Supplemental Material for the Manuscript
Simulating Dual-Pixel Images From Ray Tracing For Depth Estimation

Fengchen He, Dayang Zhao, Hao Xu, Tingwei Quan, Shaoqun Zeng
Huazhong University of Science and Technology
{linyark, dayangzhao, xuhao_2003, quantingwei, sqzeng}@hust.edu.cn

S1. Sdirt Details

S1.1. Structural parameters of the DP pixel

We select the Canon RF50mm F/1.8 lens and Canon R6 Mark II camera body as our real **reference lens** and DP sensor in the main paper. Since the reference lens data are available, with detailed specifications presented in Tab. S1 and the 2D lens layout with ray paths shown in Fig. S1(a), it is feasible to perform ray tracing on the lens. In practice, many commercial lens data, including Canon lenses, are available on open-access websites such as the Optical Bench, which facilitates optical modeling. However, performing ray tracing on the DP sensor is limited by camera manufacturers' nondisclosure of the structural parameters of the microlens and sub-pixel components within the DP pixel. Therefore, as described in Sec. 3.1 (Ray-traced DP PSF simulator) and Fig. 3(a) (DP pixel structure layout) of the main paper, we simplify the DP pixel structure by modeling the microlens as a thin lens with radius r and focal length f, defining h as the distance between the sub-pixel and the microlens, w as the sub-pixel width, and ps as the DP pixel size.

We assign a set of possible values to each structural parameter of the DP pixel and identify the optimal parameter combination through grid searching across these parameters. Instead of conducting search experiments directly on the reference lens, we employ the Canon RF35mm F/1.8 as an **isolation lens** to replace the reference lens for probing DP pixel structural parameters, thereby ensuring isolation between the reference lens and the sensor. Detailed lens data for the Canon RF35mm F/1.8 are provided in Tab. S2, and its 2D lens layout with ray paths is shown in Fig. S1(b). In the valid imaging region (Fig. 4(b) in the main paper), we select 5 object points at different positions and capture real DP PSFs through the employed camera. By evaluating the error and similarity between the real DP PSFs and the set of DP PSFs simulated with all DP pixel structural parameters, we determine an optimal parameter combination:

$$\begin{cases} h & = 0.78 * ps \\ f & = 1.44 * ps \\ w & = 0.30 * ps \\ r & = 0.50 * ps \end{cases} \tag{S1}$$

Specifically, we choose the normalized squared difference (NSD) and normalized cross-correlation (NCC) methods from OpenCV to evaluate the error and similarity between the real and simulated DP PSFs. During the search experiments, we set the F-number to F/4.0, set the DP PSF kernel size to 35, and set the focal distance to infinity. As shown in Fig. S2, we present the results of the error and similarity matching. For ease of presentation, when displaying the matching results for any one parameter, we fix the other parameters at their optimal values according to Eq. (S1).

To demonstrate the similarity between real and simulated DP PSFs under the optimal combination of structural parameters, we conduct a qualitative analysis while keeping the lens, F-number, kernel size, and focal distance unchanged from the grid search experiments. As shown in Fig. S3, we select five points evenly spaced within the valid imaging region, with only their x-coordinates increasing sequentially. Additionally, to visually compare changes in pixel values, a line plot of pixel distribution along the central row is also included in each subplot. Observing the actual DP PSF, we notice that as the object point p moves farther from the optical axis, the real PSF_L and PSF_R become more phase asymmetric. This observation aligns with the experimental results presented in Sec. 5.3 (Evaluation on simulated DP PSFs) of the main paper. When comparing the ray-traced and real DP PSFs, we find that the ray-traced DP PSF is not only highly realistic in morphology but also closely matches the real data in pixel values.

S1.2. DP PSF capture and linearization

The native resolution of the Canon R6 Mark II is 6000×4000 pixels. Due to GPU memory limitations, we reduced the spatial resolution to 768×512. Although this resolution

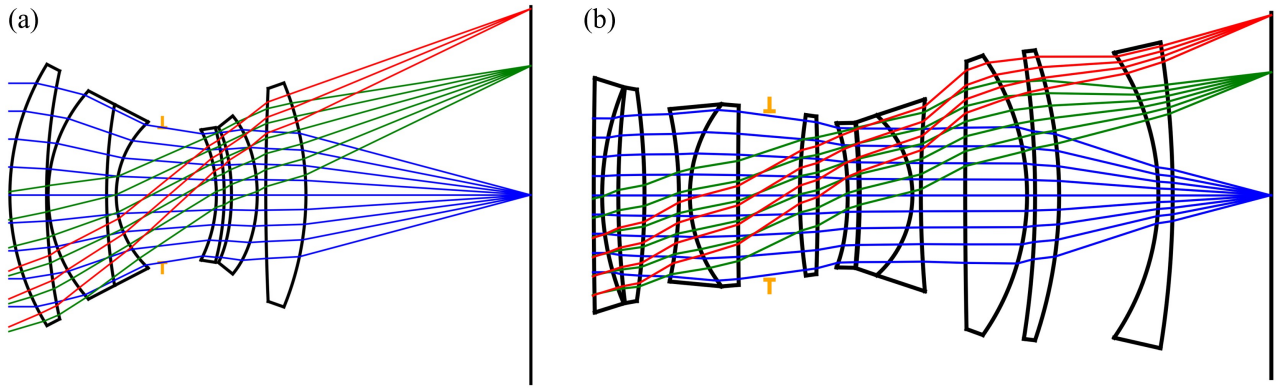

Figure S1. **The 2D lens layout with ray paths.** (a) Canon RF50mm F/1.8 lens. (b) Canon RF35mm F/1.8 lens.

