In this supplementary material, we provide additional implementation details (Appendix A), additional experiment results (Appendix B), additional ablation study (Appendix C), and limitations (Appendix D) for our K3DN framework.

A. Additional Implementation Details

K3DN uses a 3-level U-net architecture. We use AdamW optimizer [13] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate $= 3 \times 10^{-4}$, and weight decay $= 10^{-6}$. We use the ‘cosine annealing with warmups’ learning rate scheduler, and set the ‘cycle steps’, ‘warmup steps’, and ‘minimum learning rate’ to 200, 100, and $6 \times 10^{-5}$. For the DPD-blur dataset [1], our model is trained for 20k iterations in a two-stage manner. First, we train our model without the SRP blocks from scratch for 9.8K iterations. Second, we freeze all model weights and train the newly added parameters from the SRP blocks for another 10.2K iterations, while excluding the reblurring loss $\mathcal{L}_{reb}$ from the overall training loss $\mathcal{L}$ as our target is to preserve the sharp regions of defocus blurred DP pair. For the DDD-syn dataset [15] and RDPD dataset [2], we adopt resource-constrained training, as the synthetic datasets are easy to be overfitted. Specifically, our model is respectively trained for 4k and 40k iterations on the two datasets. When the performance of other methods is not available, we train them with the same iterations for a fair comparison.

Our $\mathcal{L}_{deb}$ uses a combination of Multi-Scale Charbonnier loss $\mathcal{L}_{chb}$ [25], Multi-Scale Edge loss $\mathcal{L}_{edg}$ [25] and Multi-Scale Frequency loss $\mathcal{L}_{frq}$ [14], i.e., $\mathcal{L}_{deb} = \mathcal{L}_{chb} + \lambda_2 \mathcal{L}_{edg} + \lambda_3 \mathcal{L}_{frq}$. Meanwhile, we define $\mathcal{L}_{reb}$ as a mean squared error-based loss. We set $\lambda_1 = 1 \times 10^{-1}$, and $\lambda_2 = 5 \times 10^{-2}$, $\lambda_3 = 1 \times 10^{-2}$. During optimization, we apply gradient norm clipping at $1 \times 10^{-2}$.

The detailed architecture of our K3DN framework is summarised in Tab. 13. All convolution layers apply a LeakyReLU with a negative slope of 0.2. We use Num as a column attribute to represent the number of replication for current layers. We denote bn and bc as the base number of replication and base channel. The configurations of our model variants (i.e., Tiny, Lightweight, and Large) are in Tab. 6.

<table>
<thead>
<tr>
<th>Variants</th>
<th>bn</th>
<th>bc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>Lightweight</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>Large</td>
<td>4</td>
<td>48</td>
</tr>
</tbody>
</table>

B. Additional Experiment Results

We briefly investigate the reblur capability of our model in Fig. 9. Next, we verify our model generalization ability in Fig. 11. We then study our disparity estimator that is trained in an unsupervised manner, in Fig. 12. In the following, we visualize the sub-kernel with the largest weight assigned by the disparity vector for different image regions in our PSF block (Fig. 13). Note that we linearly transform the image space to feature space and train a PSF block for better kernel visualization. Finally, we present more comparisons with state-of-the-art methods (Fig. 14, Fig. 15, Fig. 16, Fig. 17 and Fig. 18), in addition to the Fig. 6 and Fig. 7 from our main paper. Specifically, we compare with RDPD [2], KPAC [19], IFAN [10], Deep-RFT [14], DDDNet [15], RDPD [2], BAMBNet [11], and Restormer [24]. Note that we use their publicly available checkpoint to generate the all-in-focus restorations.
Table 7. Performance evaluation on Google Pixels DP image dataset from [23]. The performance of our tiny model is presented.

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>RMSE_{rel}(10^{-2})↓</th>
<th>MAE_{rel}(10^{-2})↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiener Deconv [27]</td>
<td>25.81</td>
<td>0.704</td>
<td>5.13</td>
<td>0.320</td>
</tr>
<tr>
<td>DPDNet [1]</td>
<td>25.59</td>
<td>0.777</td>
<td>5.25</td>
<td>0.340</td>
</tr>
<tr>
<td>Xin et al. [23]</td>
<td>26.69</td>
<td>0.804</td>
<td>4.93</td>
<td>0.270</td>
</tr>
<tr>
<td>IFAN [10]</td>
<td>31.49</td>
<td>0.867</td>
<td>2.66</td>
<td>0.164</td>
</tr>
<tr>
<td>Restormer [24]</td>
<td>31.27</td>
<td>0.859</td>
<td>2.73</td>
<td>0.161</td>
</tr>
<tr>
<td>Ours (Tiny)</td>
<td>25.85</td>
<td>0.794</td>
<td>5.10</td>
<td>0.380</td>
</tr>
<tr>
<td>Ours (Lightweight)</td>
<td>25.95</td>
<td>0.799</td>
<td>5.04</td>
<td>0.377</td>
</tr>
<tr>
<td>Ours (Large)</td>
<td>26.11</td>
<td>0.805</td>
<td>4.95</td>
<td>0.372</td>
</tr>
</tbody>
</table>

