Supplementary Material
BokehMe: When Neural Rendering Meets Classical Rendering

Juewen Peng[1], Zhiguo Cao[1], Xianrui Luo[1], Hao Lu[1], Ke Xian[1]*, and Jianming Zhang[2]
[1]Key Laboratory of Image Processing and Intelligent Control, Ministry of Education,
School of Artificial Intelligence and Automation, Huazhong University of Science and Technology
[2]Adobe Research
{juewenpeng, zgcao, xianruiluo, hlu, kexian}@hust.edu.cn, jianmzha@adobe.com

https://github.com/JuewenPeng/BokehMe

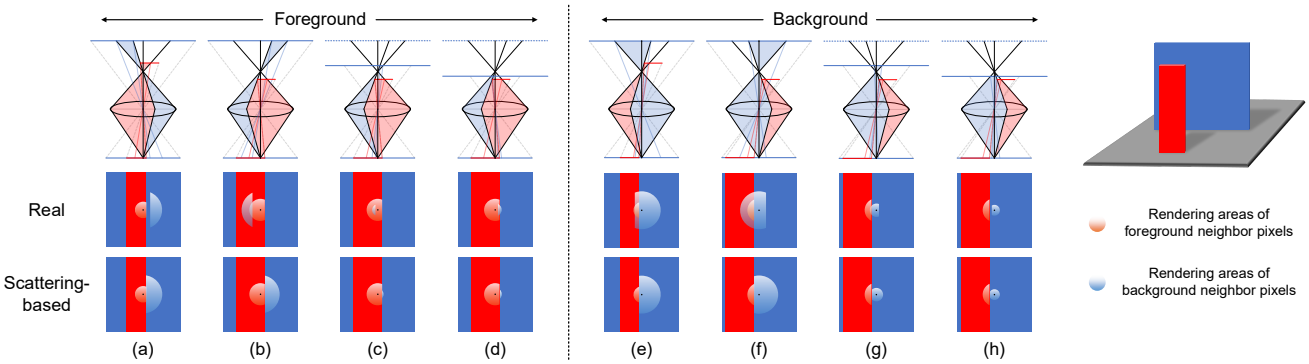

Figure 1. Comparison of real rendering and scattering-based rendering near depth discontinuities. By changing the relative positions of foreground plane (red) and background plane (blue), one can derive 8 different cases: 4 for foreground rendering; 4 for background rendering. Rendered result of the center pixel (the black dot) is the integration of its neighbor pixels on red gradient foreground plane and those on blur gradient background plane.

This document includes the following contents:

1. Extended discussion on error analysis for scattering-based rendering.
2. Algorithm implementations, including the error map generation, the scattering-based rendering method, and the dilated defocus map generation.
3. Synthesis of the training data.
4. Sensitivity experiments of the hyperparameters δ_1 and δ_2 used in the error map generation.
5. The role of the dilated defocus map in IUNet.
6. Details of the user study.
7. Additional experiments, including the comparison of fusion with error map and CNN-based fusion, and the comparison of fusing with different classical methods.
8. More qualitative comparisons with other methods.

1. Extended Discussion on Error Analysis

We show the complete 8 rendering cases in Fig. 1. In each case, we take the center pixel (the black dot) as an example and visualize the results of real rendering and scattering-based rendering. Since in the real rendering, there may be widespread occluded areas involved, *e.g.*, Fig. 1 (b), which are hard to estimate, a common practice is to assume the surface is symmetric around the pixel of interest [1, 9]. In this way, (b), (c), (f), (g) are equivalent to (a), (d), (e), (h), respectively. Thus, the aforementioned 8 cases can be simplified into 4 cases, *i.e.*, (a), (d), (e), (h).

1.1. Numerical Analysis

In the following, we will calculate the color difference between the real rendering result and the scattering-based rendering result for the simplified 4 cases. Note that all undefined mathematical symbols are annotated in Fig. 2.

Case of Fig. 2 (a). Real rendering area of the foreground is

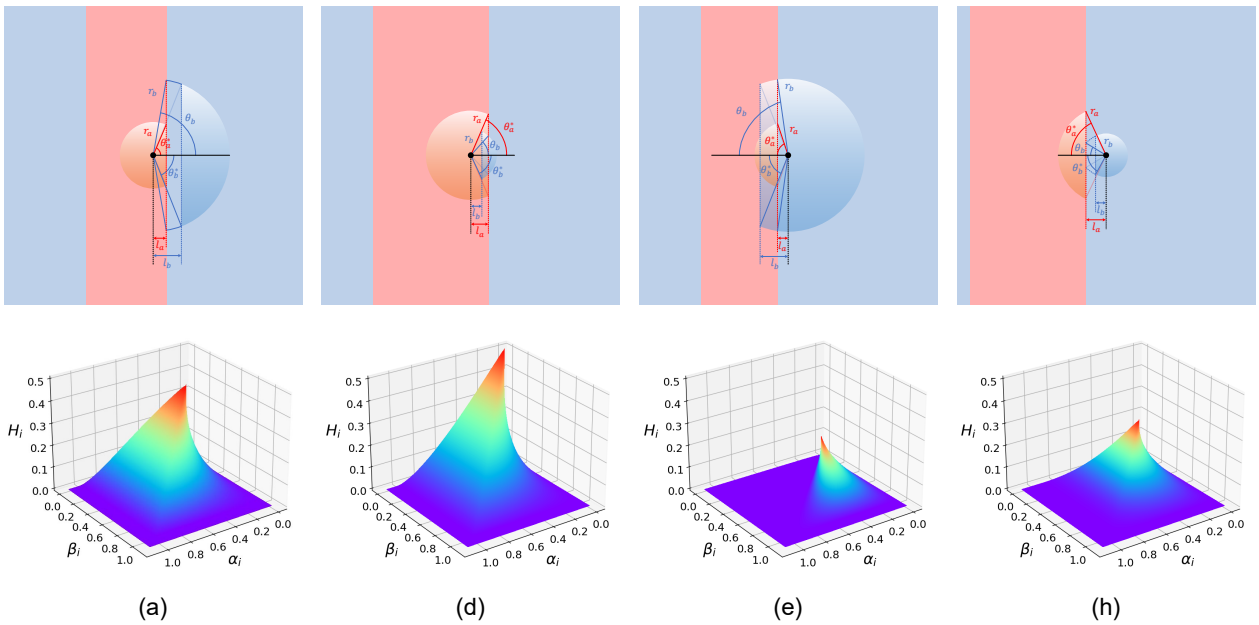

Figure 2. Simplified 4 rendering cases from Fig. 1 and corresponding graphs of the color difference between real rendering and scattering-based rendering. Subscript i for all symbols in the graphs denotes the pixel being processed. In this example, it denotes the center pixel (the black dot).