Table S1. **Canon RF50mm F/1.8 lens data.** The real reference lens used in the main paper.

Surface	Radius (mm)	Thickness (mm)	Material (n/V)	Diameter (mm)	Conic	a_4	a_6	a_8	a_{10}	a_{12}
1 (Sphere)	28.621	4.20	1.83481/42.7	29.99	0	0	0	0	0	0
2 (Sphere)	68.136	0.18		28.48	0	0	0	0	0	0
3 (Sphere)	17.772	6.70	1.79952/42.2	23.90	0	0	0	0	0	0
4 (Sphere)	59.525	1.10	1.80518/25.4	20.78	0	0	0	0	0	0
5 (Sphere)	11.427	5.27		16.78	0	0	0	0	0	0
6 (Aper)		6.20		16.24	0	0	0	0	0	0
7 (Sphere)	-16.726	0.90	1.67270/32.1	14.95	0	0	0	0	0	0
8 (Sphere)	-29.829	0.83		15.46	0	0	0	0	0	0
9 (ASphere)	-25.000	2.95	1.53110/55.9	15.52	0	-4.12032e-05	-2.90015e-07	-4.67119e-09	7.90646e-11	-9.28470e-13
10 (ASphere)	-18.373	0.98		18.14	0	-2.41619e-05	-3.29146e-07	1.91098e-10	-9.28593e-13	-2.29193e-13
11 (Sphere)	280.004	4.60	1.73400/51.5	24.43	0	0	0	0	0	0
12 (Sphere)	-34.002	25.67		25.71	0	0	0	0	0	0
Sensor				43.27						

is far below the original, our experiments on DP PSFs (see Fig. 5 in the main paper and Fig. S3) demonstrate that aberrations and phase differences in defocused regions remain clearly visible.

Capturing the real DP PSFs is a straightforward process. A single white pixel is displayed on an OLED screen (iPhone 15 Pro, 460 ppi, 55 um pixel size). We capture the PSFs using a Canon R6 Mark II (6 um pixel size) equipped with an RF50mm lens at object distances ranging from 0.5 m to 2 m. For the camera's native resolution of 6000×4000, the Nyquist-limited spatial resolution is approximately 120 um at 0.5 m and 480 um at 2 m. Given the subsequent downsampling, capturing a single bright pixel on the screen is sufficient to ensure the acquisition of the real PSFs.

The PSFs generated by optical simulations represent linear light intensity distributions, similar to those in the camera's RAW domain. To our knowledge, Canon's RAW-domain dual-pixel images are not publicly available, and currently only RGB-domain dual-pixel images can be obtained. Therefore, the PSFs captured by the camera are pixel value distributions in the RGB domain, transformed from RAW-domain light intensities by the image signal processing (ISP) pipeline. This pipeline involves non-linear and non-invertible operations such as gamma correction and contrast enhancement. Although modern ISP pipelines are highly complex, radiometric calibration remains a common and practical approach to approximately linearize RGB data and recover the underlying light intensity distributions in the RAW domain. As shown in Fig. S4, our linear calibration method consists of: 1. Capture a static scene under varying exposure times; 2. Use the image captured with the shortest exposure time (pixel values within [0, 35]) as a linear reference, scaling it according to exposure time ratios to obtain target values for the other images; 3. Fit the original pixel values of other images to these targets. We adopt a piecewise weighted fitting approach, and the fitted functions are released with our code.

S1.3. Local DP PSF convolution

Existing methods generally use the same PSF kernel to convolve the entire image. Yang *et al*. [6] provides a local convolution operation function based on PyTorch, enabling the use of different PSF kernels for each pixel. However, in our experiments, we use different DP PSF kernels for each pixel. We corrected the kernel flipping error in his function and provided an implementation for using different DP PSF kernels for each pixel, as follows:

Table S2. **Canon RF35mm F/1.8 lens data.** The real isolation lens used for detecting DP pixel structural parameters.

Surface	Radius (mm)	Thickness (mm)	Material (n/V)	Diameter (mm)	Conic	a_4	a_6	a_8	a_{10}	a_{12}
1 (Sphere)	800.000	1.00	1.80810/22.8	27.80	0	0	0	0	0	0
2 (Sphere)	33.296	1.92		25.64	0	0	0	0	0	0
3 (Sphere)	103.801	3.11	2.00100/29.1	25.57	0	0	0	0	0	0
4 (Sphere)	-86.901	4.09		25.07	0	0	0	0	0	0
5 (Sphere)	-47.674	1.30	1.51742/52.4	20.44	0	0	0	0	0	0
6 (Sphere)	17.367	5.73	1.90043/37.4	21.60	0	0	0	0	0	0
7 (Sphere)	777.674	3.72		21.26	0	0	0	0	0	0
8 (Aper)		3.62		20.16	0	0	0	0	0	0
9 (Sphere)	64.497	2.12	1.69680/55.5	19.04	0	0	0	0	0	0
10 (Sphere)	-262.934	3.56		18.69	0	0	0	0	0	0
11 (ASphere)	-35.963	1.30	1.58313/59.4	17.10	0	-4.61997e-05	-9.22837e-08	-4.60687e-10	1.65555e-13	0
12 (Sphere)	-93.550	0.13		17.19	0	0	0	0	0	0
13 (Sphere)	-84.988	6.26	1.88300/40.8	17.29	0	0	0	0	0	0
14 (Sphere)	-12.701	1.00	1.85478/24.8	18.97	0	0	0	0	0	0
15 (Sphere)	135.000	5.27		22.77	0	0	0	0	0	0
16 (Sphere)	800.000	7.35	1.90043/37.4	31.72	0	0	0	0	0	0
17 (Sphere)	-28.799	0.95		33.14	0	0	0	0	0	0
18 (Sphere)	-109.518	2.86	1.69680/55.5	34.06	0	0	0	0	0	0
19 (Sphere)	-53.092	11.79		34.38	0	0	0	0	0	0
20 (Sphere)	-29.766	1.70	1.59270/35.3	33.78	0	0	0	0	0	0
21 (Sphere)	-114.300	11.66		36.27	0	0	0	0	0	0
Sensor				43.27						