Table 8. Single image defocus deblurring of our method on the DPD-blur dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>RMSE_{rel}(10^{-2})↓</th>
<th>MAE_{rel}(10^{-2})↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (Tiny)</td>
<td>25.85</td>
<td>0.794</td>
<td>5.10</td>
<td>0.380</td>
</tr>
<tr>
<td>Ours (Lightweight)</td>
<td>25.95</td>
<td>0.799</td>
<td>5.04</td>
<td>0.377</td>
</tr>
<tr>
<td>Ours (Large)</td>
<td>26.11</td>
<td>0.805</td>
<td>4.95</td>
<td>0.372</td>
</tr>
</tbody>
</table>

Figure 10. Comparison of image restoration performance on the Google Pixels dataset [23].

Figure 11. The generalization ability of disparity-based methods (an expansion of Fig. 7). Here, we mainly consider the disparity-based approaches, i.e., IFAN [10] and Xin et al. [23] (refer to Fig. 7 for restoration results of other methods). Note that all methods are not trained and specialized for the DPD-disp dataset [16], i.e., our model and IFAN use the pretrained checkpoint on the DPD-blur dataset [1], and Xin et al. uses the provided and precalibrated kernels. We present two kinds (Gray and sRGB) of restored images for Xin et al. [23], where the sRGB restored images are generated by deblurring on each channel independently.

We study the impact (Tab. 9) of $\frac{H_k}{W_k} \times \frac{W_l}{W_d}$ in the DPD-blur dataset [1].

Table 9. Alignment of encoder and disparity estimator.

<table>
<thead>
<tr>
<th>$\frac{H_k}{W_k} \times \frac{W_l}{W_d}$</th>
<th>9 x 9</th>
<th>14 x 18</th>
<th>18 x 14</th>
<th>18 x 18</th>
<th>27 x 27</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR↑</td>
<td>26.76</td>
<td>26.77</td>
<td>26.84</td>
<td>26.72</td>
<td>26.60</td>
</tr>
</tbody>
</table>

With $\frac{H_k}{W_k} = 18$ and $\frac{W_l}{W_d} = 14$, we find the best performance. This is potentially determined by the complexity of the blur model in the dataset. During testing, to be compatible with diverse sizes of model inputs, we resize the inputs to the multiples of the spatial size, and then we rescale the model outputs to the original size.

Spatial size of the kernel set. We analyze the spatial size of the kernel set (i.e., $H_k \times W_k$) of the candidate kernel set $\mathcal{K}$ in Tab. 10.

Table 10. Impact of the spatial size of the kernel set.

<table>
<thead>
<tr>
<th>$H_k \times W_k$</th>
<th>3 x 3</th>
<th>5 x 5</th>
<th>7 x 7</th>
<th>9 x 9</th>
<th>11 x 13</th>
</tr>
</thead>
</table>

Considering the model performance, we set $H_k = 9$ and $W_k = 9$.

Number of the PSF blocks. By fixing all other components of a lightweight K3DN framework, we study the optimal number of PSF blocks in Tab. 11. Note that the

Tab. 7 and Fig. 10. Note that the brightness and contrast of restorations are adjusted for better visualization. This dataset is captured by Google Pixels smartphone, and provides 17 pairs of defocus blurred DP images and associated all-in-focus images. It covers both indoor and outdoor scenes. We test K3DN framework by using the pretrained checkpoint on the DPD-blur dataset. Similarly, we present the performance of Restormer [24] and IFAN [10], the latest state-of-the-art method, in this dataset.

Moreover, we adapt our K3DN framework to perform the single image defocus deblurring task (i.e., use the center view of the DP image) on the DPD-blur dataset. The performance is presented in Tab. 8.

C. Additional Ablation Study

All ablation studies are conducted with our lightweight model.