$$S_a^* = \pi r_a^2 \left(1 - \frac{2\theta_a^*}{2\pi}\right) + r_a \sin\theta_a^* \cdot r_a \cos\theta_a^* = r_a^2 \left[\pi - (\theta_a^* - \sin\theta_a^* \cos\theta_a^*)\right], \quad (1)$$

where

$$\theta_a^* = \arccos\frac{l_a}{r_a}. \quad (2)$$

Real rendering area of the background is

$$S_b^* = \pi r_b^2 \cdot \frac{2\theta_b^*}{2\pi} - r_b \sin\theta_b^* \cdot r_b \cos\theta_b^* = r_b^2 \left(\theta_b^* - \sin\theta_b^* \cos\theta_b^*\right), \quad (3)$$

where

$$\theta_b^* = \arccos\frac{l_b}{r_b}. \quad (4)$$

Scattering-based rendering area of the background is

$$S_b = \pi r_b^2 \cdot \frac{2\theta_b}{2\pi} - r_b \sin\theta_b \cdot r_b \cos\theta_b = r_b^2 \left(\theta_b - \sin\theta_b \cos\theta_b\right), \quad (5)$$

where

$$\theta_b = \arccos\frac{l_a}{r_b}. \quad (6)$$

According to the triangle similarity theorem, we can get

$$\frac{l_b}{l_a} = \frac{r_b}{r_a} = \frac{z_a}{z_b} \cdot \left|\frac{z_b - z_f}{z_a - z_f}\right|, \quad (7)$$

where z_a and z_b are the depths of the foreground and the background. z_f is the refocused depth. Since $r_b > r_a$, we can further derive

$$\theta_a^* = \theta_b^* \leq \theta_b. \quad (8)$$

Let

$$m = \theta_a^* - sin\theta_a^* \cos\theta_a^* = \theta_b^* - \sin\theta_b^* \cos\theta_b^*, \quad (9)$$

$$n = \theta_b - \sin\theta_b \cos\theta_b, \quad (10)$$

we can get

$$0 \leq m \leq n \leq \frac{\pi}{2}. \quad (11)$$

Considering the energy conservation, the weight of each pixel should be divided by the square of its blur radius during the fusion. Assume the foreground color is c_a and the background color is c_b, the real rendering result is

$$B^* = \frac{\frac{S_a^*}{r_a^2} \cdot c_a + \frac{S_b^*}{r_b^2} \cdot c_b}{\frac{S_a^*}{r_a^2} + \frac{S_b^*}{r_b^2}} = \frac{(\pi - m)\, c_a + m\, c_b}{\pi}. \quad (12)$$

The scattering-based rendering result is

$$B = \frac{\frac{S_a^*}{r_a^2} \cdot c_a + \frac{S_b}{r_b^2} \cdot c_b}{\frac{S_a^*}{r_a^2} + \frac{S_b}{r_b^2}} = \frac{(\pi - m)\, c_a + n\, c_b}{\pi - m + n}. \quad (13)$$

Therefore, the color difference between the real rendering result and the scattering-based rendering result is

$$H = \left|B^* - B\right| = \frac{(\pi - m)\,(n - m)}{\pi\,(\pi - m + n)} \cdot \left|c_a - c_b\right|. \quad (14)$$

We can derive that while $m = 0$, $n = \frac{\pi}{2}$, the maximum value of the color difference is obtained.

$$H_{max} = \frac{1}{3} \left| c_a - c_b \right| . \tag{15}$$

We can also derive that at this moment,

$$\frac{l_a}{r_b} = 0 , \quad \frac{r_a}{r_b} = 0 . \tag{16}$$

Following this formulation, we can calculate the color difference of other 3 cases.

Case of Fig. 2 (d). The color difference is

$$H = \left| B^* - B \right| = \frac{(\pi - m)(m - n)}{\pi (\pi - m + n)} \cdot \left| c_a - c_b \right| , \tag{17}$$

where $0 \leq n \leq m \leq \frac{\pi}{2}$. While $m = \frac{\pi}{2}$, $n = 0$, the maximum value of the color difference is obtained.

$$H_{max} = \frac{1}{2} \left| c_a - c_b \right| . \tag{18}$$

At this moment,

$$\frac{l_a}{r_a} = 0 , \quad \frac{r_b}{r_a} = 0 . \tag{19}$$

Case of Fig. 2 (e). The color difference is

$$H = \left| B^* - B \right| = \frac{m (n - m)}{\pi (\pi - n + m)} \cdot \left| c_a - c_b \right| , \tag{20}$$

where $0 \leq m \leq n \leq \frac{\pi}{2}$. While $m = \frac{(\sqrt{2}-1)\pi}{2}$, $n = \frac{\pi}{2}$, the maximum value of the color difference is obtained.

$$H_{max} = \left(\frac{3}{2} - \sqrt{2} \right) \left| c_a - c_b \right| . \tag{21}$$