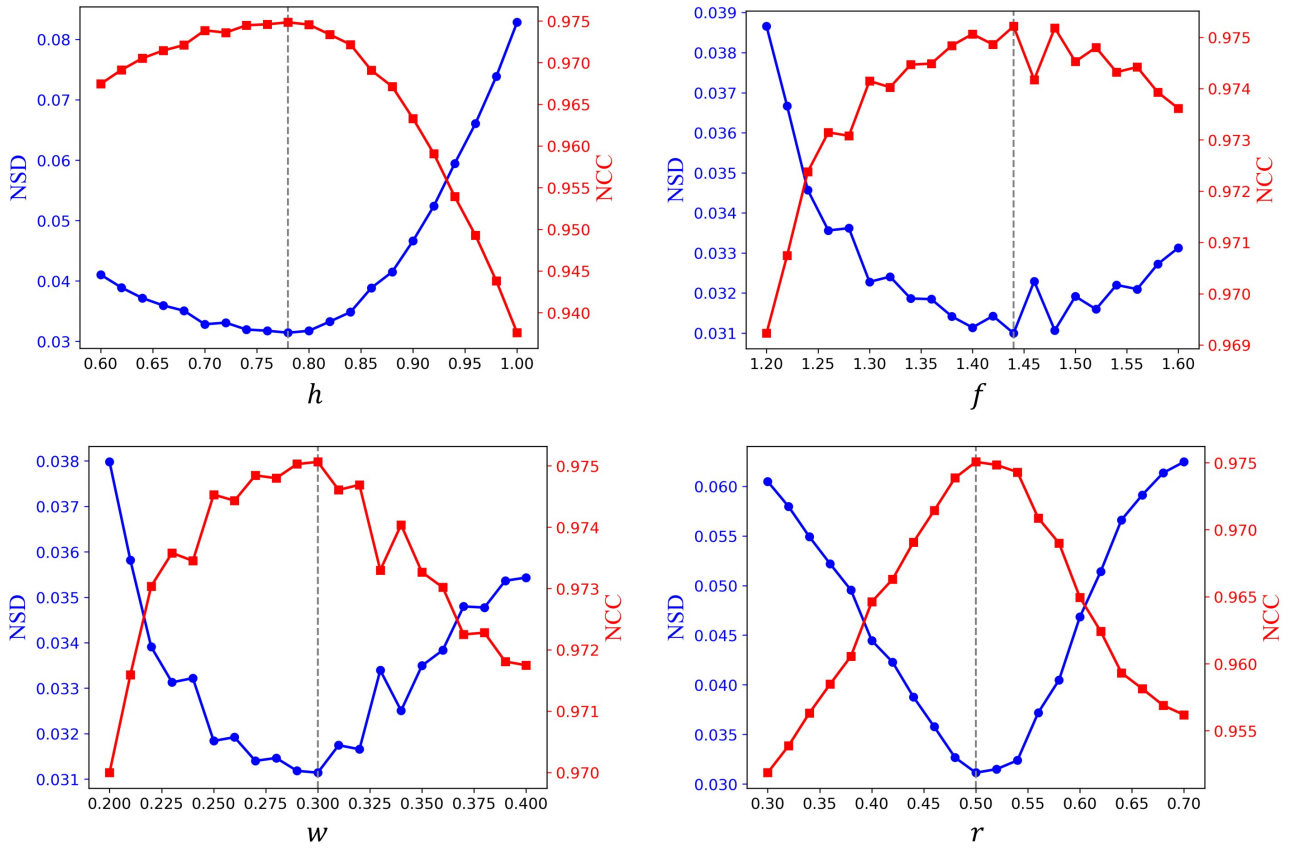

Figure S2. **Error and similarity matching results of DP pixel structural parameters using the isolation lens.** For each subplot, the horizontal axis represents the corresponding structural parameter, where the values are given in multiples of the pixel size. The left vertical axis represents the NSD matching result between real and simulated DP PSFs, while the right vertical axis represents the NCC matching result. To facilitate display, when showing the matching result for any one parameter, we fix the rest of the parameters at their optimal values according to Eq. (S1). Moreover, we use white dashed lines to mark the optimal value for each parameter.

