arXiv:2302.02234*, 2023. 2, 3, 5, 6, 7, 8, 9

[14] Y. Yang, L. Pan, L. Liu, and M. Liu. K3dn: Disparity-aware kernel estimation for dual-pixel defocus deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13263–13272, 2023. 2

[15] S. Zamir, A. Arora, S. Khan, M. Hayat, F. Khan, M. Yang, and L. Shao. Learning enriched features for fast image restoration and enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(02):1934–1948, 2022. 3, 5, 6

[16] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5728–5739, 2022. 2, 3, 5, 6, 7, 8, 9

[17] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024. 3

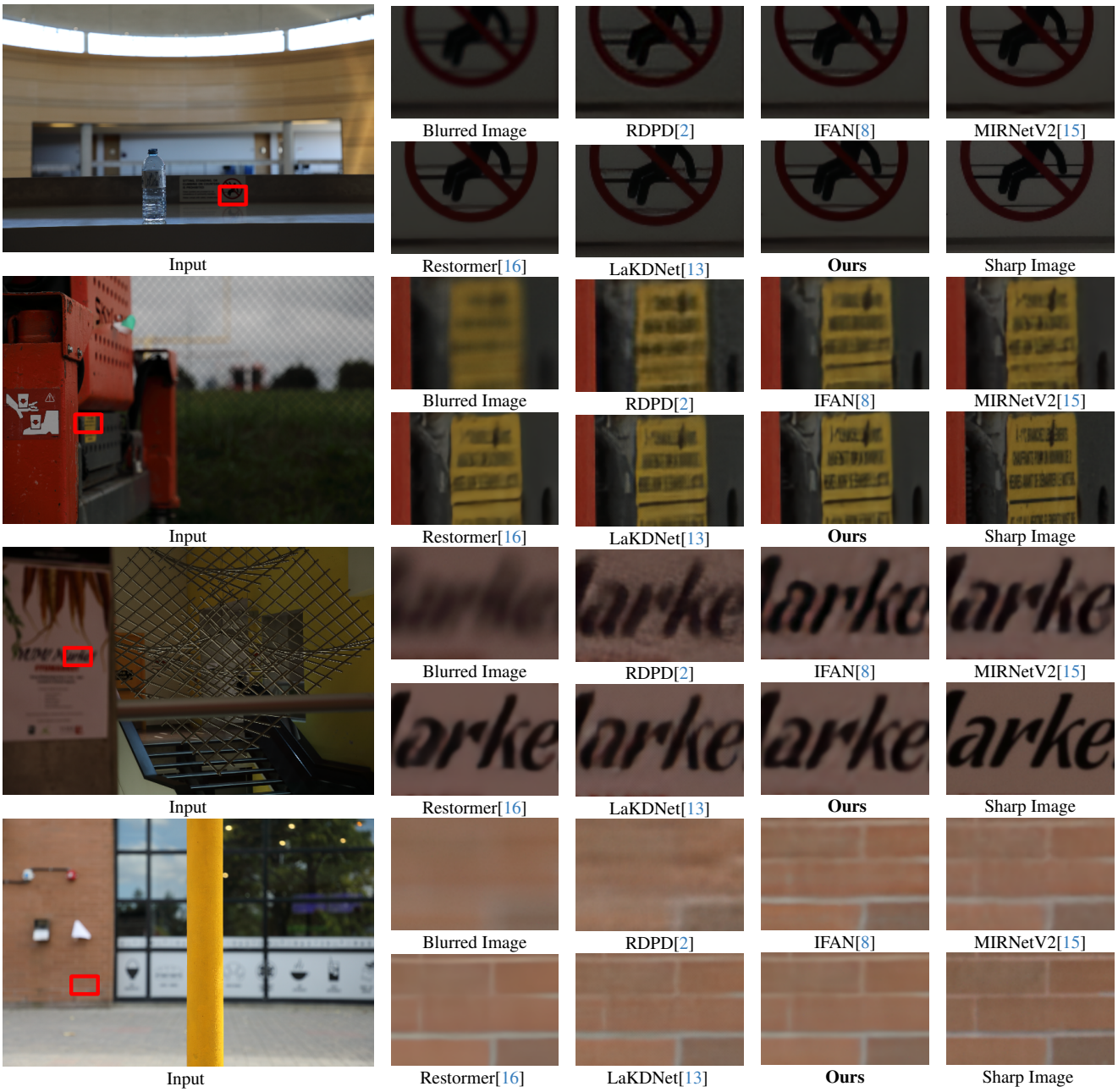


Figure 5. Comparisons of defocus deblurring performance on the DPDD dataset. All the networks take DP images as the inputs. From left to right: Blurred Image, RDPD [2], IFAN [8], MIRNetV2 [15], Restormer [16], LaKDNet [13], **Ours** (LMNet), and Sharp Image.