At this moment,

$$\frac{l_a}{r_b} = 0 , \quad \frac{r_a}{r_b} = 0 . \tag{22}$$

Case of Fig. 2 (h). The color difference is

$$H = \left| B^* - B \right| = \frac{m (m - n)}{\pi (\pi - n + m)} \cdot \left| c_a - c_b \right| , \tag{23}$$

where $0 \leq n \leq m \leq \frac{\pi}{2}$. While $m = \frac{\pi}{2}$, $n = 0$, the maximum value of the color difference is obtained.

$$H_{max} = \frac{1}{6} \left| c_a - c_b \right| . \tag{24}$$

At this moment,

$$\frac{l_a}{r_a} = 0 , \quad \frac{r_b}{r_a} = 0 . \tag{25}$$

Algorithm 1: Error map calculation

Input: Depth boundary mask M, gradient maps G^x, G^y, gradient magnitude map G, signed defocus map S, hyperparameters δ_1, δ_2
Output: Error map E^*

$E^* \leftarrow [0]$;
for $p_i \leftarrow$ `TraverseImage(`M`)` **do**
  if $M_i > 0$ **then**
    $(p_{i+}, p_{i-}) \leftarrow$ `Sample(`p_i, G_i^x, G_i^y, G_i`)` ;
    $r_{i+} \leftarrow |S_{i+}|$;
    $r_{i-} \leftarrow |S_{i-}|$;
    $r_{min} \leftarrow \min(r_{i+}, r_{i-})$;
    $r_{max} \leftarrow \max(r_{i+}, r_{i-})$;
    for $p_j \leftarrow$ `TraverseNeighbor(`p_i, r_{max}`)` **do**
      $\alpha \leftarrow \frac{l_{ij}}{r_{max}}$;
      $\beta \leftarrow \frac{r_{min}}{r_{max}}$;
      $e_{ij} \leftarrow (1-\alpha^{\delta_1}) \cdot (\frac{1}{2} + \frac{1}{2} \tanh(10(\delta_2 - \beta)))$;
      $E_j^* \leftarrow \max(E_j^*, e_{ij})$;
    end
  end
end

1.2. Graphical Analysis

We first define two variables by

$$\alpha = \frac{l_a}{\max(r_a, r_b)} , \quad \beta = \frac{\min(r_a, r_b)}{\max(r_a, r_b)} . \tag{26}$$

For the case of Fig. 2 (a) and (e), as $r_a < r_b$, we can infer

$$\alpha = \frac{l_a}{r_b} , \quad \beta = \frac{r_a}{r_b} , \tag{27}$$

so

$$\frac{l_a}{r_a} = \frac{\alpha}{\beta} , \quad \frac{l_a}{r_b} = \alpha . \tag{28}$$

Similarly, for the case of Fig. 2 (d) and (h), as $r_a > r_b$, we can infer

$$\frac{l_a}{r_a} = \alpha , \quad \frac{l_a}{r_b} = \frac{\alpha}{\beta} . \tag{29}$$

According to Eqs. (2), (6), (9), (10), (14), (17), (20), (23), (28) and (29), we can further represent H by α and β. As the explicit expression is complicated, we use the method of exhaustion to calculate H under different α and β for the 4 simplified cases, and draw their graphs in Fig. 2. Note that we assume $|c_a - c_b| = 1$ in visualization of the graphs. One can observe that for each case, the maximum value of H is consistent with the conclusion we derived in Sec. 1.1.

2. Algorithm Implementations

2.1. Error Map Generation

The target error map E^* can be represented as the spatially variant dilation of the depth boundary mask. In detail,

Algorithm 2: Pixel-wise scattering-based rendering method

Input: All-in-focus image I, signed defocus map S, gamma value γ
Output: Bokeh image B_{cr}

1 $I \leftarrow (I)^{\gamma}$;
2 $W \leftarrow [0]$;
3 $C \leftarrow [0]$;
4 **for** $p_i \leftarrow$ `TraverseImage(`I`)` **do**
5 $r_i \leftarrow |S_i|$;
6 **for** $p_j \leftarrow$ `TraverseNeighbor(`p_i, r_i`)` **do**
7 $w_{ij} \leftarrow \frac{0.5+0.5\tanh(4(r_i - l_{ij}))}{{r_i}^2+0.2}$;
8 $W_j \leftarrow W_j + w_{ij}$;
9 $C_j \leftarrow C_j + w_{ij} \cdot I_i$;
10 **end**
11 **end**
12 $B_{cr} \leftarrow \frac{C}{W}$;
13 $B_{cr} \leftarrow (B_{cr})^{\frac{1}{\gamma}}$;

given a disparity map D, we use the Sobel operator to get the gradient maps G^x and G^y and the gradient magnitude map G. Then, the depth boundary mask M can be calculated by

$$M = \mathbb{1}(G > \zeta), \tag{30}$$

where ζ is a threshold. For each pixel $p_i = (x_i, y_i)$ in the one-filled areas of M, we sample 2 neighbor pixels $p_{i^+} = (x_{i^+}, y_{i^+})$ and $p_{i^-} = (x_{i^-}, y_{i^-})$ along the gradient direction by

$$x_{i^+} = \left\lfloor x_i + \frac{G_i^x}{G_i} \right\rceil, \quad y_{i^+} = \left\lfloor y_i + \frac{G_i^y}{G_i} \right\rceil, \tag{31}$$

and

$$x_{i^-} = \left\lfloor x_i - \frac{G_i^y}{G_i} \right\rceil, \quad y_{i^-} = \left\lfloor y_i - \frac{G_i^x}{G_i} \right\rceil, \tag{32}$$

where the bracket represents the round function. According to Eq. (5) in the main paper, we can further calculate the dilation weight e_{ij} from pixel p_i to its neighbor pixel p_j, and update E_j^* to e_{ij} if $e_{ij} > E_j^*$. The whole generation process is described in Alg. 1, where function `Sample()` is equivalent to Eqs. (31) and (32). Function `TraverseImage()` is used to traverse all pixels of the image M, while function `TraverseNeighbor()` is used to traverse the neighbors of the processing pixel p_i within the range of $[-r_{max}, r_{max}]$ in both x-direction and y-direction.