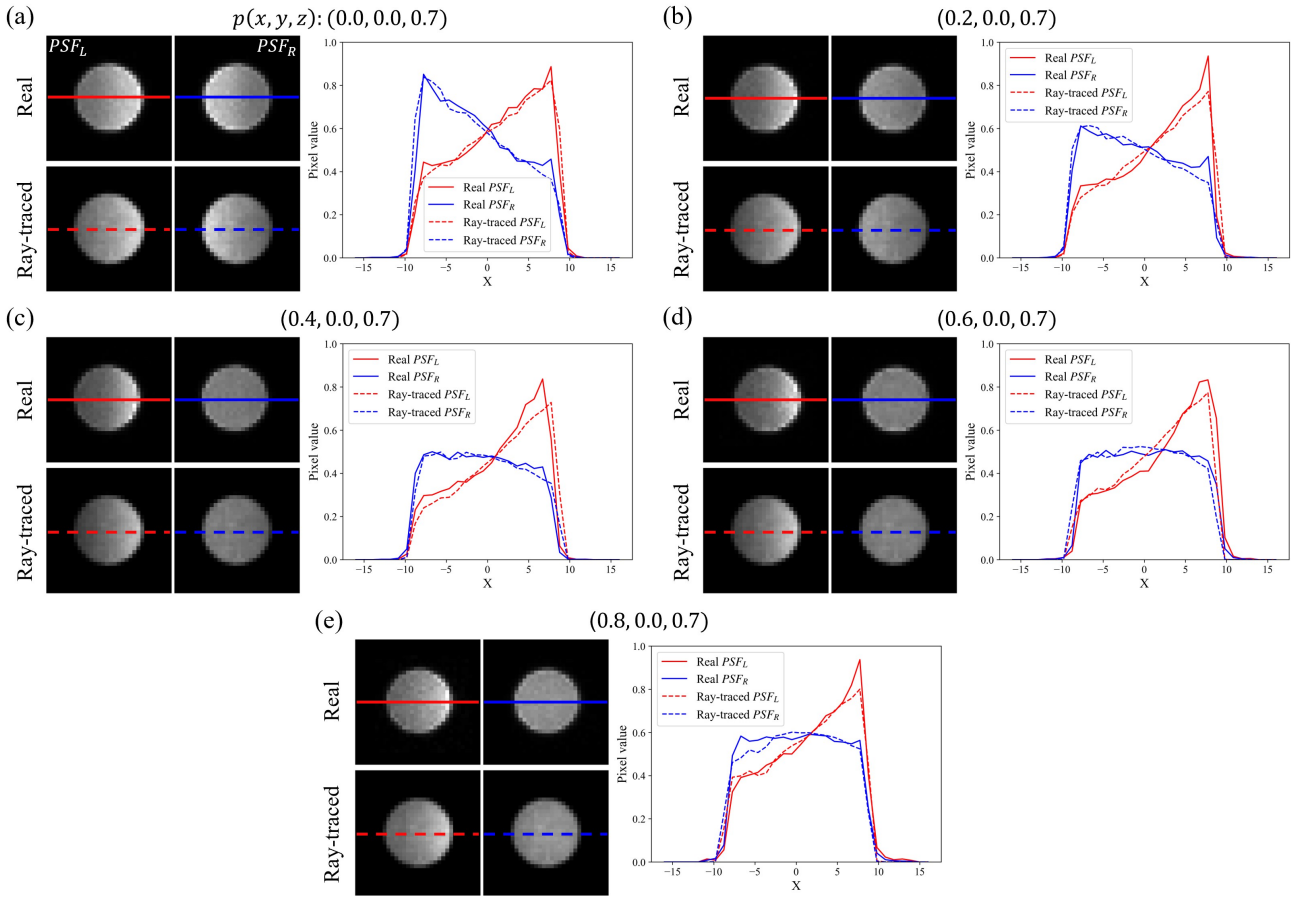

Figure S3. **Qualitative results of DP PSF for real and ray tracing.** We select 5 points within the valid imaging region, and only their x-coordinates increased sequentially, corresponding to (a) - (e). We provide not only the comparison results of real and ray-traced DP PSF (PSF_L on the left, PSF_R on the right) for these 5 points. We also present the pixel distribution curves for the central row. As the object point p moves farther away from the optical axis, the real PSF_L and PSF_R become more phase asymmetric. Our simulated DP PSFs, obtained through ray tracing, align well with the real DP PSFs.