The alignment of encoder and disparity estimator. As discussed in Sec. 3, $F_B$ and $R$ are spatially aligned with each other, while each $i$-th layer features of $F_B$ can be founded by performing a nearest neighbor interpolation. In other words, each vector $r^i \in R$ is spatially aligned with $F^i_B \in F_B$. By varying the downsampling rate (e.g., the stride of convolution) and resizing the inputs for our disparity estimator, for each $r^i$, the spatial size (i.e., $\frac{H_k}{W_k} \times \frac{W_l}{W_d}$) of the aligned feature $F^i_B$ is changed accordingly. Here,

we study the impact (Tab. 9) of $\frac{H_k}{W_k} \times \frac{W_l}{W_d}$ in the DPD-blur dataset [1].

Table 10. Impact of the spatial size of the kernel set.

<table>
<thead>
<tr>
<th>$H_k \times W_k$</th>
<th>3 x 3</th>
<th>5 x 5</th>
<th>7 x 7</th>
<th>9 x 9</th>
<th>11 x 13</th>
</tr>
</thead>
</table>

Considering the model performance, we set $H_k = 9$ and $W_k = 9$.

Number of the PSF blocks. By fixing all other components of a lightweight K3DN framework, we study the optimal number of PSF blocks in Tab. 11. Note that the

...
Figure 12. (a)-(b) Examples of input left view DP images and their associated disparity feature clusters. With obtained features from our disparity estimator, we perform a k-means algorithm to cluster similar disparity features across the image. The assigned cluster-IDs are used to colorize the latent features processed by the PSF block.

Figure 13. Sample kernels from the PSF block.

lightweight K3DN has 4 PSF blocks (i.e., $2 \times bn$ in Appendix A and Tab. 13).

Table 11. Number of PSF blocks.

<table>
<thead>
<tr>
<th>#PSF blocks</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

Conceptually, the more PSF blocks, the more complex blur models that we can handle. However, with the lightweight model size, there is limited feature semantics that can be embedded in the feature space due to the small model size. Therefore, a large number of PSF blocks can potentially harm the model generalization ability, and we find the optimal number of PSF blocks is 4.

Inference speed. We investigate the inference speed of K3DN and other state-of-the-art methods in Tab. 12. The experiments are conducted under a single NVIDIA A40 GPU. We use batch size 1, warm up the GPU for 5 iterations, and average 30 random testing results. In comparison to the latest state-of-the-art method, Restormer [24], our method has significant inference speed improvements without any performance deterioration (refer to Tab. 1, Tab. 2 and Tab. 3 for the performance comparison).

Table 12. Inference speed of past methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Restormer</th>
<th>BAMBNet</th>
<th>DeepRFT</th>
<th>DRBNNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second</td>
<td>2.38</td>
<td>0.970</td>
<td>1.03</td>
<td>0.197</td>
</tr>
<tr>
<td>Method</td>
<td>IFAN</td>
<td>Ours (Tiny)</td>
<td>Ours (Lightweight)</td>
<td>Ours (Large)</td>
</tr>
<tr>
<td>Second</td>
<td>0.142</td>
<td>0.236</td>
<td>0.318</td>
<td>0.578</td>
</tr>
</tbody>
</table>

D. Limitations

Though our PSF blocks follow the blur mode of the DP image formulation (Sec. 3.1) and our K3DN framework achieves a favorable deblur performance, the exact inversion for the model is not maintained. For example, in the deblurring and reblurring processes, our encoder and decoder do not have an exact inverse constraint (i.e., they are trained to perform encoding and decoding), and only the inversion within each PSF block is maintained. In our future work, we plan to study fully invertible network architectures for K3DN.
Table 13. K3DN architecture. We use \( \downarrow \) and \( \uparrow \) to denote downsampling and upsampling, respectively. For the PSF block, a point-wise convolution \([20]\) and a residual connection are also added to improve the feature representation ability, where the kernel sizes are specified accordingly. Note that a point-wise convolution is easy to invert by using the LU decomposition \([8]\).