2.2. Scattering-based Rendering Method

We implement the classical renderer by a pixel-wise scattering-based rendering method. In Alg. 2, we assume the aperture shape is circular. Starting from an all-in-focus image I, a corresponding signed defocus map S, and

Algorithm 3: Dilated defocus map calculation

Input: Signed defocus map S
Output: Dilated defocus map S^d

1 $S^d \leftarrow S$;
2 **for** $p_i \leftarrow$ `TraverseImage(`I`)` **do**
3 $r_i \leftarrow |S_i|$;
4 **for** $p_j \leftarrow$ `TraverseNeighbor(`p_i, r_i`)` **do**
5 $s_{ij} \leftarrow \mathbb{1}\left(\frac{l_{ij}}{r_i} < 1\right) \cdot S_i$;
6 $S_j^d \leftarrow \max(S_j^d, s_{ij})$;
7 **end**
8 **end**

gamma value γ, we first apply gamma transformation [4] to transform I to linear space. Two accumulation buffers W and C are zero-initialized. We traverse all pixels of I by function `TraverseImage()`. The blur radius r_i can be calculated by the absolute value of S_i. Then we traverse the neighbor pixels of p_i by function `TraverseNeighbor()`. In the inner loop, we first calculate the weight w_{ij} from p_i to neighbor pixel p_j. For p_i, it can only scatter to p_j if r_i is larger than the distance l_{ij} between p_i and p_j. Here, we use a soft version in the calculation of the weight w_{ij} to produce bokeh balls with smooth and natural boundaries. w_{ij} is additionally divided by the square of r_i due to the uniform distribution of energy. The calculated w_{ij} and the multiplication of w_{ij} and intensity I_i are then accumulated in W_j and C_j, respectively. After traversing all pixels, the final result B_{cr} can be obtained by the element-wise division of two accumulated buffers and the subsequent inverse gamma transformation.

To create polygonal aperture shapes, We can multiply r_i in Alg. 2 row 7 by a factor of k_{ij}, which is defined by

$$k_{ij} = \frac{\sin\left(\frac{\pi}{2} - \frac{\pi}{n}\right)}{\sin\left(\frac{\pi}{2} - \frac{\pi}{n} + \mathrm{mod}\left(\left|\arctan\left(\frac{l_{ij}^y}{l_{ij}^x}\right) + \phi\right|, \frac{2\pi}{n}\right)\right)}, \tag{33}$$

where l_{ij}^x and l_{ij}^y are the horizontal and vertical component of distance l_{ij}. n denotes the number of aperture blades and ϕ controls the rotation angle of polygonal bokeh balls.

2.3. Dilated Defocus Map Generation

Dilated defocus map S^d is used in neural renderer. It is a spatially variant dilation of the signed defocus map S. The dilation size depends on the blur radius of each pixel. The detailed generation of S^d is described in Alg. 3, where most operations are the same with Alg. 2. As a result, we can calculate B_{cr} and S^d in one pipeline for efficiency.

3. Training Dataset

To train BokehMe, we require that each training sample contains an all-in-focus image, a disparity map and a

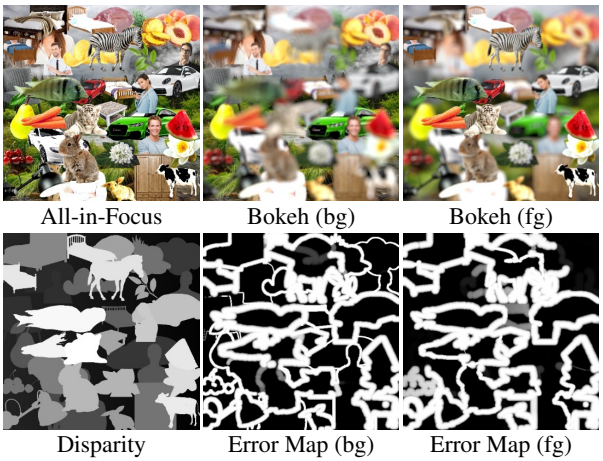

Figure 3. An example of the training data. The displayed bokeh images are synthesized with the following settings: $K = 12$, $\gamma = 2$, $d_f = 0.05$ for the second column, and $d_f = 1$ for the third column. We also show the corresponding error maps calculated by Alg. 1, where $\delta_1 = 4$, $\delta_2 = \frac{2}{3}$.

Table 1. Sensitivity experiments of δ_1 and δ_2 in the error map generation. Results are evaluated by PSNR.

δ_1 \ δ_2	0	1/2	2/3	5/6	1
0	38.35	38.35	38.35	38.35	38.35
2	38.35	40.80	40.98	40.93	40.39
4	38.35	40.75	**41.02**	40.48	40.41
6	38.35	40.66	40.75	40.40	40.33
∞	38.35	40.57	40.70	40.36	40.33

bokeh image with known blur parameter, refocused disparity, and gamma value. Existing public bokeh datasets, such as EBB! [3] and Unity [8] cannot meet our requirements, so we introduce a new synthetic bokeh dataset. The synthesis procedure is as follows: (i) We collect 150 background images and 300 foreground images with pure color background from websites. For the foreground images, we extract their alpha maps by online matting tools. (ii) For each scene, we randomly choose 36 foreground objects with random resizing and color augmentation, and place them on a background image from back to front according to their disparities. The disparity of the background image is set to a random plane while the disparity of each foreground object is set to a random value which is larger than the maximum disparity of the background. (iii) We use a simplified ray tracing method to obtain multiple bokeh images with different controlling parameters. We show an example of our training data in Fig. 3.

4. Sensitivity Experiments of δ_1 and δ_2

The hyperparameters δ_1 and δ_2 control the softness and narrowness of the target error map, which is used to fuse the results from the physical renderer and the neural renderer. Here, we analyze the impact of different combinations of both hyperparameters on the final result. The experiment is conducted on the BLB dataset (Level 3). Higher δ_1 and δ_2 represent the larger available area for the neural renderer. As shown in Table 1, the best performance is achieved while $\delta_1 = 4$ and $\delta_2 = \frac{2}{3}$.

