# Simulating Dual-Pixel Images From Ray Tracing For Depth Estimation

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

## Abstract

*Many studies utilize dual-pixel (DP) sensor phase characteristics for various applications, such as depth estimation and deblurring. However, since the DP image features are entirely determined by the camera hardware, DP-depth paired datasets are very scarce, especially when performing depth estimation on customized cameras. To overcome this, studies simulate DP images using ideal optical system models. However, these simulations often violate real optical propagation laws, leading to poor generalization to real DP data. To address this, we investigate the domain gap between simulated and real DP data, and propose solutions using the Simulating DP images from ray tracing (Sdirt) scheme. The Sdirt generates realistic DP images via ray tracing and integrates them into the depth estimation training pipeline. Experimental results show that models trained with Sdirt-simulated images generalize better to real DP data. The code and collected datasets will be available on https://github.com/LinYark/Sdirt.*

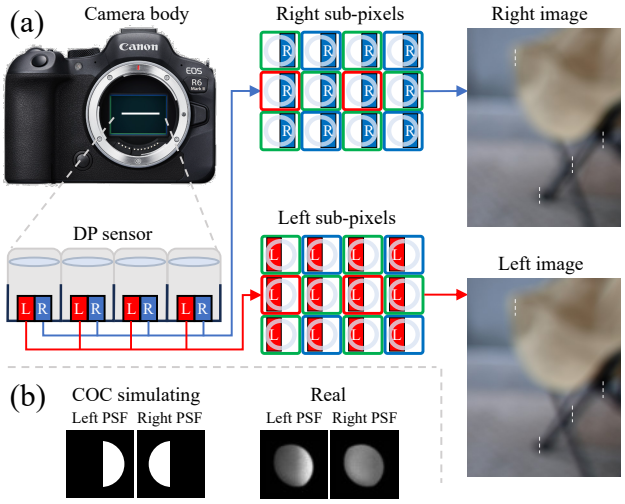


Figure 1. (a) Imaging process of a DP camera. We illustrate the slight shifts in the left and right DP images caused by phase splitting with white dashed lines. (b) Example comparison between real and COC simulated DP PSF. There exists a significant difference between the two.

## 1. Introduction

The DP sensor [15, 29] is designed to split each pixel into a left and a right sub-pixel, utilizing microlenses and sub-pixels to achieve phase splitting. This allows capturing a pair of images in one shot, known as DP images, as shown in Fig. 1(a). DP images can not only be used to autofocus [7, 11, 31] but also enhance the performance in applications such as deblurring [4, 17, 41] and depth estimation [9, 13, 23]. In the field of Depth-from-Dual-Pixel (DfDP) depth estimation, DP-depth paired datasets are scarce due to camera hardware limitations. To address the scarcity of DP-depth paired datasets, much research [2, 16, 18, 23, 26, 37] has focused on simulating DP images using RGBD datasets, aiming to provide a more flexible and accessible data source.

The key to simulating the DP image through RGBD datasets lies in the DP Point-Spread-Function (PSF). In recent years, many model-based and calibration-based DP PSF simulators have emerged. Calibration-based simulators [16, 18, 37] spent considerable time calibrating real cameras to match the DP PSF but faced issues such as interpolation errors from discrete calibration and neglect of lens and DP sensor parameters, limiting model transferability. Model-based simulators [2, 23, 26] used ideal optical models or aberration analysis models to directly calculate the DP PSF, thus bypassing the calibration step. However, these models perform poorly on real DP images due to ignoring lens aberration and DP phase splitting characteristics. As shown in Fig. 1(b), the ideal thin lens model's simulated Circle-of-Confusion (COC) DP PSF has a significant domain gap with the real one, affecting depth estimation.

To address the domain gap between real DP images and DP images simulated by model-based simulators, we propose the Sdirt scheme. Sdirt consists of a ray-traced DP PSF simulator and a pixel-wise DP image rendering module, which can accurately generate DP images with aberration and phase characteristics. Specifically, for a fixed focus lens and DP sensor system with known parameters, we use ray tracing [35, 39, 40] to calculate the spatially varying DP

PSF and train a multilayer perceptron (MLP) to predict it. Subsequently, we convolve the per-pixel DP PSF with the All-in-Foucs (AiF) image to simulate the DP images captured by a real camera. After training a DfDP model with DP images simulated by Sdirt, the model can accurately estimate depth using optical aberration and phase information, which enhances its ability to generalize to real DP images.

We conduct extensive experiments to validate the effectiveness of Sdirt. The experimental results indicate that, compared to other model-based simulators, the DP PSFs and DP images generated by Sdirt are closer to real images, and the DfDP model trained on Sdirt achieves better generalization. In summary, our contributions are four-fold:

- We propose a ray-traced DP PSF simulator that calculates the spatially varying DP PSF, addressing the domain gap between simulated and real DP PSF caused by lens aberrations and sensor phase splitting.
- We propose a pixel-wise DP image rendering module that uses an MLP to predict the DP PSF for each pixel, narrowing the gap between simulated and real DP images.
- Experimental results of depth estimation demonstrate that the DfDP model trained on Sdirt generalizes better to real DP images.
- We collected a DP-depth paired real test set, with an open lens structure and fixed focus, for use in subsequent research.

## 2. Related Work

### 2.1. DP Datasets

Most professional and mobile cameras have DP sensors, but only Canon cameras [1, 25, 26] and Google Pixel phones [9, 34, 43] provide DP data. In the field of depth estimation, supervised tasks rely on DP-depth paired datasets [1, 9, 26, 43]. However, when the lens or sensor is changed, or the focal length and aperture settings are adjusted, the point spread function (PSF) is altered, requiring significant time and effort to re-collect DP-depth paired data. Consequently, many researchers are focusing on simulating DP-depth paired datasets using RGBD datasets.

### 2.2. DP PSF simulator

Simulating DP images from RGBD datasets relies on DP PSF, with recent progress in model-based [16, 18, 37] and calibration-based [2, 23, 26] DP PSF simulators. In calibration-based simulators, Xin *et al*. [37] acquired full-space DP PSF by sampling discrete points in space and interpolating, while Li *et al*. [18] acquired it through size scaling using the COC model. Furthermore, Li *et al*. [16] generated DP PSF through a U-Net network, which requires a large amount of DP-depth paired data. Calibration-based simulators are time-consuming and prone to interpolation and scaling errors, while model-based simulators circumvent these issues. Pan *et al*. [23] simplified the optical system as an ideal thin lens and used a symmetrically divided rectangular aperture to calculate DP PSF. Punnappurath *et al*. [26] fitted DP PSF as the left-right flip symmetric kernel depletion shape. Abuolaim *et al*. [2] used aberration analysis models to calculate DP PSF, but the resulting DP PSF remained symmetric. These studies oversimplify optical propagation, neglecting aberrations and phase splitting, causing a large gap between simulated and real DP data.

### 2.3. Applications driven by DP data

In recent years, DP data-driven applications such as depth estimation [9, 10, 13, 14, 23, 24, 43], deblurring [3, 4, 17, 27, 38, 41, 42], and refocus [5, 7, 11, 32] have undergone rapid development. Additionally, Kang *et al*. [13] utilized DP data for facial scanning, similar to depth estimation tasks. Wadhwa *et al*. [34] employed DP data for synthetic shallow depth-of-field imaging, a popular application in mobile photography recently. Shi *et al*. [28] integrated DP hardware with structured diffractive optical elements for complex image encoding and high-precision multimodal reconstruction. In addition, DP data can also be used for disparity estimation [22, 36], rain [18], and reflections [25] removal. It is foreseeable that DP data-driven applications will continue to emerge in various new fields.

## 3. Method of Sdirt

Training the DfDP network requires DP images as input. As shown in Fig. 2(c), the module achieves pixel-wise rendering of DP images by convolving the DP PSF corresponding to each point in the depth map with AiF RGB image. The DP PSF for each point is obtained through inference using a pre-trained MLP network. As illustrated in Fig. 2(b), the offline training process of the MLP network uses ray-traced DP PSF (Fig. 2(a)) as Ground Truth (GT).

### 3.1. Ray-traced DP PSF simulator

As shown in Fig. 2(a), lens ray tracing can accurately obtain the landing point on the sensor surface [8, 35, 39, 40]. We construct a ray set $A$ containing $n$ rays, with all rays originating from a spatial object point $p$. Then, the image of the aperture stop in the object space of the lens is regarded as the entrance pupil, and $n$ points are densely sampled on this entrance pupil plane. The initial direction vector of each ray in $A$ is set to point from $p$ to the corresponding sampling point of the entrance pupil sequentially. As $A$ traverses each surface of the lens, its position and direction strictly follow the unique optical characteristics of the mirror surface, undergoing precise refraction according to Snell's law. After a series of refractions through the lens, $A$ finally lands on the sensor plane. We denote $A$'s landing point on the sensor as $O$ and its ray direction as $D$.