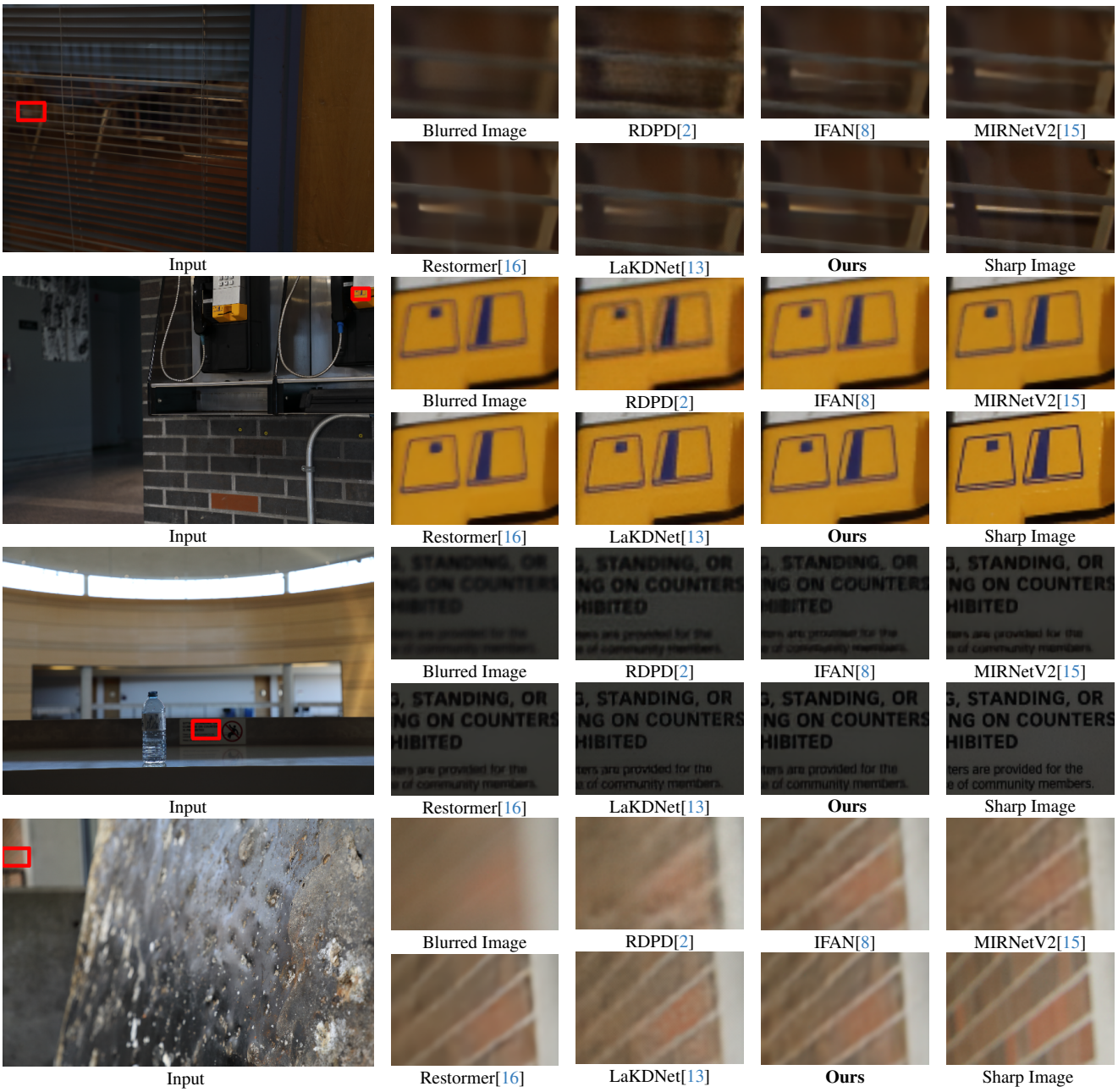


Figure 6. Comparisons of defocus deblurring performance on the DPDD dataset. All the networks take DP images as the inputs. From left to right: Blurred Image, RDPD [2], IFAN [8], MIRNetV2 [15], Restormer [16], LaKDNet [13], **Ours** (LMNet), and Sharp Image.