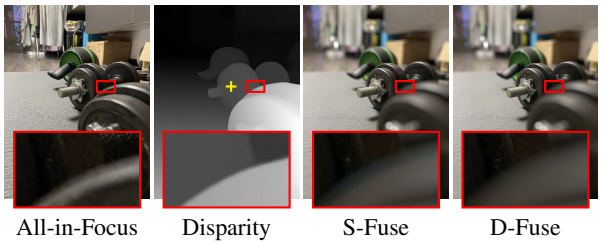

Figure 4. Comparison of "S-Fuse" and "D-Fuse" in IUNet. "S-Fuse": fusion with the mask thresholded by the signed defocus map; "D-Fuse": fusion with the mask thresholded by the dilated defocus map. The rough refocused plane is labelled with a yellow cross on the disparity map.

5. The Role of Dilated Defocus Map

In IUNet, we use the dilated defocus map instead of the signed defocus map to produce a mask, and use it to fuse the output of the neural network and the bilinear upsampled input bokeh image. The reason is that the negative effects caused by defocus clipping will spread during the rendering, especially when the focal plane targets background. We visualize an example in Fig. 4. One can see that using the dilated defocus map leads to a more natural color transition around the object boundary.

6. User Study

We build a website for user study (Fig. 5). For each scene, we provide an all-in-focus image captured by iPhone 12, a darker image labelling the focal point with a yellow cross, and the results of different rendering methods displayed in a random order. To enable detailed observation, we provide two magnification windows for the images in two rows. When the mouse moves over the image, the magnification window will provide local zoomed viewing. Participants are required to select one option from "Good", "Normal", and "Bad" for each anonymous method. The final comparison is based on the number of votes.

7. Additional Experiments

In this section, all experiments are conducted on the BLB dataset (Level 3).

Comparison of fusion with error map and CNN-based fusion. We fuse the results of the classical renderer and the neural renderer by the predicted error map. It aims to

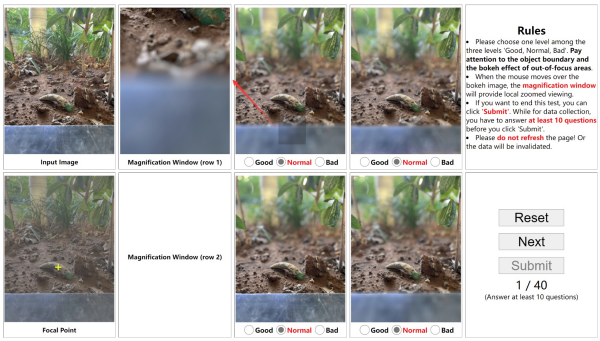

Figure 5. Interface of the website for user study.

separate the rendered areas of the two renderers as much as possible, which can avoid bokeh style confusion, especially when using a non-circular aperture in the classical renderer. In addition, we show in Table 2 that this way outperforms CNN-based fusion quantitatively. The fusion network is made up of three 3×3 convolution layers (32 channels) with the defocus map and the two rendered results as input.

Comparison of fusing with different classical methods. We use the scattering-based rendering method as our classical renderer because it works well in depth-continuous areas. Although this method causes serious artifacts at depth discontinuities, these areas will be replaced with the results of the neural renderer. In contrast, other classical methods sacrifice the accuracy on depth-continuous areas for more natural boundary transition, resulting in lower performance when combining with our neural renderer (Table 3).

8. More Qualitative Results

We show more comparison results with other methods on 3 test datasets: BLB (Fig. 6), EBB400 (Fig. 7), and IPB (Figs. 8 and 9). One can observe that BokehMe can remove boundary artifacts in classical rendering methods and overcome the limited blur range and poor highlight rendering in neural rendering methods. Note that Wang *et al*. [7] claim that their proposed DeepLens can generate high-resolution results of arbitrary resolutions. However, the released code can only handle the image up to 2K. Therefore, for the IPB dataset with the image resolution of 3024×4032, bokeh images produced by DeepLens are a little blurry within in-focus areas. In addition, despite being superior to DeepFocus [8], DeepLens still lacks a good mechanism to produce arbitrarily large blur sizes. Two examples are shown in the first two rows of Fig. 7. DeepLens produces abnormal colors in out-of-focus areas, while our approach maintains high-quality rendering. Another thing that should be noted is that by using disparity augmentation during training, our approach is able to handle slightly corrupted disparity boundaries but cannot fix the missing structure. However, we observe that for most scenes, our rendered results are natural without obvious artifacts.

Table 2. Comparison of fusion with error map (Err-fusion) and CNN-based fusion (CNN-fusion).

Methods	CNN-fusion	Err-fusion (Ours)
PSNR	40.53	**41.02**
SSIM	0.9909	**0.9915**

Table 3. Comparison of fusing with different classical methods.