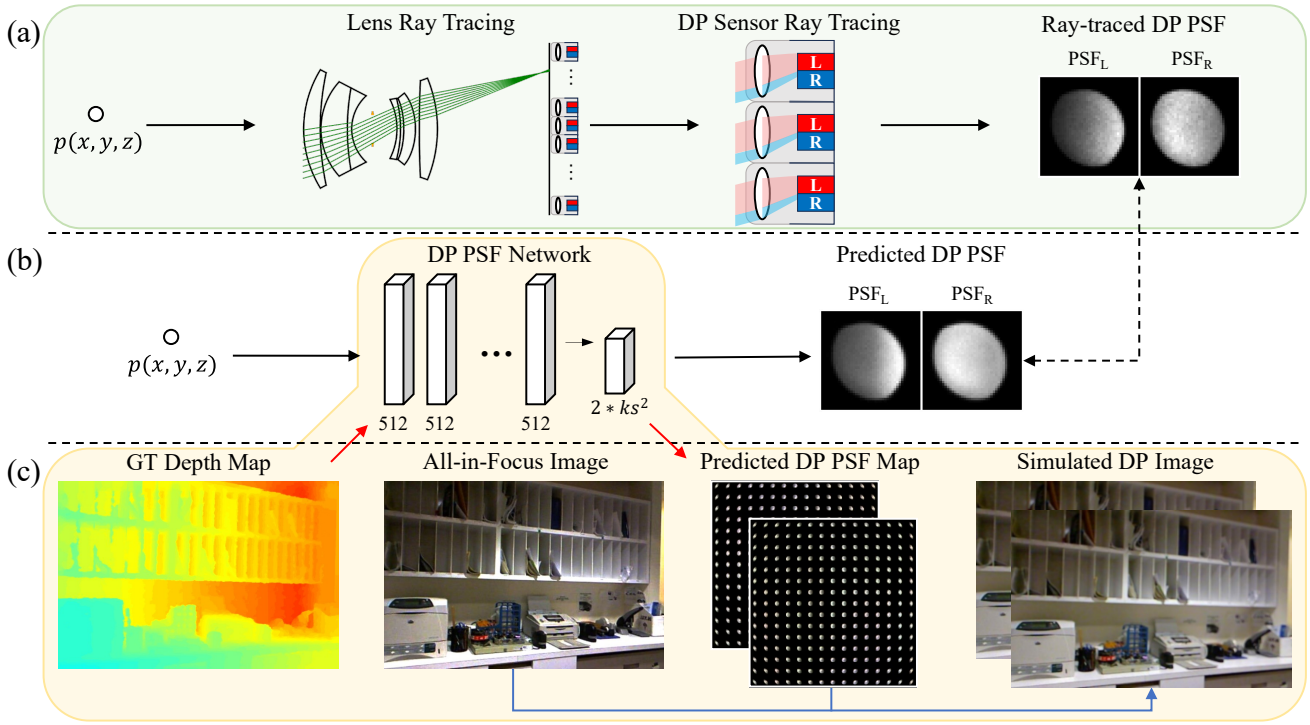


Figure 2. **Simulating dual pixel images from ray tracing pipeline.** (a) Ray-traced DP PSF simulator. Calculates spatially varying DP PSF for lens and DP sensor through ray tracing. (b) DP PSF prediction network. Trains an MLP network to predict DP PSF, using ray-traced DP PSF as ground truth. (c) Pixel-wise DP image rendering module. The network predicts the DP PSFs for all points in the depth map (red pass). Then, the DP PSF is convolved with the AiF RGB image to render the simulated DP image (blue pass).

DP sensor ray tracing is limited by camera manufacturers' non-disclosure of microlens and sub-pixel structure. As a result, we simplify the DP composite pixel structure (Fig. 3(a)) based on past research [15, 29]. We model the microlens as a thin lens with radius $r$ and focal length $f$, setting $h$ for distance between the sub-pixel and the microlens, $w$ for sub-pixel width, and $ps$ for DP composite pixel size.

The DP PSF can be obtained by separately calculating the cumulative integration of rays for each sub-pixel. We analyze which sub-pixel the kth ray $A_k$ in the ray set $A$ ultimately enters. Through lens ray tracing, we know the landing point $O_k(x_k, y_k, z_k)$ and direction $D_k(x'_k, y'_k, z'_k)$ of the kth ray $A_k$ on the sensor surface. Based on the landing point $O_k$, we can easily calculate that $A_k$ lies within the DP composite pixel $(i, j)$, having surface center coordinates $(x_i, y_i)$. For $O_k$, the situations differ depending on whether it is within or outside the microlens:

**When $O_k$ is within the microlens,** as shown in Fig. 3(b), $A_k$ is refracted by the microlens and then enters a sub-pixel. According to geometric optics theory, $A_k$ and the principal ray incident on the thin lens at the same angle converge on the focal plane of the thin lens. We only need to focus on a few boundary lines $x_{L1}, x_{M1}, x_{R1}$ to determine which sub-pixel $A_k$ currently enters. If $x_k$ is in the interval $[x_{L1}, x_{M1}]$, it enters the left sub-pixel; if in the interval $[x_{M1}, x_{R1}]$, it enters the right sub-pixel; otherwise, it is considered a missing ray. The calculation methods for the boundary lines are as follows:

$$\begin{aligned} x_{L1} &= x_i + w - (f * \tan\theta - w) * h/(f - h) \\ x_{M1} &= x_i - (f * \tan\theta) * h/(f - h) \\ x_{R1} &= x_i - w - (f * \tan\theta + w) * h/(f - h) \end{aligned} \tag{1}$$

Where $\tan\theta$ is the tangent angle at which $A_k$ landing on the sensor surface, equivalent to $x'_k/z'_k$.

**When $O_k$ is outside the microlens,** as shown in Fig. 3(c), $A_k$ enters the sub-pixel directly without refraction. Similarly, we only need to focus on a few boundary lines $x_{L2}, x_{M2}, x_{R2}$ to determine which sub-pixel $A_k$ currently enters:

$$\begin{aligned} x_{L2} &= x_i + w - h * \tan\theta \\ x_{M2} &= x_i - h * \tan\theta \\ x_{R2} &= x_i - w - h * \tan\theta \end{aligned} \tag{2}$$

The left PSF $PSF_L$ can be calculated by integrating the set of rays $A$ emitted from point $p$ over all left sub-pixels:

$$PSF_L(i, j) = \sum_{k=1}^{n} A_k * \delta_{L,k}(i, j) \tag{3}$$

where $\delta_{L,k}(i, j)$ represents the energy distribution of ray $A_k$ in the left sub-pixel at DP composite pixel $(i, j)$. The ray energy distribution is assumed to be a unit impulse, which

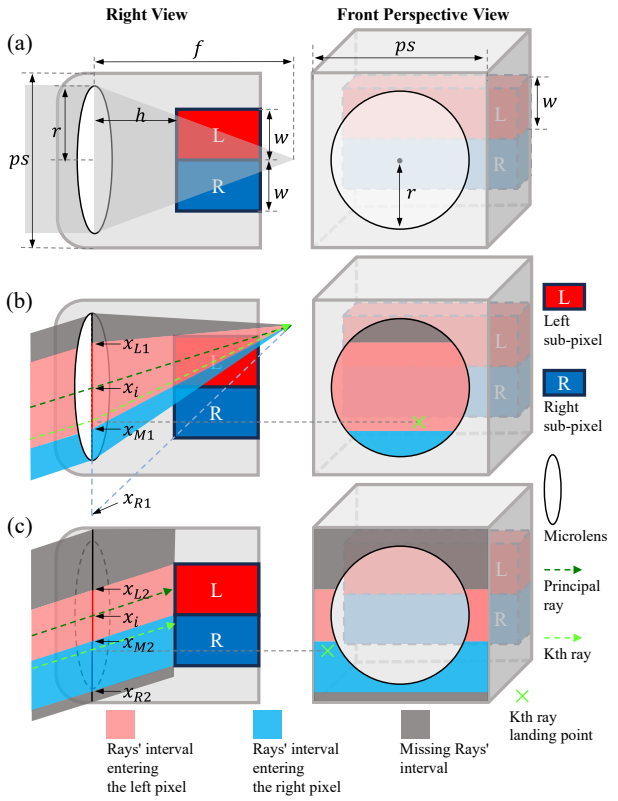


Figure 3. (a) DP composite pixel structure. The left side is a right view, and the right side is a front perspective view. (b) On hitting the microlens, the kth ray lands on left/right sub-pixel in red/blue interval, else missed. (c) Without hitting the microlens, the kth ray lands on left/right sub-pixel in red/blue interval, else missed.

equals zero unless it eventually enters the left sub-pixel of DP composite pixel $(i, j)$. By following the same calculation steps as for $PSF_L$, $PSF_R$ can also be computed.

### 3.2. Pixel-wise DP image rendering module

As shown in Fig. 2(b), we reduce computational costs of ray-traced DP PSF by training an MLP network to predict it, inspired by [33, 39]. The network input is the normalized coordinate $p$ within the camera's valid imaging region (Fig. 4(a)), which is a frustum defined by field of view, sensor size, and the set minimum and maximum depths. We normalize the $(x, y)$ coordinates to $[-1, 1]$. After fixing the focus distance, the kernel size $ks$ of the DP PSF is set to a size such that the DP PSF for any point within the valid imaging region can be fully displayed. Finally, we configure the network structure with 5 hidden layers, each containing 512 neurons, and an output layer with $2 * ks^2$ neurons. We use the L2 function as the loss:

$$Loss = L_2(\hat{PSF_L}, PSF_L) + L_2(\hat{PSF_R}, PSF_R) \quad (4)$$

During training, we use ray-traced DP PSFs as GT and apply max normalization to avoid challenging samples. During inference, we apply sum normalization to the predicted DP PSFs to mimic the vignetting correction in the camera image signal processor.

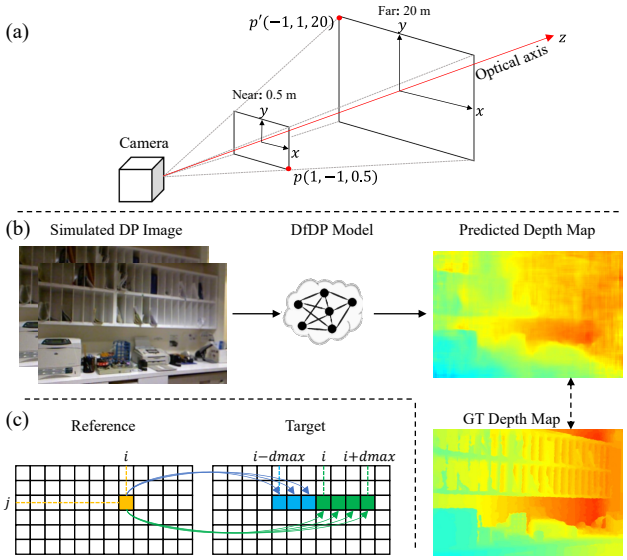


Figure 4. (a) The valid imaging region is a frustum, and we normalize the $(x, y)$ coordinates to $[-1, 1]$. (b) The DfDP model takes DP image as input to predict the detph map. (c) During cost volume generation in [6], we stack original disparity (green arrows) and add reverse disparity (blue arrows), with $dmax$ as max displacement.

