Abstract

The dual-pixel (DP) hardware works by splitting each pixel in half and creating an image pair in a single snapshot. Several works estimate depth/inverse depth by treating the DP pair as a stereo pair. However, dual-pixel disparity only occurs in image regions with the defocus blur. The heavy defocus blur in DP pairs affects the performance of matching-based depth estimation approaches. Instead of removing the blur effect blindly, we study the formation of the DP pair which links the blur and the depth information. In this paper, we propose a mathematical DP model which can benefit depth estimation by the blur. These explorations motivate us to propose an end-to-end DDDNet (DP-based Depth and Deblur Network) to jointly estimate the depth and restore the image. Moreover, we define a reblur loss, which reflects the relationship of the DP image formation process with depth information, to regularise our depth estimate in training. To meet the requirement of a large amount of data for learning, we propose the first DP image simulator which allows us to create datasets with DP pairs from any existing RGBD dataset. As a side contribution, we collect a real dataset for further research. Extensive experimental evaluation on both synthetic and real datasets shows that our approach achieves competitive performance compared to state-of-the-art approaches.

1. Introduction

The Dual-Pixel (DP) sensor has been used by DSLR (digital single-lens reflex camera) and smartphone cameras to aid focusing. Though the DP sensor is designed for auto-focus [32, 16, 15], it is used in applications such as, depth estimation [10, 29, 43], defocus deblurring [1], reflection removal [30], and shallow Depth-of-Field (DoF) images synthesis [36]. In this paper, we model the imaging process of the DP sensor theoretically, and show its effectiveness for simultaneous depth estimation and image deblurring.

A DP camera simultaneously captures two images, one formed from light rays passing through the right half of the aperture, and one from those passing through the left half of the aperture. Because of the displacement of these two halves, the two images form a stereo pair. The primary role of DP cameras is to enable auto-focus. The set of points in the world that are exactly in focus forms a plane parallel to the lens (perpendicular to the axis of the camera), and points that lie on this plane are both in focus, and imaged with zero-disparity in the two images. Points that lie further away from the camera are imaged with positive disparity in the two images and those that are closer have negative disparity. By detecting and modelling these disparities, it is possible to compute scene depth, as in standard stereo imaging. (In auto-focus these disparities are used to move the sensor, or the lens, to focus on any part of the scene.)

At the same time, points other than those lying in the in-focus plane will be significantly out of focus, which raises the possibility of using depth-from-defocus techniques to estimate depth in the scene. This out-of-focus effect is more significant than with ordinary stereo pairs, making stereo matching potentially more challenging for DP cameras. However, because of the small base-line, between the
two half-apertures, occlusion effects are minimal.

Thus, computing depth from a DP camera can be seen as an out-of-focus stereo estimation problem. This paper aims to combine depth-from-disparity and depth-from-defocus approaches in a single network, demonstrating the advantage of modelling them both simultaneously.

To accomplish this, we study the imaging process of a DP pair and provide a mathematical model handling the intrinsic problem of depth from defocus blur for a DP pair. Our method can jointly recover an all-in-focus image and estimate the depth of a scene (see Fig. 1). Our contributions are summarised as follows:

1. We propose a theoretical DP model to explicitly define the relationship between depth, defocus blur and all-in-focus image;
2. We design an end-to-end DP-based Depth and Deblur Network (DDDNet) to jointly estimate the depth map and restore the sharp image;
3. We formulate a reblur loss based on our DP model which is used to regularise depth estimate in training;
4. We create a DP image simulator, which enables us to create DP datasets from any RGB-D dataset;
5. We collect a real dataset to stimulate further research.

Extensive experimental evaluation on both synthetic and real data shows that our approach achieves competitive performance compared to state-of-the-art approaches.

2. Related Work

For space reasons, we briefly review depth estimation from monocular, stereo, DP, and defocus.

**Monocular.** Supervised monocular depth estimation methods typically rely on large training datasets [22, 9]. Self-supervised methods [31, 34, 20, 12] often use estimated depths and camera poses to generate synthesized images, serving as supervision signals.