Methods	NR	NR+VDSLR [9]	NR+SteReFo [2]	NR+Scatter (Ours)
PSNR	39.21	40.44	37.12	**41.02**
SSIM	0.9896	0.9900	0.9872	**0.9915**

References

[1] Guillaume Abadie, Steve McAuley, Evegenii Golubev, Stephen Hill, and Sebastien Lagarde. Advances in real-time rendering in games. In *ACM SIGGRAPH 2018 Courses*, pages 1–1. 2018. 1

[2] Benjamin Busam, Matthieu Hog, Steven McDonagh, and Gregory Slabaugh. Sterefo: Efficient image refocusing with stereo vision. In *Proc. IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 0–0, 2019. 6, 7, 8

[3] Andrey Ignatov, Jagruti Patel, and Radu Timofte. Rendering natural camera bokeh effect with deep learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 418–419, 2020. 5

[4] Haiting Lin, Seon Joo Kim, Sabine Süsstrunk, and Michael S Brown. Revisiting radiometric calibration for color computer vision. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 129–136. IEEE, 2011. 4

[5] Photopea. https://www.photopea.com. 9, 10

[6] René Ranftl, Katrin Lasinger, D Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 8

[7] Lijun Wang, Xiaohui Shen, Jianming Zhang, Oliver Wang, Zhe Lin, Chih-Yao Hsieh, Sarah Kong, and Huchuan Lu. Deeplens: Shallow depth of field from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 6, 7, 8, 9, 10

[8] Lei Xiao, Anton Kaplanyan, Alexander Fix, Matthew Chapman, and Douglas Lanman. Deepfocus: Learned image synthesis for computational displays. *ACM Transactions on Graphics (TOG)*, 37(6):1–13, 2018. 5, 6, 7, 8

[9] Yang Yang, Haiting Lin, Zhan Yu, Sylvain Paris, and Jingyi Yu. Virtual dslr: High quality dynamic depth-of-field synthesis on mobile platforms. *Electronic Imaging*, 2016(18):1–9, 2016. 1, 6, 7, 8, 9, 10

[10] Xuaner Zhang, Kevin Matzen, Vivien Nguyen, Dillon Yao, You Zhang, and Ren Ng. Synthetic defocus and look-ahead autofocus for casual videography. *ACM Transactions on Graphics (TOG)*, 38:1 – 16, 2019. 7

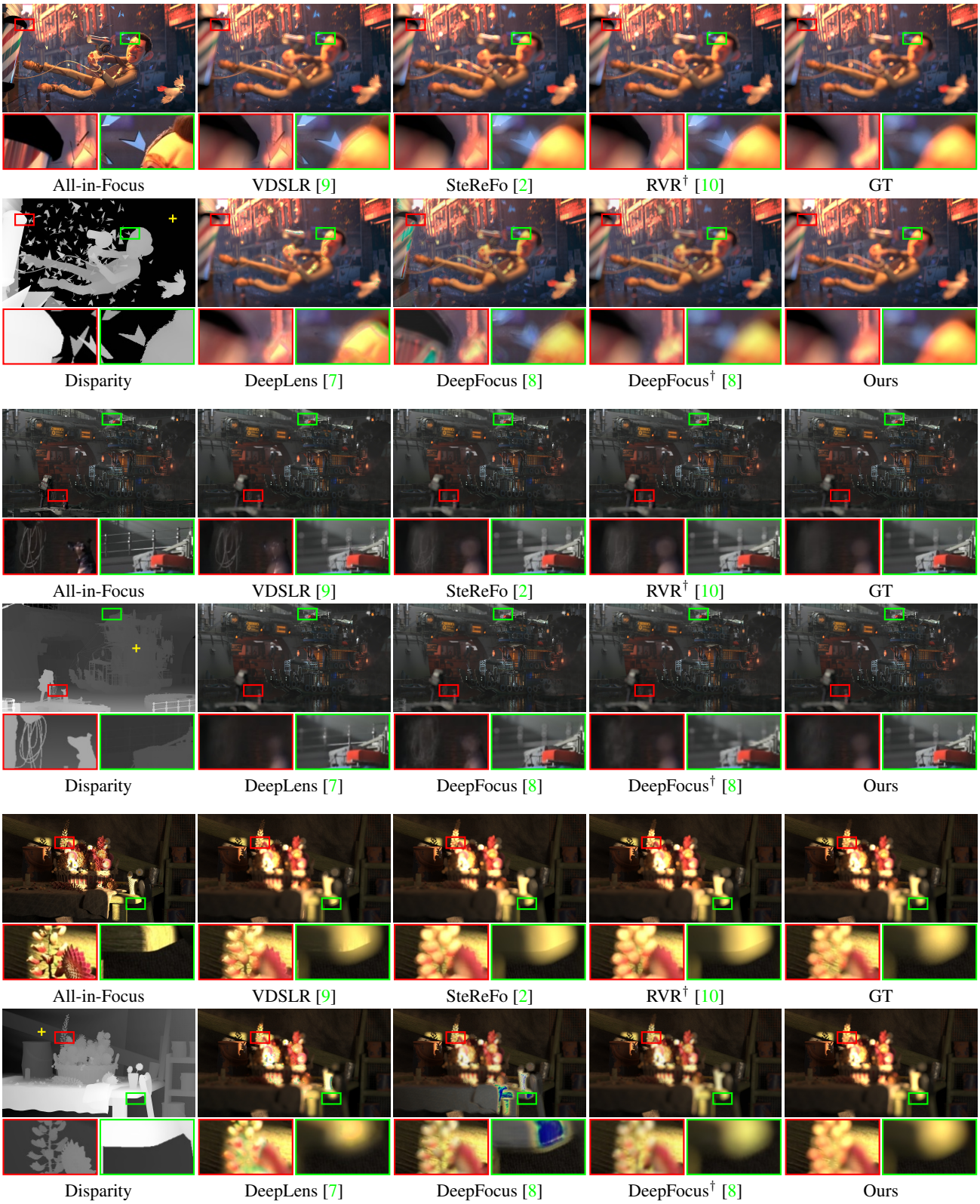

Figure 6. Qualitative results on the BLB dataset. The rough refocused plane is labelled with a yellow cross on the disparity map.