Due to the lack of shift-invariance of DP PSF, we perform local convolution of each pixel's DP PSF with the AiF image to simulate the DP image captured by a camera. The RGBD dataset [21, 30] provides the paired RGB image $I_{RGB}$ and depth map $I_D$. As shown in Fig. 2(c), we treat each pixel in $I_D$ as a spatial object point and use the trained MLP network to predict its DP PSF. Using $I_{RGB}$ as the AiF image, we perform pixel-wise local convolution [39] to render DP image containing aberration and phase information.

## 4. DfDP Model Based on Sdirt

As shown in Fig. 4(b), the input of the DfDP model is provided by the pixel-wise DP image rendering module of the Sdirt in real-time. In each training iteration, Sdirt feeds the DfDP model with simulated DP images of shape (B, C, H, W). Here, B is the batch size, C is the number channels, and H and W are the DP image height and width. The model outputs a predicted depth map of shape (B, 1, H, W). We use the L1 function as the loss:

$$L = L_1(\hat{I_D}, I_D) \quad (5)$$

We select [6] as the DfDP model and make reasonable adjustments to its cost volume step to accommodate the DP data. In binocular images, disparity formed by points at different depths is always in the same direction, thus the cost volume generation process in [6] is also unidirectional, as

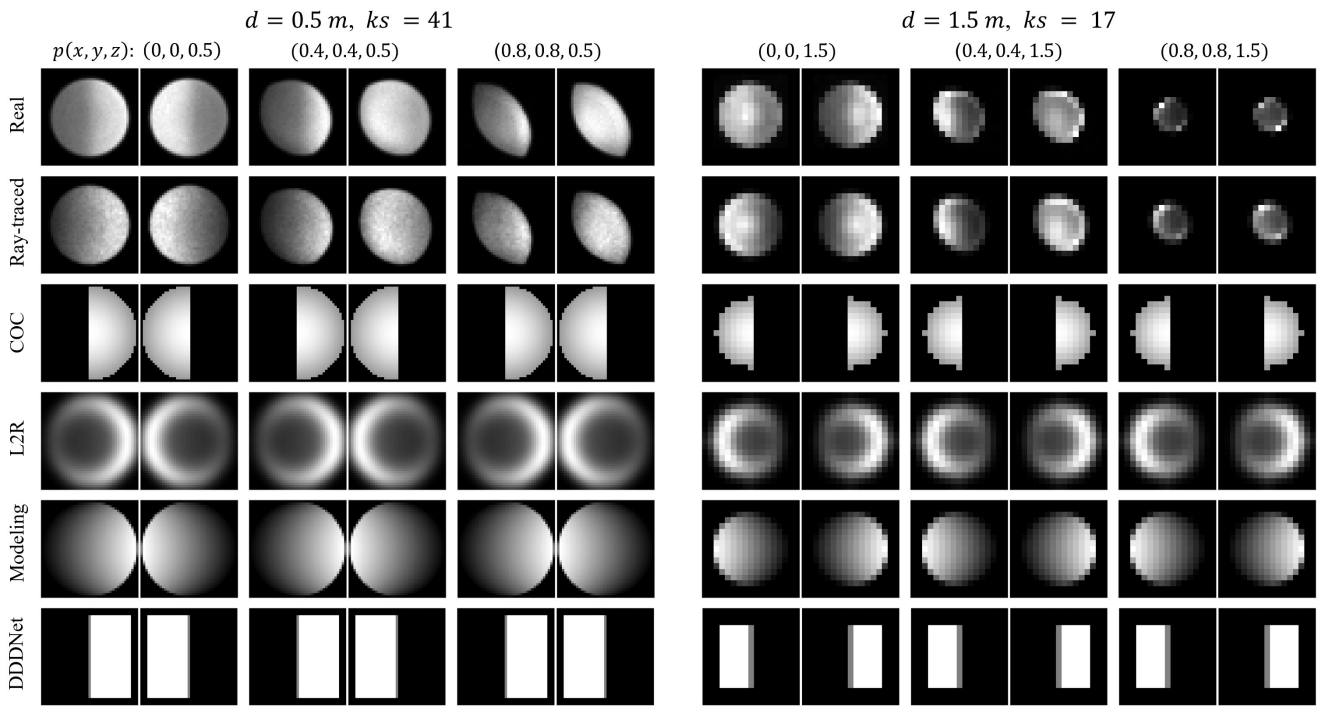


Figure 5. **DP PSF simulator representation result.** Evaluate the real and simulated (ours, COC, L2R [2], Modeling [26] and DDDNet [23] ) F/1.8 DP PSFs at two depths (0.5m, 1.5m) and three different positions. As spatial object point $p$ moves further from the optical axis, the real $PSF_L$ and $PSF_R$ become more asymmetric, and aberrations increase. Their simulators neglect aberrations and DP phase splitting, causing a large gap between simulated and real DP PSF. Only our ray-traced simulator predicts realistic results at all depths and positions.

shown by the green connecting arrows Fig. 4(c). However, in DP images, disparity formed by object points before and after the focus distance is opposite in direction. Therefore, we add blue connecting arrows to extend the cost volume generation process, in order to enhance the DfDP model's ability to capture reverse disparity.

# 5. Sdirt Representation Result

## 5.1. Implementation details

The Sdirt scheme employs a Canon RF50mm lens and a R6 Mark II camerabody as the our real camera. The simulated DP sensor has dimensions of $24mm \times 36mm$ and a resolution of $512 \times 768$. We set the focus distance of both the simulated and real cameras to 1 meter, the valid imaging range to 0.5 to 20 meters, the F-number to F/4, and the ray set $A$ to containing 4096 rays. When training the MLP, the kernel size $ks$ to 21 to fully display the DP PSF of any point within the valid imaging region. We train the network for 100,000 iterations, selecting 128 random points within the valid imaging region during each iteration. We conducted our experiments on a 12700K CPU and a 3090 GPU.

Following [39], we assume chromatic aberration is well corrected compared to other aberrations and defocus effects, allowing us to use a 550 nm wavelength to reduce ray tracing costs. For details on obtaining the structure of the DP composite pixel, please refer to the Supplementary.

## 5.2. Comparison methods

For comparison, we select all existing model-based DP PSF simulators: Modeling, L2R [2], DDDNet [23], and added a COC simulator. Although [24] is a more recent work, its simulator fully reuses DDDNet [23], so we do not include it for comparison. Calibration-based simulators and constraint-training methods are data-driven and require real DP-depth paired data, making them fundamentally different from model-based simulators, so no comparison is made. We directly used the source code of these DP PSF simulators [2, 26], and replicated the simulator [23] that only provided executable files. To ensure fairness, we set the focus distance, F-number, and sensor size in their simulator to be consistent with ours.

## 5.3. Evaluation on DP PSF simulator

To facilitate the observation of aberration and phase information, we temporarily set the F-numbers of both the simulated and real cameras to F/1.8. We capture a screen displaying a bright spot to obtain the real DP PSF.

As shown in Fig. 5, we evaluate the DP PSF ($PSF_L$ on left, $PSF_R$ on right) of spatial object points $p$ at various depths and positions within the valid imaging region. Observing the real DP PSFs, it can be found that defocus

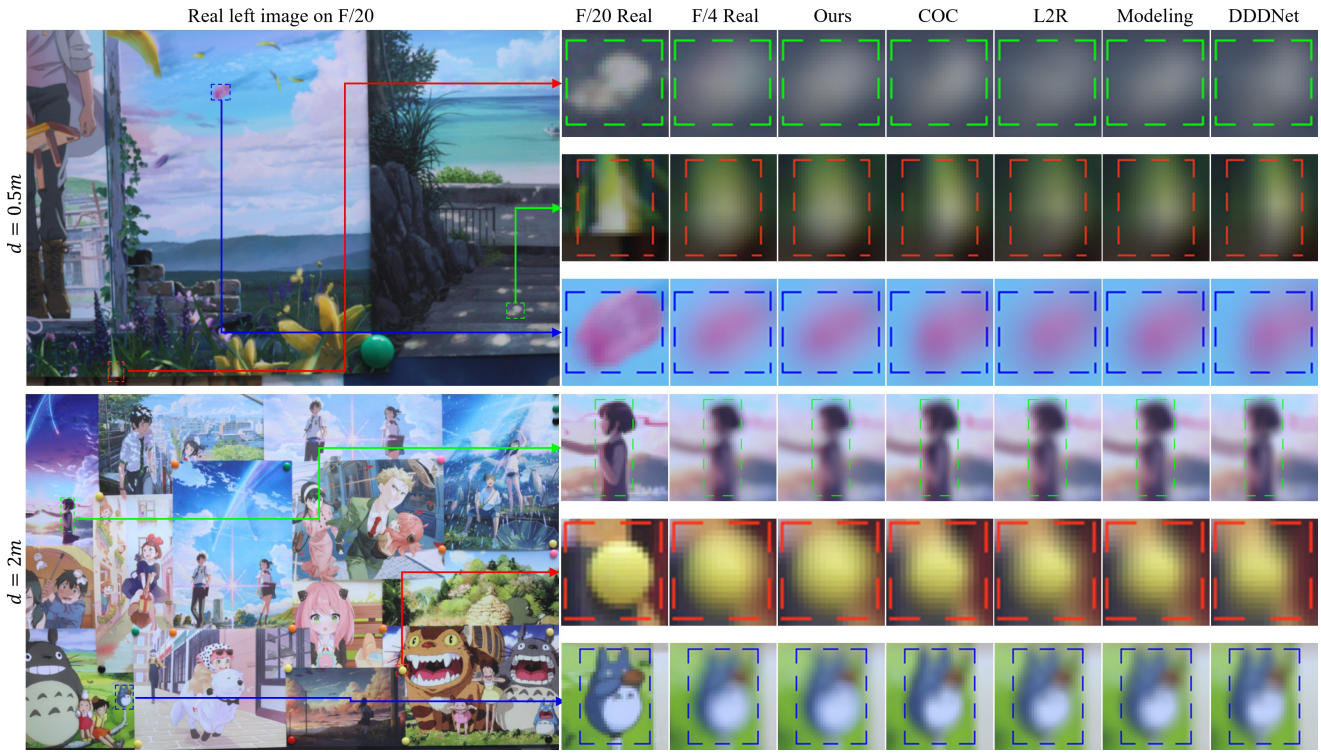


Figure 6. **Qualitative results of simulated DP images.** Evaluate the similarity between simulated (ours, COC, L2R [2], Modeling [26], and DDDNet [23]) and real F/4 defocused left DP images at two depths (0.5m and 2m). Compared to real F/4 defocused images, images simulated by other methods exhibit varying sizes of patterns, incomplete shapes, and texture shifts in different directions before and after the focus distance (1m). Our scheme produces the most realistic simulated images.