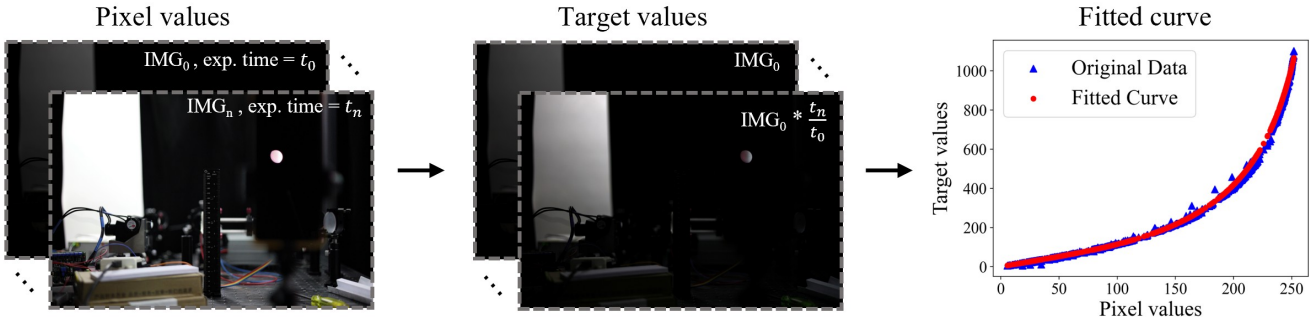

Figure S4. **Overview of our linear calibration method.** The plot shows three components: pixel values from images captured under varying exposure times, target values derived by scaling the shortest exposure image (pixel values within [0, 35]) according to exposure time ratios, and the fitted curve obtained by applying a piecewise weighted fitting approach to align original pixel values with target values. The fitted functions are publicly available in our released code.

```
def local_dp_psf_render(input, dp_psf,
    kernel_size=21):
    """ Render DP image with local DP PSF.
    Use different DP PSFs for different pixels.
    Args:
        input (Tensor): The image to be blurred (
            N, C, H, W).
        dp_psf (Tensor): Per pixel local DP PSFs
            (1, H, W, 2, ks, ks)
        kernel_size (int): Size of the DP PSFs.
            Defaults to 21.
    Returns:
        output (Tensor): Rendered DP image (N, 2*
            C, H, W)
    """

    b,c,h,w = input.shape
    pad = int((kernel_size-1)/2)

    # 1. pad the input with replicated values
    inp_pad = torch.nn.functional.pad(input, pad
        =(pad,pad,pad,pad), mode='replicate')

    # 2. Create a Tensor of varying DP PSF
    kernels = dp_psf.reshape(-1, 2, kernel_size,
        kernel_size)
    kernels_flip = torch.flip(kernels, [-2, -1])
    kernels_rgb = torch.stack(c*[kernels_flip],
        2)

    # 3. Unfold input
    inp_unf = torch.nn.functional.unfold(inp_pad,
         (kernel_size,kernel_size))

    # 4. Multiply kernel with unfolded
    x1 = inp_unf.view(b,c,-1,h*w)
    x2_l = kernels_rgb[:,0,...].view(b, h*w, c,
        -1).permute(0, 2, 3, 1)
    x2_r = kernels_rgb[:,1,...].view(b, h*w, c,
        -1).permute(0, 2, 3, 1)
    y_l = (x1*x2_l).sum(2)
    y_r = (x1*x2_r).sum(2)

    # 5. Fold output
    render_l = torch.nn.functional.fold(y_l,(h,w)
        ,(1,1))
    render_r = torch.nn.functional.fold(y_r,(h,w)
        ,(1,1))

    return torch.cat([render_l,render_r], dim=1)
```

S1.4. DP PSF prediction network

During the training of the DP PSF prediction network, as described in Sec. 5.1 (Implementation details) of the main paper, we set the aperture to F/4, ks to 21, and trained for 100,000 iterations. After training, we evaluated the fitting quality of the DP PSF network.

We present qualitative and quantitative comparison results between our MLP network and recent work [5, 6], As shown in Fig. S5, we evaluate DP PSF at three depths and positions, using ray-traced DP PSF as GT. Their results show significant errors compared to ray-traced ones at large PSF radii. This is due to their training scheme using sum normalization for GT, which is effective for small PSF radii but becomes less accurate for larger ones. In contrast, our network, also using only MLP, can predict accurate results at all depths and positions, even at large PSF radii. The only difference is that we use max normalization for GT DP PSFs during training, and apply sum normalization to the predicted DP PSFs during inference to approximate the uniform intensity distribution of cameras with vignetting compensation. Compared to the DP PSF obtained through ray tracing, the results output by the MLP network are smoother. This is because the number of rays sampled in ray tracing is limited by memory and computation time, with only 4096 rays set.

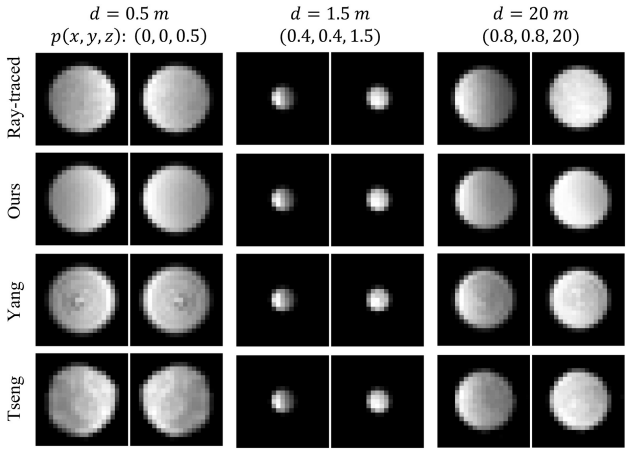

Figure S5. **Qualitative results of DP PSF prediction networks.** Evaluate the ray-traced and network-predicted (ours, Yang *et al*. [6], and Tseng *et al*. [5]) F/4 DP PSFs at three depths (0.5m, 1.5m, 20m) and positions. For small DP PSF radii, all network predictions closely match ray-traced results. For larger radii, the predicted results by [5, 6] show significant deviations, attributed to differences in normalization schemes with ours during training.

Table S3. Quantitative results of DP PSF prediction networks.