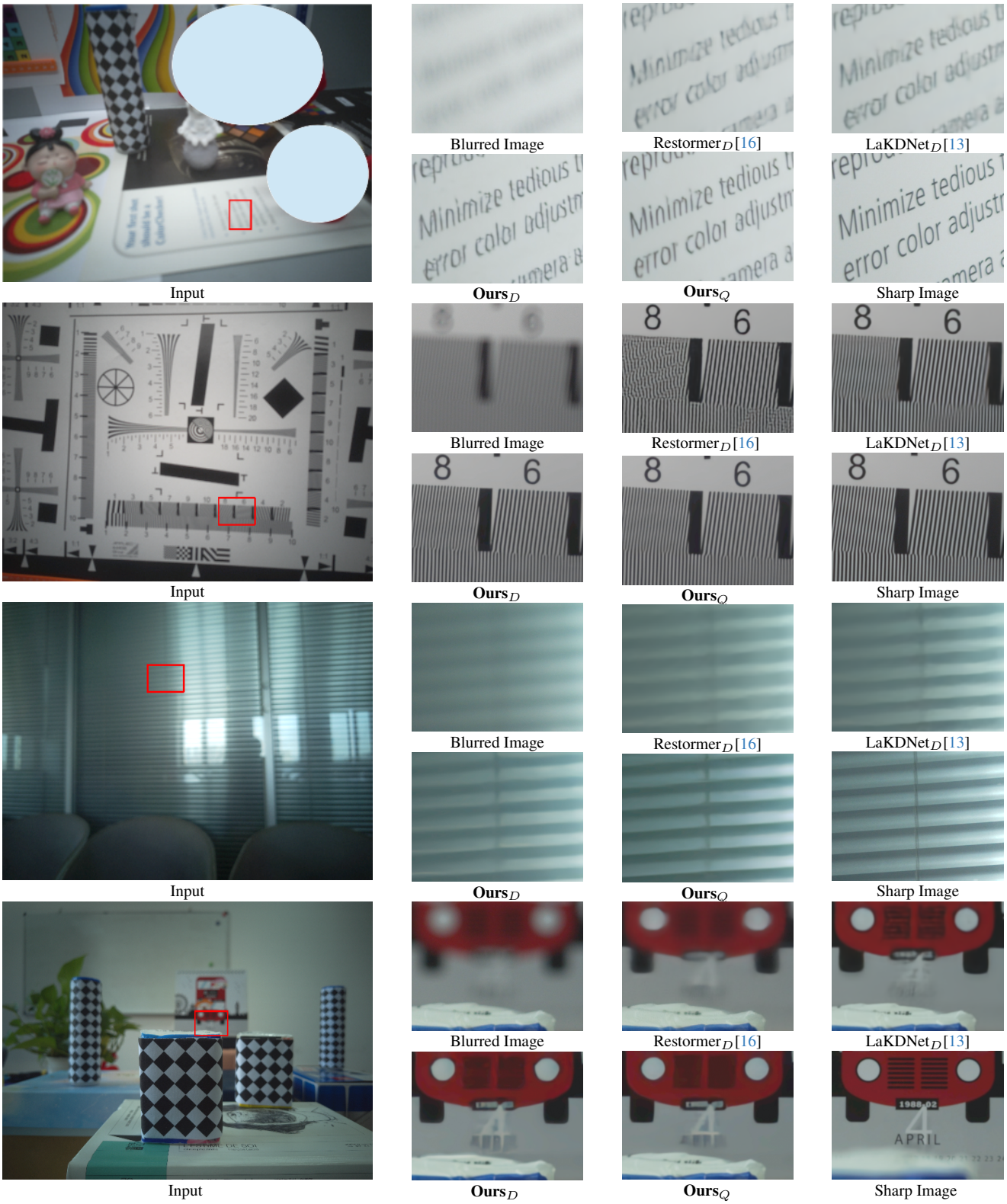


Figure 7. Comparisons of defocus deblurring performance on the QPDD dataset. The suffix D indicates two-view (L, R) inputs, and the suffix Q indicates four-view (L, R, T, B) inputs. From left to right: Blurred Image, Restormer$_D$ [16], LaKDNet$_D$ [13], **Ours**$_D$, **Ours**$_Q$, and Sharp Image.