(focus distance is 1m) causes a significant phase difference between the real $PSF_L$ and $PSF_R$, and the phase difference at depths of 0.5m and 1.5m is opposite. When $p$ is on the optical axis, all other simulators fail to provide accurate DP PSF results close to reality. Farther from the optical axis, real $PSF_L$ and $PSF_R$ become asymmetric, with less significant phase differences and more off-axis aberrations. The differences between DP PSFs provided by other simulators and real DP-PSFs become more apparent. In contrast, our ray-traced simulator predicts accurate results at all depths and positions.

We present quantitative results in Tab. 1. Specifically, we choose the Normalized Squared Difference (NSD) method and the Normalized Cross-Correlation (NCC) method from OpenCV to evaluate the error and similarity between the real and simulated DP PSFs. We sampled 50 DP PSFs at 0.5m and 1.5m within the valid imaging region. Since the DP PSF is rotationally symmetric around the optical axis, we performed uniform sampling in the first quadrant. Our method achieves the highest similarity (0.915 NCC) and lowest error (0.133 NSD), demonstrating that we have addressed the domain gap between simulated and real DP PSFs caused by lens aberrations and sensor phase splitting.

In subsequent experiments, the F-number will be adjusted back to F/4 due to the large blur kernel size caused by defocus at F/1.8, which resulted in GPU memory shortages.

Table 1. **Quantitative results of DP PSF simulators.** The NCC and NSD between the simulated and real DP PSFs at 50 points were evaluated. Our method achieved the best similarity and the lowest error.

| Method | Ours | COC | L2R [2] | Modeling [26] | DDDNet [23] |
|---|---|---|---|---|---|
| NCC↑ | **0.915** | 0.672 | 0.638 | 0.707 | 0.589 |
| NSD↓ | **0.133** | 0.448 | 0.523 | 0.423 | 0.625 |

### 5.4. Evaluation on simulated DP image

When comparing simulated and real DP images, to explore the degradation of simulated DP images caused by defocus effects at different depths, we captured richly textured planar images from 0.5m to infinity, perpendicular to the camera's optical axis, with denser sampling at shallower depths of field. This resulted in 56 scenes with varying depths, including 11 with infinite depth. For each scene, real DP images were captured at F/4 and F/20. The F/20 image served as the AiF RGB image, replicating its capture-time depth information into the depth map. Then, we input the F/20 RGB-depth paired images into DP simulators to obtain the simulated F/4 defocused DP image.

As shown in Fig. 6, we compare the qualitative analysis results of the defocused left DP images at depths of 0.5m

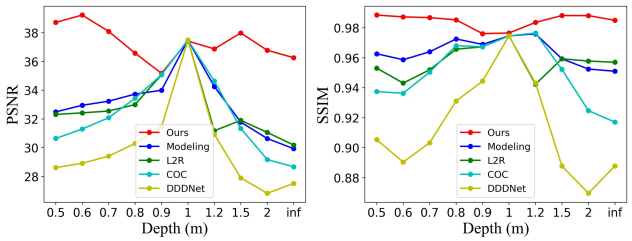


Figure 7. **Quantitative results of simulated DP images.** Evaluate the similarity of 56 real and simulated (ours, COC, L2R [2], Modeling [26] and DDDNet [23]) planar scene F/4 DP images at different depths using PSNR (↑) and SSIM (↑) metrics. As the depth of simulated images deviates further from the focus distance (1m), the less realistic other methods become, whereas our scheme maintains the highest accuracy across all depths.

and 2m. Due to errors in simulating the DP PSF, other methods result in simulated images with textures shifted to the right at a depth of 0.5m and to the left at a depth of 2m, compared to the real left DP image. Moreover, other methods show different image texture sizes and incomplete shapes. However, the DP images simulated by our Sdirt method show minimal deviation in image textures.

As shown in Fig. 7, which presents quantitative analysis results for all scenes, we assess the similarity between simulated and real images using PSNR and SSIM metrics. As the simulated image's depth moves further away from the focus distance (1m), the neglect of aberrations and phase splitting in other methods leads to a more significant decrease in their metrics. In contrast, our Sdirt method achieves the best results at all depths, with average PSNR/SSIM values of 37.1982/0.9845. These values indicate that our simulated DP images are very realistic.

# 6. DfDP Estimation Results

## 6.1. Implementation details

To train the DfDP model, we simulate F/4 DP images using the Sdirt based on RGBD datasets [21, 30] . NYU Depth dataset [30] is parsed by [12] and contains 50,688 indoor scenes. We use AdamW optimizer [19] and CosineAnnealing scheduler [20] with an initial learning rate of $1 \times 10^{-4}$. The training batch size is 4, and we train for 50 epochs. For each epoch, 2000 DP-depth paired data are randomly selected for training. After training, we directly evaluate the model's generalization on the real-world test set without any fine-tuning.

## 6.2. A new test set DP119

Compared to previous simulation methods, Sdirt requires more physical priors. As shown in Tab. 2, there is no public dataset that meets the following conditions: 1. Real DP-depth paired data for simulating DP images. 2. Known lens structure for ray tracing. 3. A fixed focus distance, as changing it alters the DP PSF of the same object point. Therefore, we collected the DP119 dataset, which consists of 45 planar scenes, 44 box scenes, and 30 casual scenes, totaling 119 scenes.

Table 2. Summary of existing DP datasets.

| | DPDD [1] | L2R [2] | DPNet [9] | DP5K [16] | DDDNet [23] | Modeling [26] | DP119 Ours |
|---|---|---|---|---|---|---|---|
| Real captured | ✓ | × | ✓ | ✓ | × | ✓ | ✓ |
| Paired depth | × | ✓ | ✓ | ✓ | ✓ | × | ✓ |
| Lens structure | ✓ | - | × | ✓ | - | × | ✓ |
| Fixed focus dist. | ✓ | - | × | × | - | × | ✓ |

The casual scenes represent common scenarios, suitable for evaluating the robustness of simulation models. The planar and box scenes are richly textured and do not contain textureless areas, which helps the model utilize aberration and phase difference cues for depth estimation. Textureless areas, even when defocused, do not introduce any aberrations or phase differences, making it difficult to estimate depth and interfering with the evaluation. Therefore, planar and box scenes are ideal for assessing whether the model addresses the domain gap between simulated and real DP images. Real DP images are captured by the Canon RF50mm lens with a R6 Mark II at F/4 and 1m focus. GT depth maps for planar scenes are created by the depths at the capture time, while those for box and casual scenes are obtained from LiDAR scans. For more information about the DP119 test set, please refer to the Supplementary.

## 6.3. Evaluation

We present qualitative and quantitative results for the DfDP model trained by COC, L2R [2], Modeling [26], DDDNet [23] and our Sdirt. We deploy their DfDP model with their DP PSF simulator source code, and all methods use the same depth estimation network structure as ours for a fair comparison.

Figure 8 shows the depth estimation results for four casual scenes, with a color bar in meters added to decorate the images. It can be seen that although other methods provide relative positional information in the center region, they fail to correctly estimate relative positional relationships in the edge regions, and both the center and edge regions have large absolute positional errors. This is because the DP PSFs predicted by these methods exhibit phase symmetry and shift-invariance, while real DP PSFs exhibit significant phase asymmetry and aberrations (which do not have shift-invariance), causing their methods to deviate from the real DP PSF characteristics, resulting in large errors, especially in the edge regions where the errors are most severe. The results of L2R [2] and Modeling [26] are similar because the peak positions of their DP PSFs are quite close. In contrast, our method provides accurate relative positional information across the entire image area with minimal absolute positional errors, as our DP PSF is highly consistent with the real DP PSF in both the center and edge regions.