Method	L1 error ↓	L2 error ↓	Time (s) ↓
Ray-traced	-	-	801.544
Ours	**6.887e-5**	**6.830e-8**	**0.395**
Yang *et al*. [6]	1.487e-4	1.381e-7	**0.395**
Tseng *et al*. [5]	1.124e-4	7.685e-7	43.96

We present quantitative results in Tab. S3. Specifically, we compare L1 and L2 errors for 50 different depths and positions, as well as the time cost for predicting a DP PSF map for all pixels (a total of 512×768) in the depth map. Our network outperforms those developed by Yang *et al*. [6] and Tseng *et al*. [5] in accuracy, and its low time cost enables high-speed image rendering. In contrast, the network proposed by Tseng *et al*. [5] incurs high time costs due to its inclusion of convolutional layers.

Overall, rendering a DP image using a DP PSF map is very fast (with a fixed cost of 0.2 seconds). However, generating these DP PSFs through ray tracing is time-consuming (801.544 seconds). We use MLP to save time, achieving high speed (0.395 seconds), which enables real-time rendering of DP images for DP data-driven tasks during each iteration.

S2. Depth-from-Dual-Pixel Details

S2.1. Adjustment of cost volume

In the field of DfDP, we do not pursue the optimal DfDP network structure. Instead, we select [2] as the DfDP model and make reasonable adjustments to its cost volume step to accommodate the DP data.

Existing binocular disparity matching algorithms typically only consider unidirectional disparity shifts when stacking left and right image features to form the cost volume. This is because in binocular images, the disparity formed by points at any depth is always in the same direction. However, in DP images, the disparity direction formed by object points before and after the focal distance is opposite. Therefore, as shown in Fig. 4(d) (main paper), we add reverse disparity while stacking the original disparity. Our implementation is as follows:

```
def get_dp_cost_volume(x, y, d_max=20):
    """ Get DP image cost volume
    Stack original disparity and add reverse
        disparity.
    Args:
        x (Tensor): Left DP image feature (B, C,
            H, W).
        y (Tensor): Right DP image feature (B, C,
             H, W).
        d_max (int): Max displacement. Defaults
            to 20.
    Returns:
        cost (Tensor): Cost volume of DP image
            features (B, 2*C, d_max, H, W)
    """

    B, C, H, W = x.size()
    cost = torch.zeros(B, C*2, d_max, H, W).
        type_as(x)
    for i in range(d_max):
        d = i-d_max//2
        if d < 0:
            cost[:, :C, i, :, :d]=x[:, :, :, :d]
            cost[:, C:, i, :, :d]=y[:, :, :, -d:]
        elif d == 0:
            cost[:, :C, i, :, :]=x
            cost[:, C:, i, :, :]=y
        if d > 0:
            cost[:, :C, i, :, d:]=x[:, :, :, d:]
            cost[:, C:, i, :, d:]=y[:, :, :, :-d]

    return cost
```

S2.2. A new real-world test set DP119

We used the Canon RF50mm lens and Canon R6 Mark II camera to capture a new dataset, setting the aperture to F/4 and focusing at 1 m. Each scene provides real DP-depth paired data. Finally, we collected the DP119 dataset, which includes 45 planar scenes, 44 box scenes, and 30 casual scenes, totaling 119 scenes.

For the planar scenes, we captured richly textured posters images from 0.5 m to 2 m perpendicular to the camera's optical axis, with denser sampling at shallower depths. The dataset includes five distinct scenes at nine different distances (0.5 m, 0.6 m, 0.7 m, 0.8 m, 0.9 m, 1.0 m, 1.2 m, 1.5 m, 2 m). Real DP images were captured at apertures F/4 and F/20, adjusting ISO to match light intensity. F/20 images were used as AiF images, with their depth value copied into the depth map. Unlike box and casual scenes, planar scenes have simple depth structures, which avoid common issues such as holes in LiDAR depth maps and misalignment between depth and RGB images. These scenes are useful for evaluating both depth estimation models and the realism of DP simulators. As shown in Fig. S6(a), we present a sample set of planar scenes.

For the box scenes, we covered all the boxes and backgrounds with posters. The boxes were randomly placed within a range of 0.5 m to 2 m, and each time we took a shot, we significantly adjusted the number, texture, and placement of the boxes, deliberately creating situations of overlap, tilt, and occlusion. Depth was captured using the LiDAR of an iPhone 15 Pro. Both planar and box scenes are richly textured, aiding the model in utilizing aberration and phase difference cues for depth estimation. Textureless areas, even when defocused, do not introduce any aberrations or phase differences, which would interfere with depth estimation. Therefore, planar and box scenes are ideal for evaluating whether the DP simulator addresses the domain gap between simulated and real DP images. As shown in Fig. S6(b), we present samples of box scenes.

For the casual scenes, we directly captured ordinary real-world scenes, intentionally controlling the depth within 0.5 m to 10 m, as scenes beyond 10m would yield unreliable results from the depth sensor. Among the 30 casual scenes, 20 indoor scenes were captured with depth obtained using a binocular structured-light camera (Orbbec Gemini2). For the 10 outdoor scenes, since the binocular structured-light camera performs poorly outdoors, we used the LiDAR of an iPhone 15 Pro to capture depth. As shown in Fig. S6(c), we present samples of casual scenes.