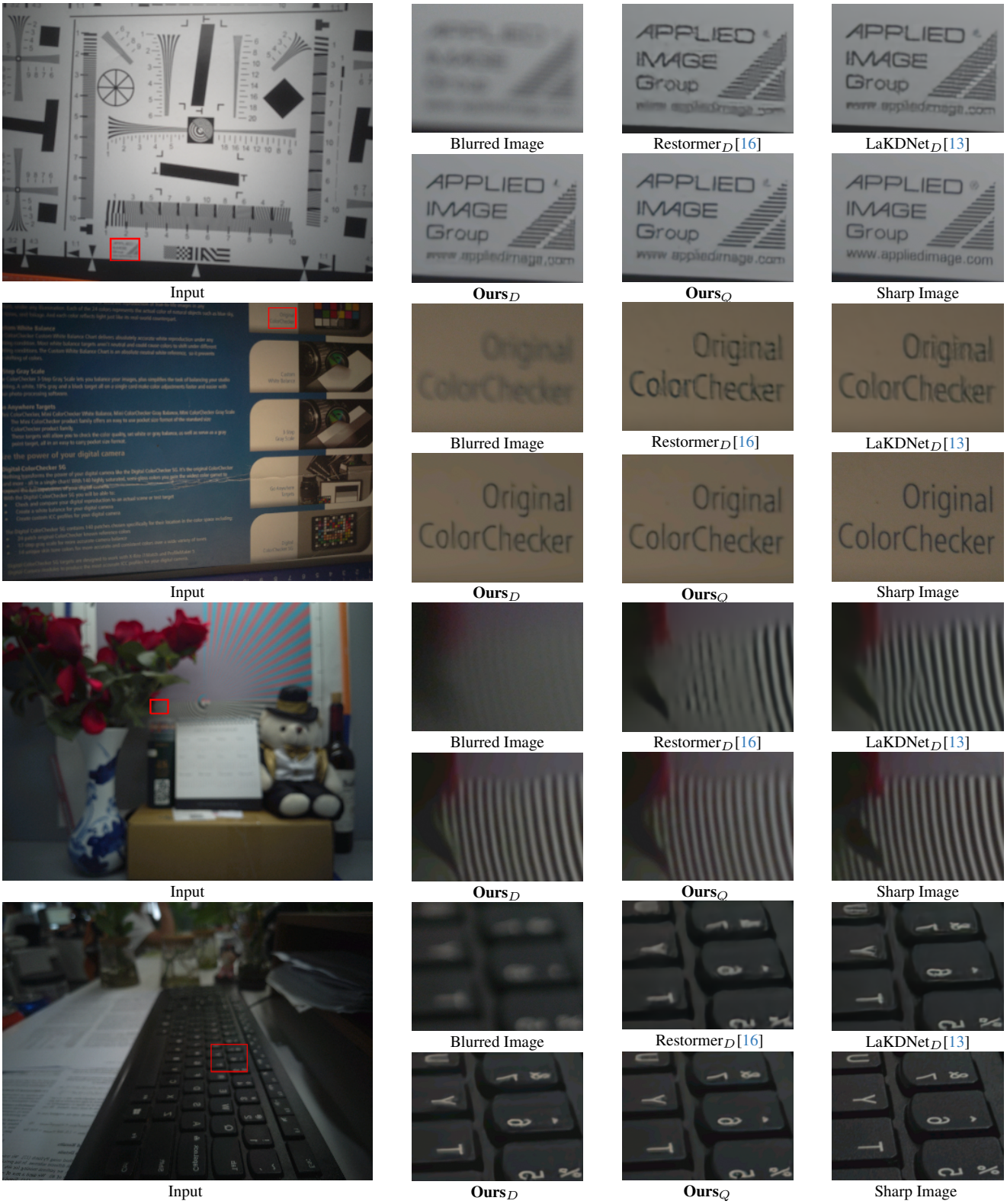


Figure 8. Comparisons of defocus deblurring performance on the QPDD dataset. The suffix D indicates two-view (L, R) inputs, and the suffix Q indicates four-view (L, R, T, B) inputs. From left to right: Blurred Image, Restormer$_D$ [16], LaKDNet$_D$ [13], **Ours**$_D$, **Ours**$_Q$, and Sharp Image.