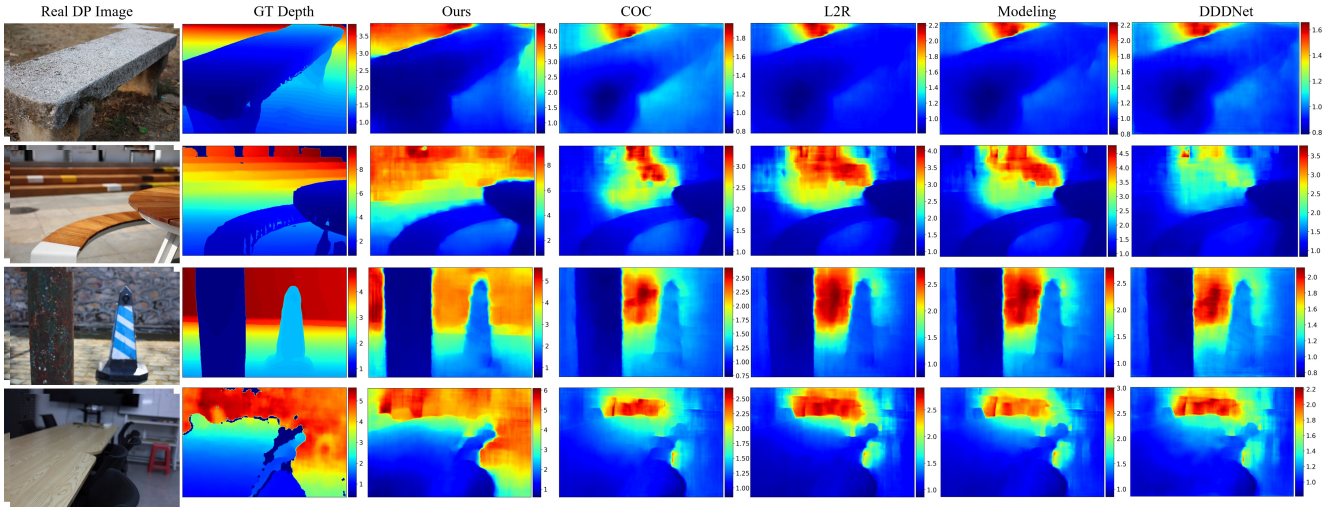


Figure 8. **Qualitative results of depth estimation.** Evaluate DfDP models by with COC, L2R [2], Modeling [26], DDDNet [23] and our Sdirt at four casual scenes. Each result image is decorated with a color bar in meters. Their depth estimation results show partial accuracy in relative positional relationships but large absolute positional errors. Our depth estimation results, however, demonstrate accurate in both relative and absolute positions with minimal errors. (Best viewed in colour and enlarge on screen.)

Table 3. **Quantitative results of depth estimation.** The DfDP model trained based on the Sdirt scheme achieves the best performance across most scenarios and evaluation metrics.

| Scene | Method | MAE↓ | MSE↓ | Abs.r.↓ | Sq.r.↓ | Acc-1↑ | Acc-2↑ |
|---|---|---|---|---|---|---|---|
| planar | Ours | **0.0845** | **0.0109** | **0.0871** | **0.0095** | **0.9849** | **0.9997** |
| | COC | 0.2085 | 0.1001 | 0.1801 | 0.0659 | 0.6670 | 0.8990 |
| | L2R [2] | 0.2418 | 0.1271 | 0.2112 | 0.0841 | 0.6319 | 0.8536 |
| | Modeling [26] | 0.2284 | 0.1142 | 0.2004 | 0.0766 | 0.6496 | 0.8725 |
| | DDDNet [23] | 0.2583 | 0.1485 | 0.2191 | 0.0958 | 0.5648 | 0.8089 |
| box | Ours | **0.1197** | **0.0339** | **0.0906** | **0.0231** | **0.9474** | **0.9812** |
| | COC | 0.3375 | 0.1804 | 0.2442 | 0.1116 | 0.4412 | 0.8277 |
| | L2R [2] | 0.3866 | 0.2284 | 0.2803 | 0.1412 | 0.3651 | 0.7156 |
| | Modeling [26] | 0.3655 | 0.2055 | 0.2660 | 0.1278 | 0.3907 | 0.7758 |
| | DDDNet [23] | 0.4177 | 0.2676 | 0.2975 | 0.1636 | 0.3456 | 0.6274 |
| casual | Ours | **0.2702** | **0.2294** | **0.4632** | 0.7241 | **0.8236** | **0.9314** |
| | COC | 0.7925 | 1.8579 | 0.5461 | 0.6821 | 0.3318 | 0.6103 |
| | L2R [2] | 0.8170 | 1.7487 | 0.5597 | 0.6719 | 0.2760 | 0.5315 |
| | Modeling [26] | 0.7934 | 1.7256 | 0.5510 | **0.6655** | 0.2978 | 0.5732 |
| | DDDNet [23] | 0.8931 | 2.0624 | 0.5752 | 0.7135 | 0.2481 | 0.4685 |

We evaluate all models on the DP119 test set using the following metrics to assess the quantitative results: Mean Absolute Error (MAE), Mean Squared Error (MSE), Absolute Relative Error (Abs.r.), Squared Relative Error (Sq.r.), accuracy with $\delta < 1.25$ (Acc-1) and accuracy with $\delta < 1.25^2$ (Acc-2). Since the planar and box scenes are richly textured and do not have interference from textureless areas, the simulation models can rely on abundant aberration and phase difference cues for depth estimation. As shown in Tab. 3, in the simplest planar scene, our model achieves the best results across all metrics (0.9849 Acc-1). In the box scenes, all models experience a performance drop compared to the planar scenes, but we still achieve the optimal metrics (0.9474 Acc-1). These results demonstrate that our model is highly realistic and has effectively addressed the domain gap between simulated and real DP images. Furthermore, we test the robustness of all models on the casual scenes. Due to the textureless areas in the casual scenes, the performance of all models significantly degrades. However, our model still achieves the best metrics (0.8236 Acc-1), far surpassing the second-best model (0.3318 Acc-1). This further demonstrates that our model is the most realistic and has the best generalization performance. For more sample results from the DP119 test set, please refer to the Supplementary.

## 7. Conclusion and Discussion

Simulated DP images can address the scarcity of DP-depth paired data but face a domain gap between simulated and real DP data. In this work, we propose Sdirt to bridge this domain gap. Specifically, we calculate the DP PSF for points in object space using ray tracing, and leverage a network to predict it. Then, we render DP images based on the predicted DP-PSFs. Experimental results show that the proposed Sdirt scheme can simulate more realistic DP data. Moreover, depth estimation models trained based on Sdirt generalize better on real DP images.

We believe that the Sdirt scheme is not limited to DfDP tasks, it can provide extra depth information for any task with known optical imaging system parameters, promising significant applications in scenarios such as smartphones, automobiles, and microscopes in the future. However, Sdirt is only applicable to cameras equipped with a fixed-focus lens (the structure must be open) and a DP sensor (DP images must be available). Currently, only Canon (5D4, R series) meets these requirements. To further expand the application of Sdirt, more camera and smartphone manufacturers need to make these data accessible.

## References


[1] Abdullah Abuolaim and Michael S Brown. Defocus deblurring using dual-pixel data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 111–126. Springer, 2020. 2, 7

[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*, pages 2289–2298, 2021. 1, 2, 5, 6, 7, 8

[3] Abdullah Abuolaim, Radu Timofte, and Michael S Brown. Ntire 2021 challenge for defocus deblurring using dual-pixel images: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 578–587, 2021. 2

[4] Abdullah Abuolaim, Mahmoud Afifi, and Michael S Brown. Improving single-image defocus deblurring: How dual-pixel images help through multi-task learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1231–1239, 2022. 1, 2

[5] Hadi Alzayer, Abdullah Abuolaim, Leung Chun Chan, Yang Yang, Ying Chen Lou, Jia-Bin Huang, and Abhishek Kar. Dc2: Dual-camera defocus control by learning to refocus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21488–21497, 2023. 2

[6] 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. 4

[7] Myungsub Choi, Hana Lee, and Hyong-euk Lee. Exploring positional characteristics of dual-pixel data for camera autofocus. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13158–13168, 2023. 1, 2

[8] Geoffroi Côté, Fahim Mannan, Simon Thibault, Jean-François Lalonde, and Felix Heide. The differentiable lens: Compound lens search over glass surfaces and materials for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20803–20812, 2023. 2

[9] Rahul Garg, Neal Wadhwa, Sameer Ansari, and Jonathan T Barron. Learning single camera depth estimation using dual-pixels. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7628–7637, 2019. 1, 2, 7

[10] Bhargav Ghanekar, Salman Siddique Khan, Pranav Sharma, Shreyas Singh, Vivek Boominathan, Kaushik Mitra, and Ashok Veeraraghavan. Passive snapshot coded aperture dual-pixel rgb-d imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25348–25357, 2024. 2

[11] Chi-Jui Ho, Chin-Cheng Chan, and Homer H Chen. Afnet: A convolutional neural network approach to phase detection autofocus. *IEEE Transactions on Image Processing*, 29:6386–6395, 2020. 1, 2

[12] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. 7

[13] Minjun Kang, Jaesung Choe, Hyowon Ha, Hae-Gon Jeon, Sunghoon Im, In So Kweon, and Kuk-Jin Yoon. Facial depth and normal estimation using single dual-pixel camera. In *European Conference on Computer Vision*, pages 181–200. Springer, 2022. 1, 2

[14] Donggun Kim, Hyeonjoong Jang, Inchul Kim, and Min H Kim. Spatio-focal bidirectional disparity estimation from a dual-pixel image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5023–5032, 2023. 2

[15] Masahiro Kobayashi, Michiko Johnson, Yoichi Wada, Hiromasa Tsuboi, Hideaki Takada, Kenji Togo, Takafumi Kishi, Hidekazu Takahashi, Takeshi Ichikawa, and Shunsuke Inoue. A low noise and high sensitivity image sensor with imaging and phase-difference detection af in all pixels. *ITE Transactions on Media Technology and Applications*, 4(2):123–128, 2016. 1, 3

[16] Feiran Li, Heng Guo, Hiroaki Santo, Fumio Okura, and Yasuyuki Matsushita. Learning to synthesize photorealistic dual-pixel images from rgbd frames. In *2023 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11. IEEE, 2023. 1, 2, 7

[17] Yu Li, Yaling Yi, Dongwei Ren, Qince Li, and Wangmeng Zuo. Learning dual-pixel alignment for defocus deblurring. *arXiv preprint arXiv:2204.12105*, 2022. 1, 2

[18] Yizhou Li, Yusuke Monno, and Masatoshi Okutomi. Dual-pixel raindrop removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 2