S2.3. More DfDP results on the DP119 test set

Fig. S7 presents the depth estimation results of various simulators on different scenes in the DP119 dataset. From planar scenes 1 and 2, we observe that due to increased aberrations and asymmetric phase differences at the edge regions,

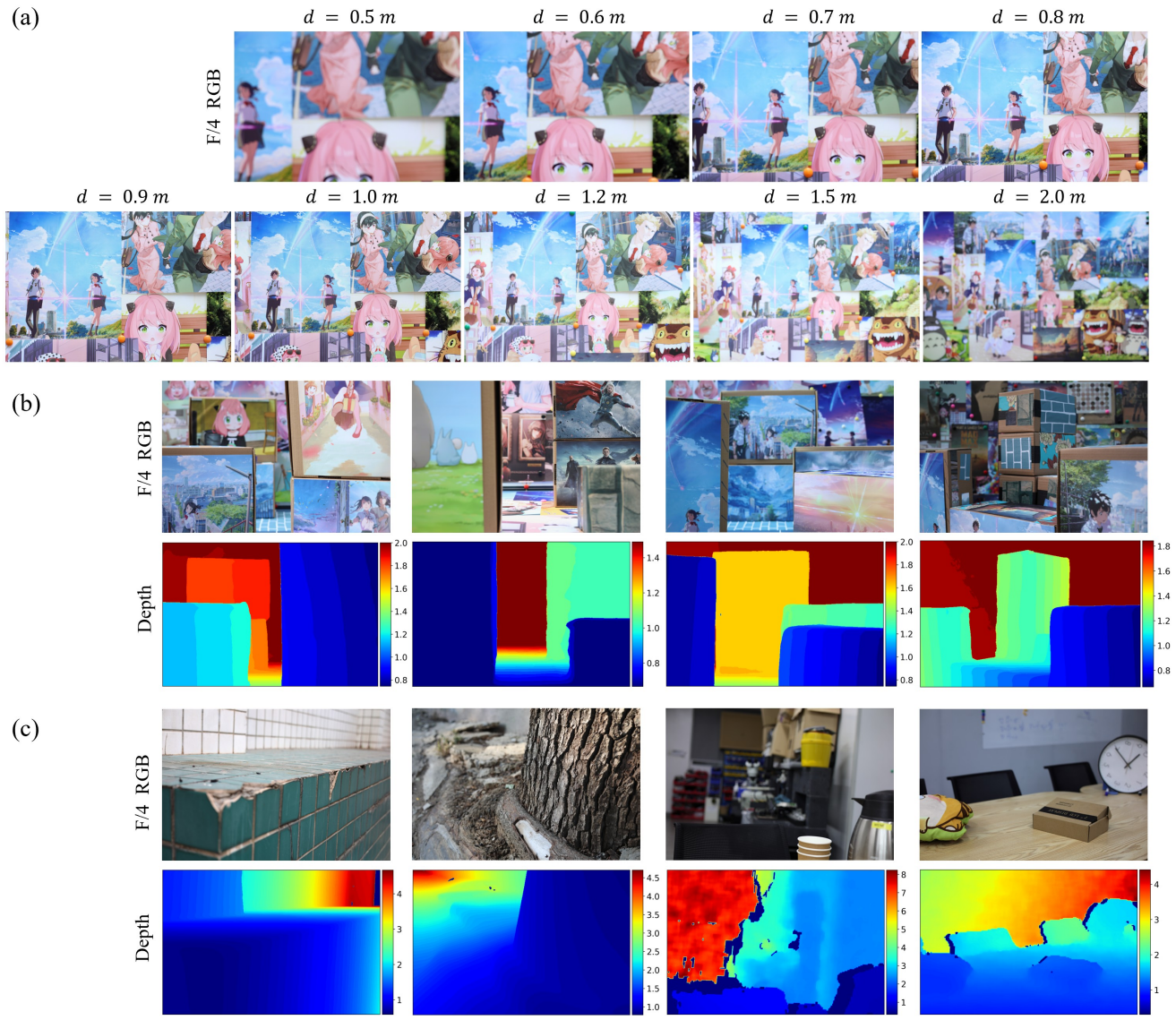

Figure S6. **Examples of scenes in the DP119 dataset.** Each depth map is decorated with a color bar in meters. (a) Planar scene samples. The d is the planar depth value at the time of capture. (b) Box scene samples. (c) Casual scene samples.

all models, including ours, exhibit significant relative positional errors in both central and edge regions. However, our model provides more accurate absolute depth estimates, whereas other models tend to bias toward the focus distance (1 m). The depth estimation results of L2R [1] and Modeling [4] are similar, as their DP PSF peak positions are spatially close. A similar trend is observed in box scenes 1 and 2, where other models show a strong bias toward the focal distance (1 m) and exhibit large relative positional errors in both central and edge regions. In contrast, our model maintains superior performance.

In casual scenes 1 and 2, the textures are rich, and the scenes are simple. Models can predict depth not only using aberrations and phase differences but also by leveraging additional structural cues. As a result, the relative positional errors in the center and edge regions were alleviated for all models in these two scenes. However, the absolute depth estimates from other models still show a clear bias toward the focus distance (1 m). Moreover, in casual scenes 3 and 4, large textureless areas appear, missing aberrations and phase differences. Depth estimation models can only infer the depth of these textureless areas along structural cues, leading to significant relative positional errors in textureless regions for all models.

Overall, our model outperforms others in both relative and absolute position accuracy. This demonstrates the high realism of our DP simulator and its ability to bridge the domain gap between simulated and real DP images.