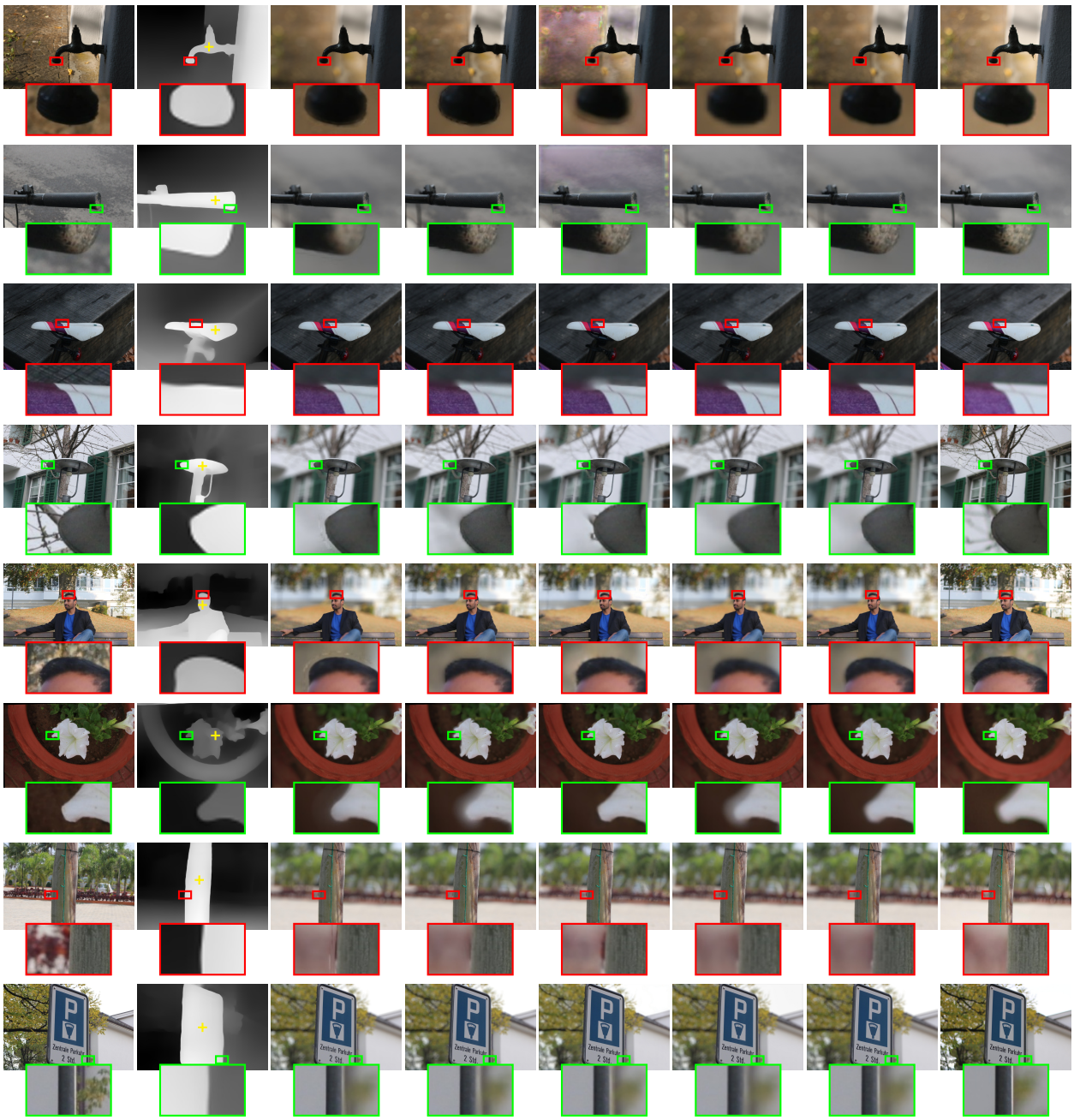

Figure 7. Qualitative results on the EBB400 dataset. "All-in-Focus" and "GT" refer to the wide and narrow DoF images captured by Canon 7D DSLR camera. The rough refocused plane is labelled with a yellow cross on the disparity map predicted by MiDaS [6].