[19] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7

[20] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 7

[21] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 4, 7

[22] Sagi Monin, Sagi Katz, and Georgios Evangelidis. Continuous cost aggregation for dual-pixel disparity extraction. In *2024 International Conference on 3D Vision (3DV)*, pages 675–684. IEEE, 2024. 2

[23] 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. 1, 2, 5, 6, 7, 8

[24] Liyuan Pan, Richard Hartley, Liu Liu, Zhiwei Xu, Shah Chowdhury, Yan Yang, Hongguang Zhang, Hongdong Li,

and Miaomiao Liu. Weakly-supervised depth estimation and image deblurring via dual-pixel sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2, 5

[25] Abhijith Punnappurath and Michael S Brown. Reflection removal using a dual-pixel sensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1556–1565, 2019. 2

[26] 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. 1, 2, 5, 6, 7, 8

[27] Lingyan Ruan, Martin Bálint, Mojtaba Bemana, Krzysztof Wolski, Hans-Peter Seidel, Karol Myszkowski, and Bin Chen. Self-supervised video defocus deblurring with atlas learning. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2

[28] Zheng Shi, Ilya Chugunov, Mario Bijelic, Geoffroi Côté, Jiwoon Yeom, Qiang Fu, Hadi Amata, Wolfgang Heidrich, and Felix Heide. Split-aperture 2-in-1 computational cameras. *ACM Transactions on Graphics (TOG)*, 43(4):1–19, 2024. 2

[29] Eun Sub Shim, Kyungho Lee, Junghyung Pyo, Wooseok Choi, Jungbin Yun, Taesub Jung, Kyungduck Lee, Seyoung Kim, Chanhee Lee, Seungki Baek, et al. All-directional dual pixel auto focus technology in cmos image sensors. In *2021 Symposium on VLSI Technology*, pages 1–2. IEEE, 2021. 1, 3

[30] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pages 746–760. Springer, 2012. 4, 7

[31] Przemysław Śliwiński and Paweł Wachel. A simple model for on-sensor phase-detection autofocusing algorithm. *Journal of Computer and Communications*, 1(06):11, 2013. 1

[32] Kunal Swami. Adjust your focus: Defocus deblurring from dual-pixel images using explicit multi-scale cross-correlation. In *International Conference on Computer Vision and Image Processing*, pages 318–330. Springer, 2023. 2

[33] 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. 4, 2, 5

[34] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Transactions on Graphics (ToG)*, 37(4):1–13, 2018. 2

[35] Congli Wang, Ni Chen, and Wolfgang Heidrich. do: A differentiable engine for deep lens design of computational imaging systems. *IEEE Transactions on Computational Imaging*, 8:905–916, 2022. 1, 2

[36] Zhuofeng Wu, Doehyung Lee, Zihua Liu, Kazunori Yoshizaki, Yusuke Monno, and Masatoshi Okutomi. Disparity estimation using a quad-pixel sensor. *arXiv preprint arXiv:2409.00665*, 2024. 2

[37] Shumian Xin, Neal Wadhwa, Tianfan Xue, Jonathan T Barron, Pratul P Srinivasan, Jiawen Chen, Ioannis Gkioulekas, and Rahul Garg. Defocus map estimation and deblurring from a single dual-pixel image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2228–2238, 2021. 1, 2

[38] Hao Yang, Liyuan Pan, Yan Yang, Richard Hartley, and Miaomiao Liu. Ldp: Language-driven dual-pixel image defocus deblurring network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24078–24087, 2024. 2

[39] Xinge Yang, Qiang Fu, Mohamed Elhoseiny, and Wolfgang Heidrich. Aberration-aware depth-from-focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1, 2, 4, 5

[40] Xinge Yang, Qiang Fu, and Wolfgang Heidrich. Curriculum learning for ab initio deep learned refractive optics. *Nature communications*, 15(1):6572, 2024. 1, 2

[41] Yan Yang, Liyuan Pan, Liu Liu, and Miaomiao Liu. K3dn: Disparity-aware kernel estimation for dual-pixel defocus deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13263–13272, 2023. 1, 2

[42] Dafeng Zhang and Xiaobing Wang. Dynamic multi-scale network for dual-pixel images defocus deblurring with transformer. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022. 2

[43] Yinda Zhang, Neal Wadhwa, Sergio Orts-Escolano, Christian Häne, Sean Fanello, and Rahul Garg. Du 2 net: Learning depth estimation from dual-cameras and dual-pixels. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 582–598. Springer, 2020. 2

# Simulating Dual-Pixel Images From Ray Tracing For Depth Estimation

Supplementary Material

## 8. Sdirt Details

### 8.1. Structure parameters of DP composite pixels

We select the Canon RF50mm F/1.8 lens and Canon R6 Mark II camera body as our reference lens and DP sensor. Since the lens parameters are known, with detailed lens data presented in Tab. 4 and the 2D lens structure with ray paths illustrated in Fig. 9(a), it is feasible to perform ray tracing on the lens. However, performing ray tracing on the DP sensor is limited by camera manufacturer's nondisclosure of microlens and sub-pixel structure parameters. Therefore, as described in Sec. 3.1 (Ray-traced DP PSF simulator) and Fig. 3(a) (DP composite pixel structure) of the main paper, we simplify the DP composite pixel structure by modeling the microlens as a thin lens with a radius of $r$ and a focal length of $f$, set $h$ to the distance between the sub-pixel and the microlens, set $w$ to the sub-pixel width, and set $ps$ to the size of the DP composite pixel.

We assign a set of possible values to each structure parameter of the DP composite pixel and identify the optimal parameter combination through grid searching across these parameters. Instead of conducting search experiments directly on the reference lens, we replace it with the Canon RF35mm F/1.8 lens to probe the structure parameters, ensuring isolation between the reference lens and the sensor. Detailed lens data for the Canon RF35mm F/1.8 lens are provided in Tab. 5, and its 2D lens structure with ray paths is illustrated in Fig. 9(b). In the valid imaging region (Fig. 4(a) in the main paper), we select 5 object points at different positions and capture real DP PSFs through the camera. By evaluating the error and similarity between the real DP PSF and the set of DP PSFs simulated with all structure parameters, we determine an optimal parameter combination:

$$\begin{cases} h &= 0.78 * ps \\ f &= 1.44 * ps \\ w &= 0.60 * ps \\ r &= 0.50 * ps \end{cases} \tag{6}$$

Specifically, we choose the Normalized Squared Difference (NSD) method and the Normalized Cross-Correlation (NCC) method 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. 10, 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. (6).

To demonstrate the similarity between real and simulated DP PSFs under the optimal combination of structure parameters, we conduct a qualitative analysis while keeping the lens, F-number, kernel size, and focal distance unchanged from the search experiments. As shown in Fig. 12, we select five points at equal intervals 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 rows is also included in Fig. 12. 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 asymmetrical. This observation aligns with the experimental results presented in Sec. 5.3 (Evaluation on DP PSF simulator) of the main paper. When comparing the ray-traced and real DP PSF, 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.

### 8.2. Local DP PSF convolution

Existing methods generally use the same PSF kernel to convolve the entire image. Yang *et al*. [39] 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:

```
def local_dp_psf_render(input, dp_psf,
    kernel_size=21):
    """ Render DP image with local DP PSF.
    Use the different DP PSF kernel 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
```

Table 4. **Canon RF50mm F/1.8 lens data.** The reference lens used in the 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 | | | | | | |

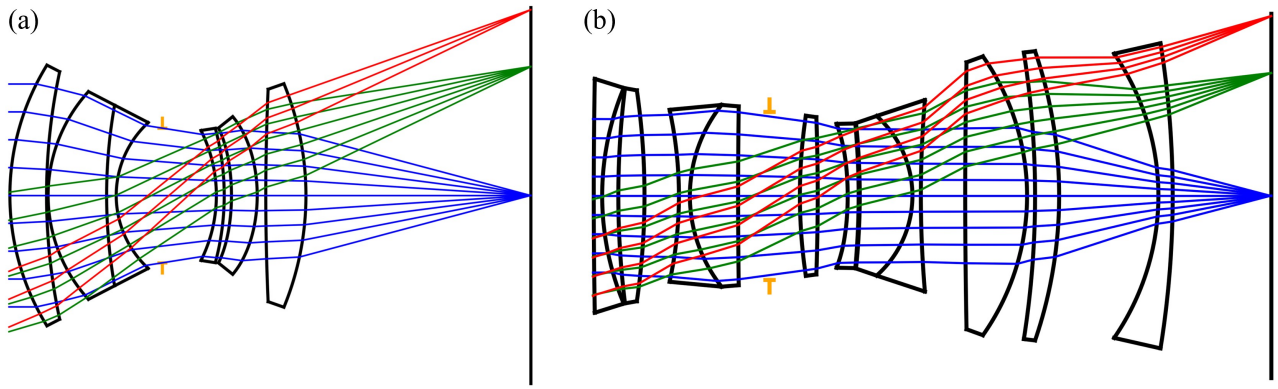


Figure 9. **The 2D lens structure with ray paths.** (a) Canon RF50mm F/1.8 lens. (b) Canon RF35mm F/1.8 lens.

```
    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)
    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)
```

### 8.3. DP PSF predict 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 [33, 39], As shown in Fig. 12, 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 adopt max normalization for GT DP PSF during training and mimic vignetting correction to scale predicted DP PSF when inference. 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.