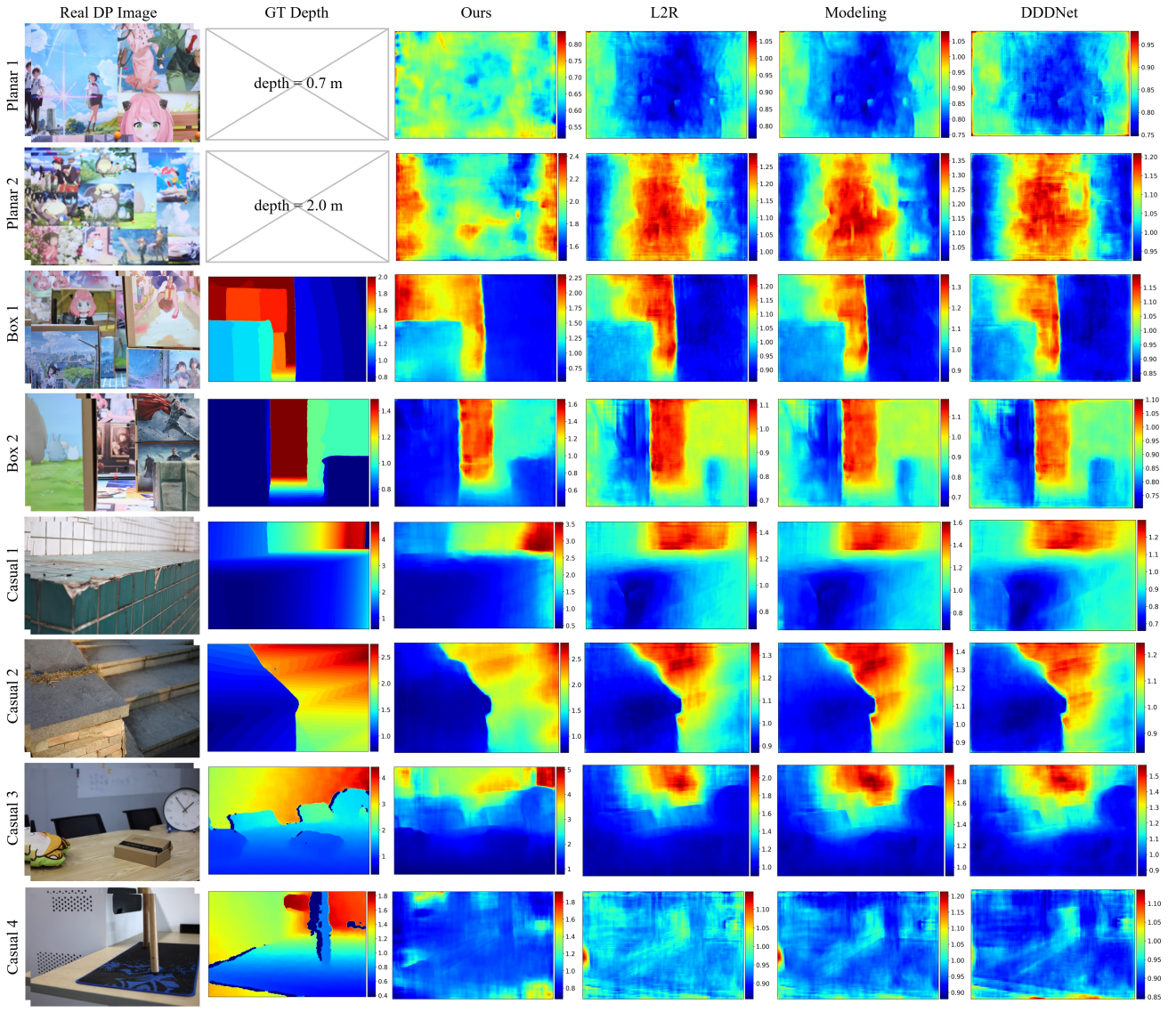

Figure S7. **The depth estimation results from different simulators on the DP119 test set.** Evaluate DfDP models with L2R [1], Modeling [4], DDDNet [3] and our Sdirt on various DP119 dataset scenes. Each result image is decorated with a color bar in meters. Their depth estimation results show partial accuracy in relative positional relationships but large absolute positional errors. Our depth estimation results, however, demonstrate accurate in both relative and absolute positions with minimal errors. Furthermore, textureless areas lead to degradation in all models. (Best viewed in color and enlarge on screen.)

References

[1] 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*, pages 2289–2298, 2021. 7, 8

[2] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *Advances in neural information processing systems*, 33:22158–22169, 2020. 6

[3] Liyuan Pan, Shah Chowdhury, Richard Hartley, Miaomiao Liu, Hongguang Zhang, and Hongdong Li. Dual pixel exploration: Simultaneous depth estimation and image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2021. 8

[4] Abhijith Punnappurath, Abdullah Abuolaim, Mahmoud Afifi, and Michael S Brown. Modeling defocus-disparity in dual-pixel sensors. In *2020 IEEE International Conference*

on Computational Photography (ICCP), pages 1–12. IEEE, 2020. 7, 8

[5] Ethan Tseng, Ali Mosleh, Fahim Mannan, Karl St-Arnaud, Avinash Sharma, Yifan Peng, Alexander Braun, Derek Nowrouzezahrai, Jean-Francois Lalonde, and Felix Heide. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Transactions on Graphics (TOG)*, 40(2):1–19, 2021. 5

[6] Xinge Yang, Qiang Fu, Mohamed Elhoseiny, and Wolfgang Heidrich. Aberration-aware depth-from-focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2, 5