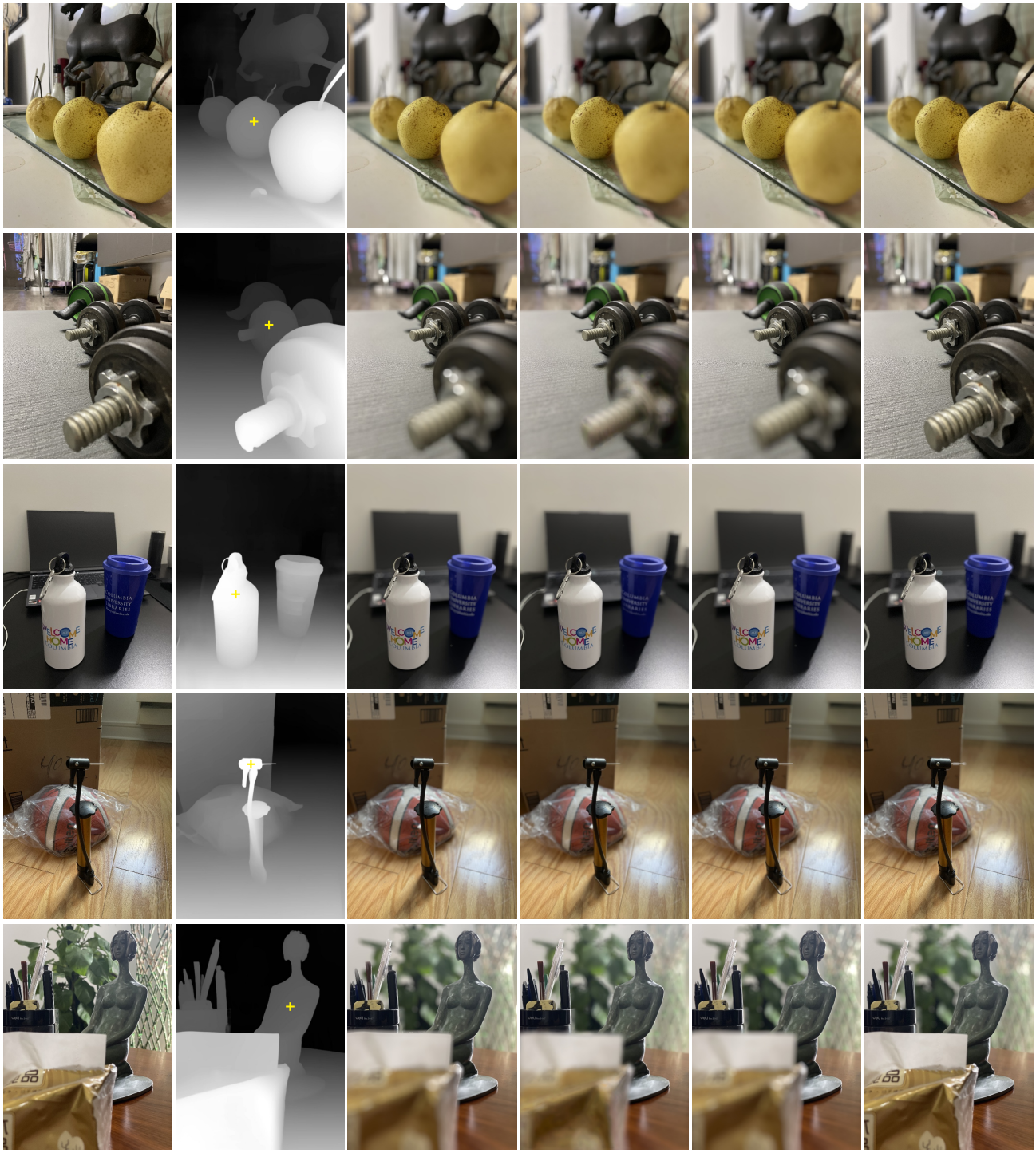

Figure 8. Qualitative results of user study on the IPB dataset. "All-in-Focus" refers to the wide DoF image captured by iPhone 12. The rough refocused plane is labelled with a yellow cross on the disparity map extracted from the captured image by Photopea [5].

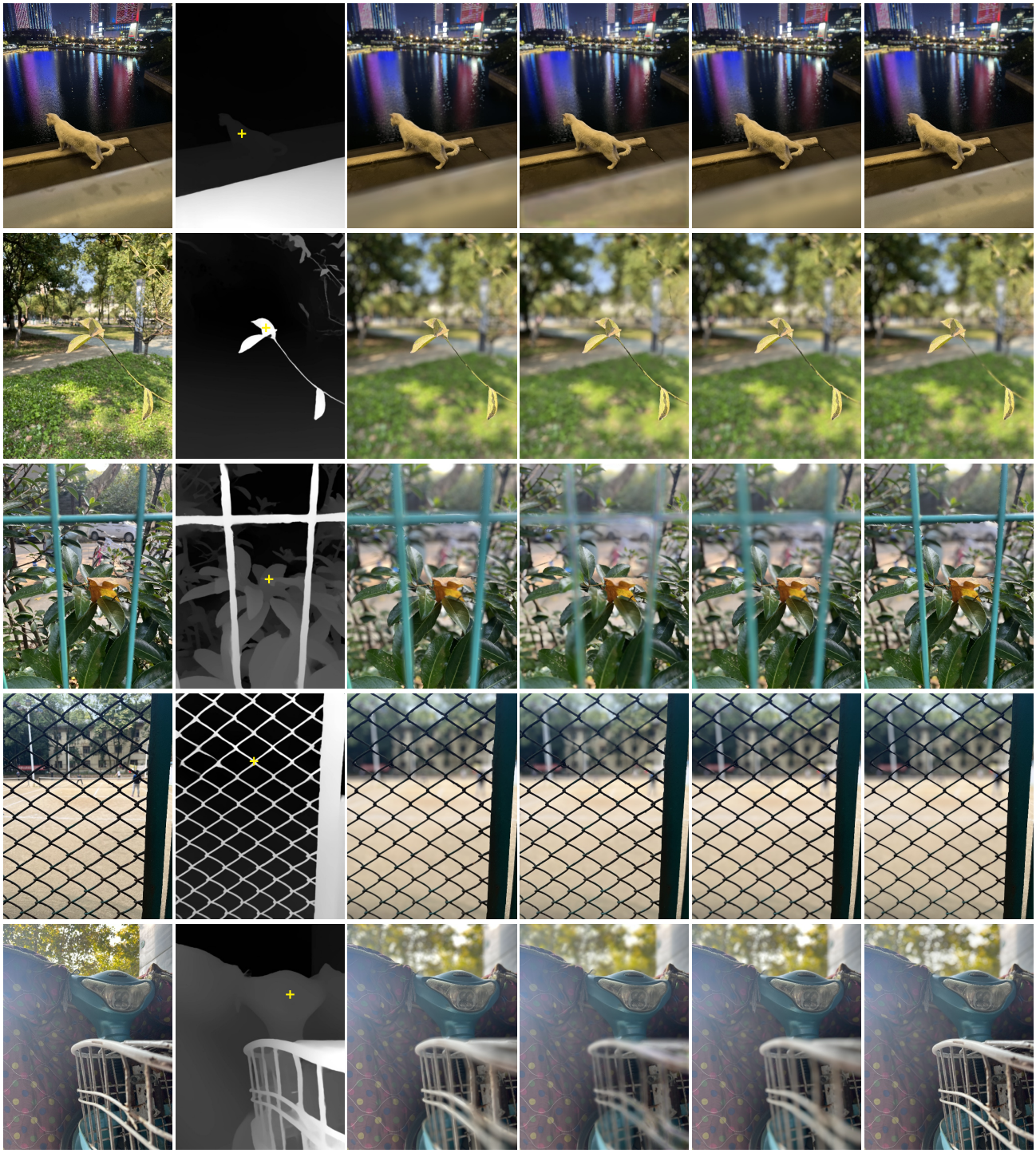

Figure 9. Qualitative results of user study on the IPB dataset. "All-in-Focus" refers to the wide DoF image captured by iPhone 12. The rough refocused plane is labelled with a yellow cross on the disparity map extracted from the captured image by Photopea [5].