We present quantitative results in Tab. 6. 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 $512 \times 768$) in the depth map. Our network outperforms those developed by Yang *et al*. [39] and Tseng *et al*. [33] in accuracy, and its low time cost enables high-speed image rendering. In contrast, the network proposed by Tseng *et al*. [33] incurs high time costs due to

Table 5. **Canon RF35mm F/1.8 lens data.** The lens used in detecting DP composite pixel structure 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 | | | | | | |

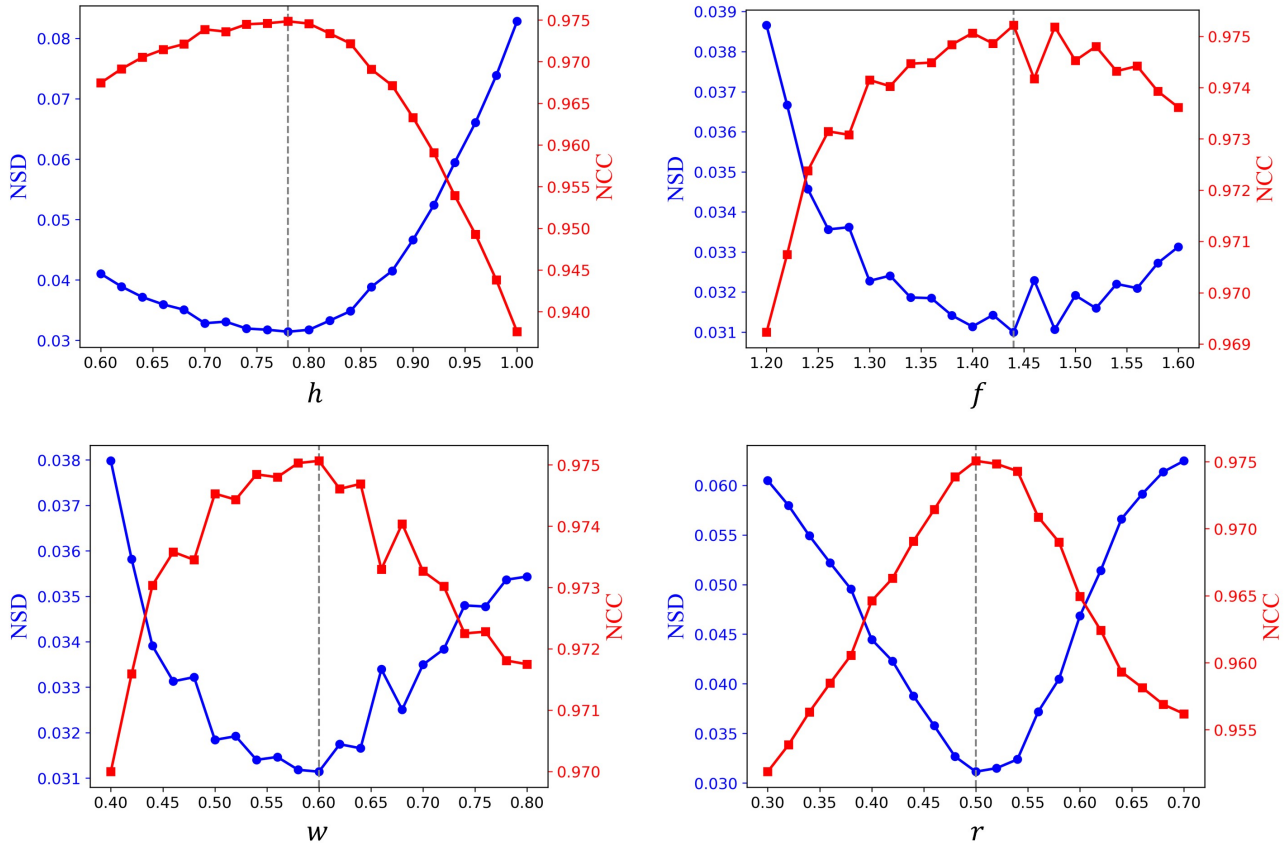


Figure 10. **Error and similarity matching results of structure parameters.** In the matching results, the horizontal axis represents structural parameters (multiples of pixel size). The left vertical axis represents the NSD matching result between real and simulated DP PSFs, while the right vertical axis represents the NCC matching result. To facilitate display, when showing the matching result for any one parameter, we fix the other parameters at their optimal values according to to Eq. (6). Moreover, we use white dashed lines to mark the optimal value for each parameter.

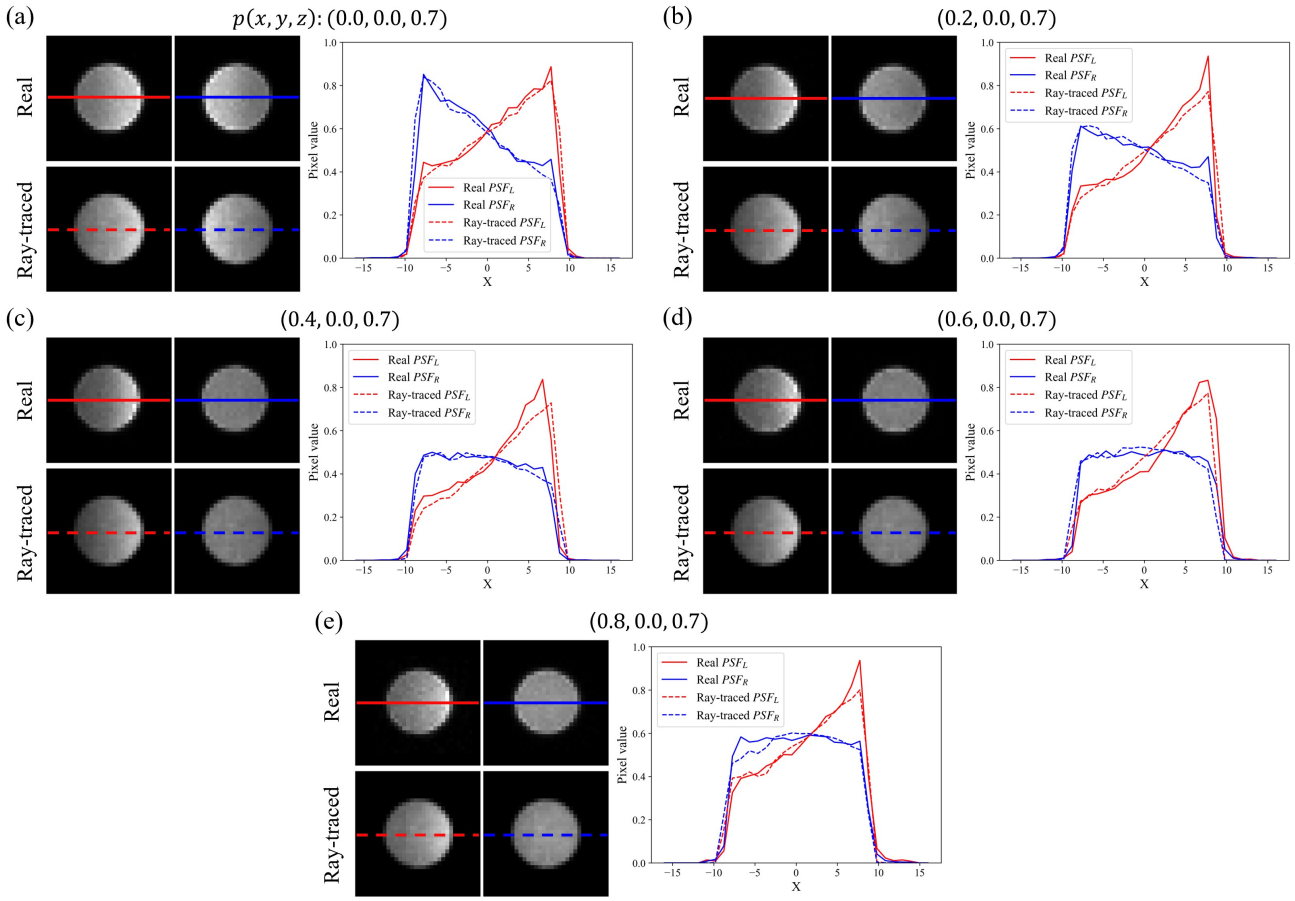


Figure 11. **Qualitative analysis results of DP PSF for real and ray tracing.** We select 5 points within the valid imaging region, and only their x-coordinates increased sequentially, corresponding to (a) - (e). We provide not only the comparison results of real and ray-traced DP PSF ($PSF_L$ on the 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 asymmetric. Our simulated DP PSF, obtained through ray tracing, aligns well with the real DP PSF.

Table 6. Quantitative results of DP PSF predict networks.

| Method | L1 error ↓ | L2 error ↓ | Time(s) ↓ |
|---|---|---|---|
| Ray-traced | - | - | 801.544 |
| Ours | **6.887e-5** | **6.830e-8** | **0.395** |
| Yang *et al*. [39] | 1.487e-4 | 1.381e-7 | **0.395** |
| Tseng *et al*. [33] | 1.124e-4 | 7.685e-7 | 43.96 |

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.

# 9. Depth-from-Dual-Pixel model details

## 9.1. Adjustment of cost volume

In the field of Depth-from-Dual-Pixel (DfDP), we do not pursue the optimal DfDP network structure. Instead, we select [6] 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 a single-direction misalignment to stack left and right image features, forming a 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(c) (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
```

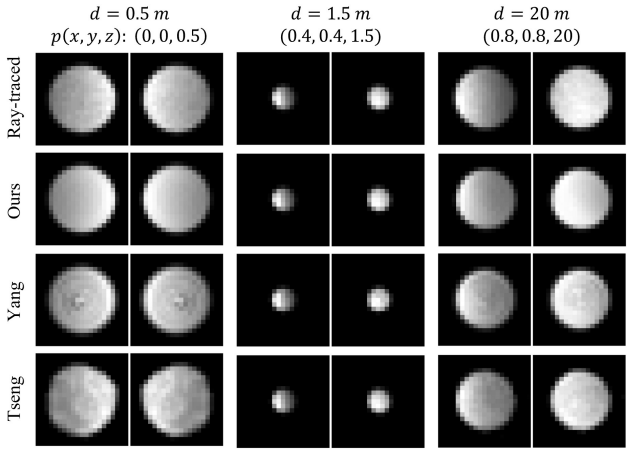


Figure 12. **DP PSF predict network representation result.** Evaluate the ray-traced and network-predicted (ours, Yang *et al*. [39], and Tseng *et al*. [33]) F/4 DP PSFs at three depths (0.5m, 1.5m, 20m) and positions. For small DP PSF radii, all network predictions closely match ray-traced results. For larger radii, the predicted results by [33, 39] show significant deviations, attributed to differences in normalization schemes with ours during training.