<table>
<thead>
<tr>
<th>Disparity Estimator</th>
<th>Type</th>
<th>Input</th>
<th>Activation</th>
<th>Kernel</th>
<th>Channel</th>
<th>Stride</th>
<th>Padding</th>
<th>Dilation</th>
<th>Num</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoder</td>
<td>conv</td>
<td>( b_1 )</td>
<td>ReLU</td>
<td>3</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_1 )</td>
</tr>
<tr>
<td>conv</td>
<td>( b_2 )</td>
<td>ReLU</td>
<td>3</td>
<td>64</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_2 )</td>
<td></td>
</tr>
<tr>
<td>conv</td>
<td>( b_3 )</td>
<td>ReLU</td>
<td>3</td>
<td>128</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>( b_3 )</td>
<td></td>
</tr>
<tr>
<td>AvgPool</td>
<td>( b_4 )</td>
<td>-</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_4 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AvgPool</td>
<td>( b_5 )</td>
<td>-</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_5 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv</td>
<td>( b_6 )</td>
<td>ReLU</td>
<td>3</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_6 )</td>
<td></td>
</tr>
<tr>
<td>conv</td>
<td>( b_7 )</td>
<td>ReLU</td>
<td>3</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_7 )</td>
<td></td>
</tr>
<tr>
<td>conv</td>
<td>( b_9 )</td>
<td>ReLU</td>
<td>3</td>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>( b_9 )</td>
<td></td>
</tr>
</tbody>
</table>

\( \downarrow \) Shared Feature Extractor.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>PSF {( F_B, R )}</th>
<th>( \downarrow ) Deblurring Framework. Shared Encoder, PSF Blocks, and Decoder.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \downarrow )</td>
<td>dconv ( \hat{F}_B )</td>
<td>LeakyReLU ( {b_4, {b_6:16}, {b_7:32}} ) ReLU ( {9,1} ) ( 4 \times bc ) 1 ( {5,1} ) 1 ( 2 \times bn ) ( \hat{F}_B )</td>
</tr>
<tr>
<td>( \downarrow )</td>
<td>res ( d_1 )</td>
<td>( {b_4, {b_6:16}, {b_7:32}} ) ( {b_4, {b_6:16}, {b_7:32}} ) ( {9,1} ) ( 4 \times bc ) 1 ( {5,1} ) 1 ( 2 \times bn ) ( \hat{F}_B )</td>
</tr>
</tbody>
</table>

Decoder

<table>
<thead>
<tr>
<th>Encoder</th>
<th>PSF {( F_1, R )}</th>
<th>( \downarrow ) Reblurring Framework. Shared Encoder, PSF Blocks, and Decoder.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \downarrow )</td>
<td>dconv ( \hat{F}_1 )</td>
<td>LeakyReLU ( {b_4, {b_6:16}, {b_7:32}} ) ReLU ( {9,1} ) ( 4 \times bc ) 1 ( {5,1} ) 1 ( 2 \times bn ) ( \hat{F}_1 )</td>
</tr>
<tr>
<td>( \downarrow )</td>
<td>res ( d_1 )</td>
<td>( {b_4, {b_6:16}, {b_7:32}} ) ( {b_4, {b_6:16}, {b_7:32}} ) ( {9,1} ) ( 4 \times bc ) 1 ( {5,1} ) 1 ( 2 \times bn ) ( \hat{F}_1 )</td>
</tr>
</tbody>
</table>

\( \downarrow \) Reblurring Framework. Shared Encoder, PSF Blocks, and Decoder.
|-------------|--------------|-------------|--------------|----------------|------|

Figure 14. Comparison of image restoration performance on the DPD-blur dataset [1]. The large sharp images in the first column are ground-truth sharp images. The small sharp images in the second column are cropped images from the green bounding box in the large ground-truth sharp images. The blurred images in the second column are corresponding input blurry images ($B_L$).
Figure 15. Comparison of image restoration performance on the DPD-blur dataset [1]. The large sharp images in the first column are ground-truth sharp images. The small sharp images in the second column are cropped images from the green bounding box in the large ground-truth sharp images. The blurred images in the second column are corresponding input blurry images (B_L).
Figure 16. Comparison of image restoration performance on the DPD-blur dataset [1]. The large sharp images in the first column are ground-truth sharp images. The small sharp images in the second column are cropped images from the green bounding box in the large ground-truth sharp images. The blurred images in the second column are corresponding input blurry images (B).
Figure 17. Comparison of image restoration performance on the DPD-blur dataset [1]. The large sharp images in the first column are ground-truth sharp images. The small sharp images in the second column are cropped images from the green bounding box in the large ground-truth sharp images. The blurred images in the second column are corresponding input blurry images ($B_L$).
Figure 18. Comparison of image restoration performance on the DPD-blur dataset [1]. The large sharp images in the first column are ground-truth sharp images. The small sharp images in the second column are cropped images from the green bounding box in the large ground-truth sharp images. The blurred images in the second column are corresponding input blurry images (B_L).
References


[22] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibil-