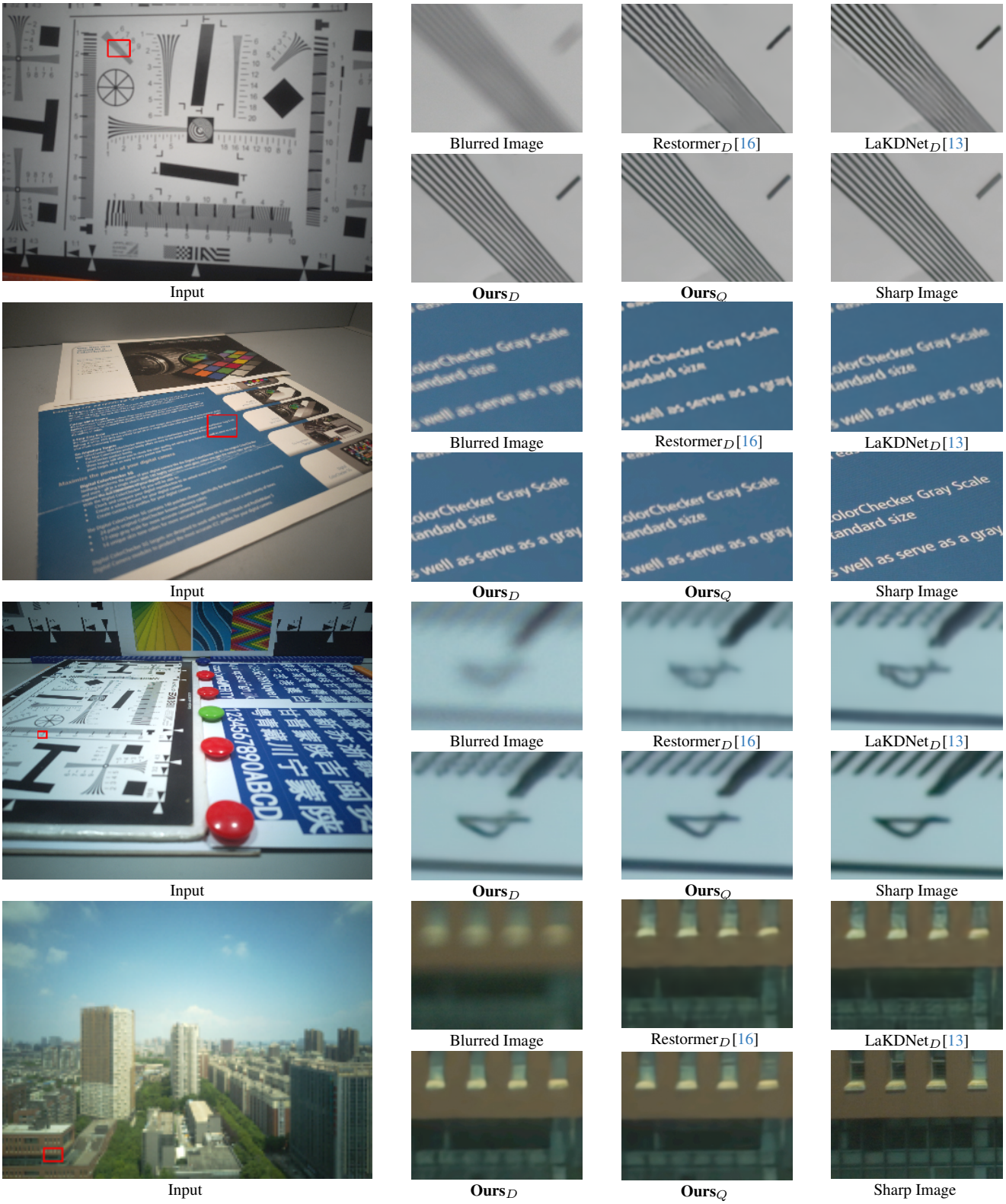


Figure 9. Comparisons of defocus deblurring performance on the QPDD dataset. The suffix D indicates two-view (L, R) inputs, and the suffix Q indicates four-view (L, R, T, B) inputs. From left to right: Blurred Image, Restormer$_D$ [16], LaKDNet$_D$ [13], **Ours**$_D$, **Ours**$_Q$, and Sharp Image.