```
    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
```

### 9.2. A new test set DP119

We used the Canon RF50mm lens and Canon R6 Mark II camera to capture the dataset, setting the aperture to F/4 and focusing at 1 meter. 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 meters to 2 meters perpendicular to the camera's optical axis, with denser sampling at shallower depths of field. The dataset includes five sets of samples and nine different distances (0.5m, 0.6m, 0.7m, 0.8m, 0.9m, 1.0m, 1.2m, 1.5m, 2m). 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. Due to the simple depth structure of planar scenes, we avoid issues like holes in LiDAR depth maps or 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. 13(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 0.5m to 2m, 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. 13(b), we present samples of box scenes.

For the casual scenes, we directly captured ordinary real-world scenes, intentionally controlling the depth within 0.5m to 10m, 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. 13(c), we present samples of casual scenes.

### 9.3. More DfDP results on the DP119 test set

Fig. 14 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, 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 (1m). The depth estimation results of L2R [2] and Modeling [26] are similar, as their DP PSF peak positions are

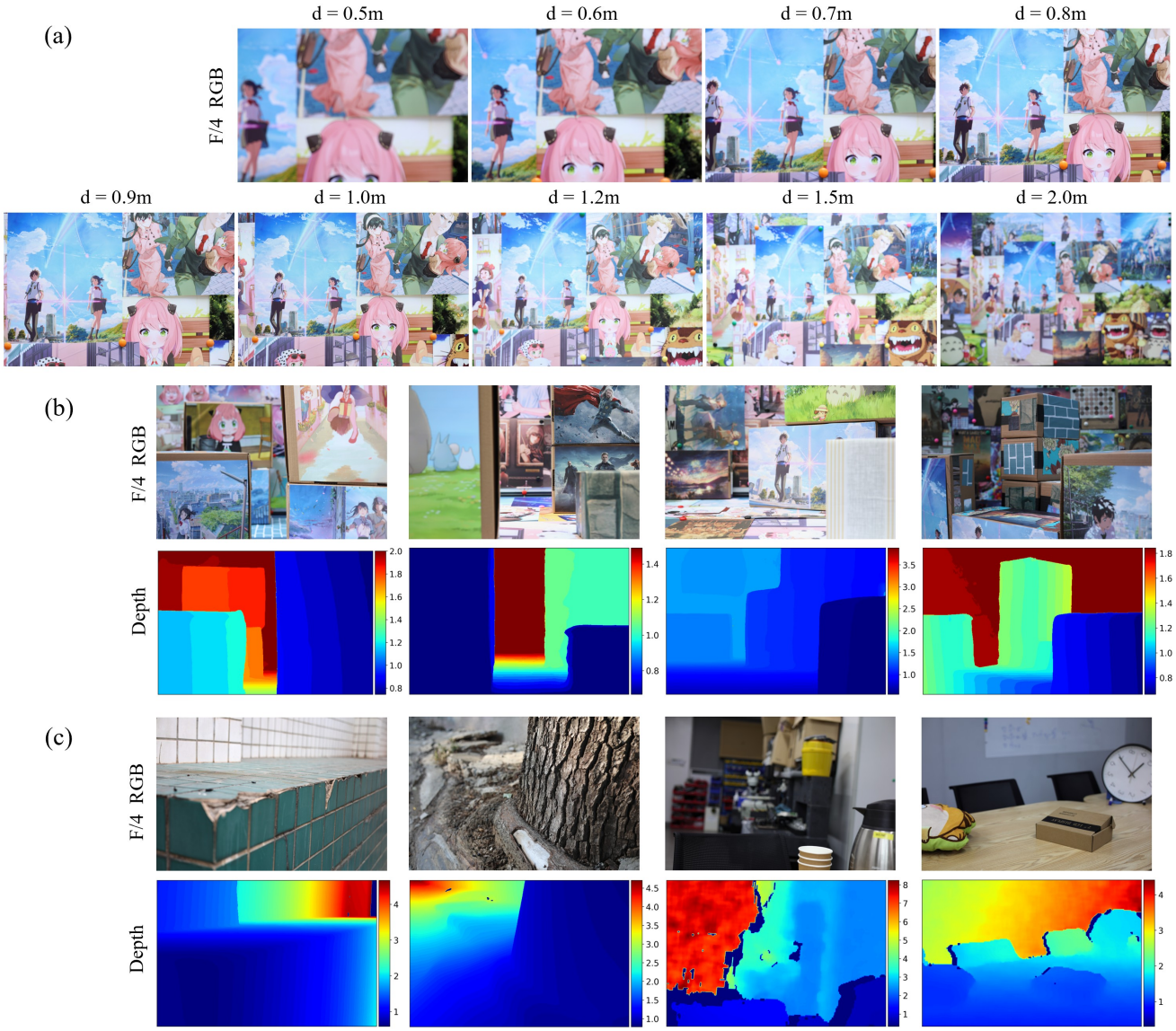


Figure 13. **Examples of scenes in the DP119 dataset.** Each depth map is decorated with a color bar in meters. (a) Planar scene samples. The d is the planar depth value at the time of capture. (b) Box scene samples. (c) Casual scene samples.

spatially close. A similar trend is observed in box scenes 1 and 2, where other models show a strong bias toward the focal distance (1m) 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. In addition to aberration and phase difference information, the models can also use additional structural information to predict depth. 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 (1m). Moreover, in casual scenes 3 and 4, large textureless areas appear, missing aberration and phase difference cues. Depth estimation models can only infer the depth of these textureless areas along structural information, leading to significant relative positional position 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.

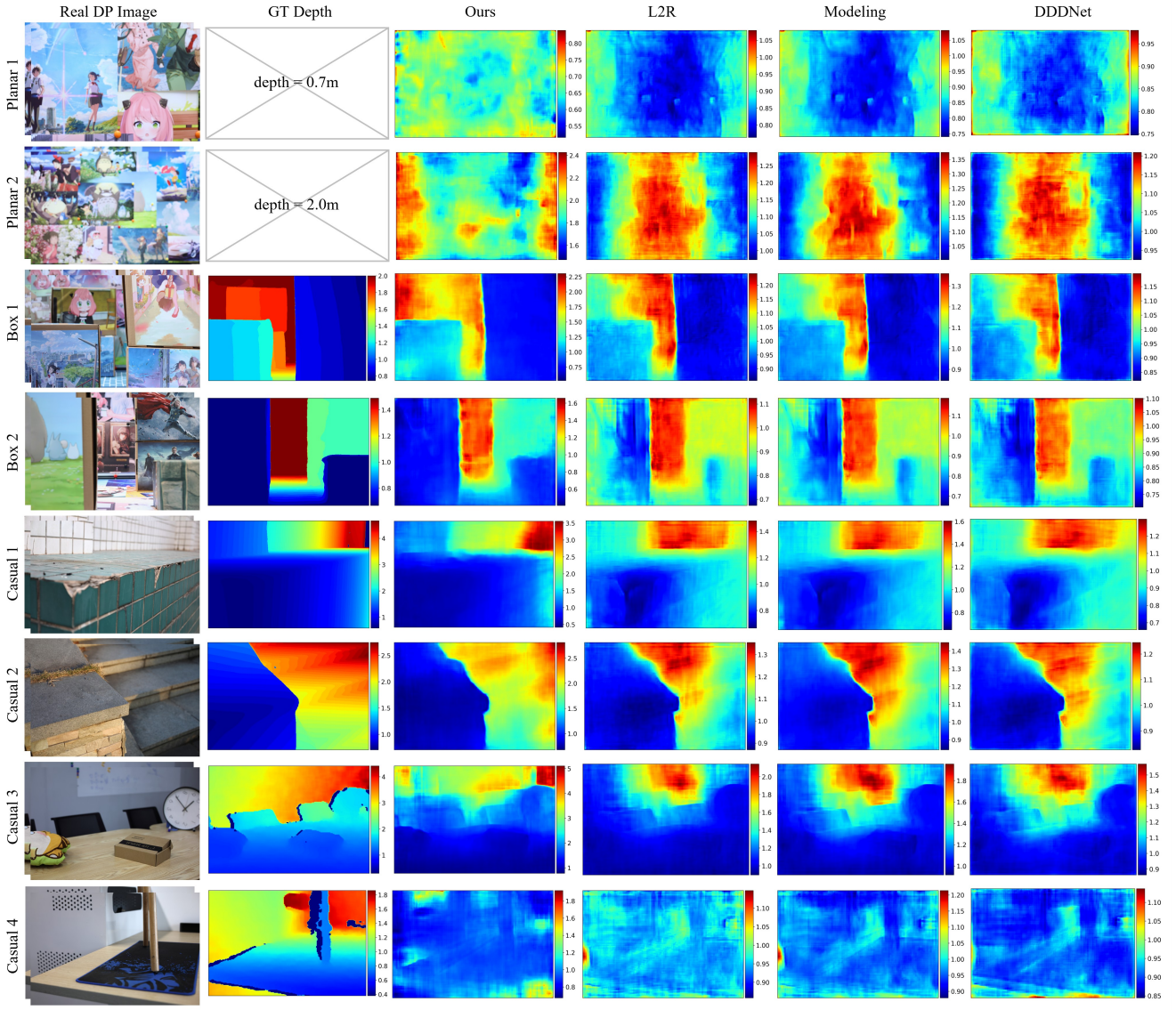


Figure 14. **The depth estimation results from different simulators on the DP119 test set.** Evaluate DfDP models by with L2R [2], Modeling [26], DDDNet [23] and our Sdirt on various DP119 dataset scenes. Each result image is decorated with a color bar in meters. Their depth estimation results show partial 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 colour and enlarge on screen.)