**Stereo.** To summarize, a stereo depth estimation framework [13, 41, 25, 6, 27, 40, 2] often contains: matching cost computation, cost aggregation, and disparity optimization. Recently, several speed-up methods [37, 39] are available.

**Dual-pixel.** A DP pair has defocus blur and tiny baseline. To tackle the tiny baseline, Wadhwa et al. [36] utilize a small search size in block matching. Garg et al. [10] present the first learning models for affine invariant depth estimation to tackle affine ambiguity. Zhang et al. [43] present a Du2Net that uses a wide baseline stereo camera to compensate for the DP camera. However, it is specifically tailored for the ‘Google Pixel’ (narrow aperture, small DoF). The above three methods ignored the blur cue that could in principle fill in the parts where the disparity is imprecise (due to the tiny baseline). To tackle the defocus blur in a DP pair, Aboualam et al. [1] first propose a DPDeblur that remove the blur effect blindly. Recently, Punnappurath et al. [29] use a point spread function (PSF) to model the defocus blur and formulate the DP disparity by the PSF. However, the symmetry assumption of the PSF (only holds for constant depth region) limits its application for real-world scenario. Moreover, the method is time-consuming and has three steps (not end-to-end). In contrast, we study the imaging process of a DP pair and give a mathematical model jointly handling the intrinsic problem of depth from defocus blur for a DP pair.

**Depth from defocus and deblurring.** Depth from defocus has the same geometric constraints as disparity but different physiological constraints [8]. The estimated depth, also dubbed as defocus map, is commonly used to guide the deblurring [3, 33, 38, 26, 23, 44, 21]. DMENet [19] proposes the first end-to-end CNN architecture to estimate a spatially varying defocus map and use it for deblurring.

**DP data.** Only Canon and Google provide DP data to customers though most DSLR and smartphone cameras have the DP sensor. Several researchers [36, 10, 43] use ‘Google Pixel’ to collect data. However, smartphone cameras use a fixed and narrow aperture that cannot be adjusted. Canon is used in [1, 30, 29] with different aperture sizes, but it is expensive. Both of the two sensors are hard to get the associated GT information, such as depth. In parallel, the rise of deep learning has led to an explosion of demands for data. In this paper, we present the first DP simulator that synthesizes DP images with GT, from any RGB-D data.

In summary, both the stereo cue and defocus cue provided by a DP pair benefit the depth estimation. In this paper, we proposed a DP model in Section 3 and a DDDNet in Section 4 to jointly estimate the depth and restore the image.

3. Dual-Pixel Image Formation

In this section, we first discuss the formation of the two DP images and the defocus blur in Section 3.1. Then, in Section 3.2, we present our DP model and explore how to synthesize DP images from the RGB-D image. Based on the DP model, we build a DP simulator in Section 3.3.

3.1. Model of a dual-pixel camera

A DP camera can be modelled as an ordinary camera with a lens satisfying the thin-lens model, in which two (or more) images $I_L$ and $I_R$ are simultaneously captured. Here, the subscripts $L$ and $R$ denote the left and right view separately. In the model, the focal plane of the camera can be considered as consisting of two identical placed focal planes, one of which captures light rays coming from the left side of the lens, and the other captures light from the right side of the lens. The two images, $I_L$ and $I_R$, can be considered as images taken by two different (but coplanar) lenses, corresponding to regions in the aperture of a thin-lens.

To be general, let it be assumed that the light for image $I_L$ passes through a region $A_L$ of the lens, and that for image $I_R$ comes from a region $A_R$. Let us consider the case
where $A_L$ and $A_R$ are infinitesimally small, consisting of just two points $C_L$ and $C_R$ (see Fig. 2). Choose a coordinate frame such that the lens lies on the plane $X = 0$, and the centre of the lens is at the origin. In addition, let the “world” lie to the left of the lens (that is with points $X < 0$), and the camera lie to the right of the lens. Let $W$ consisting of points $X = (X, Y, Z) \text{ }^\dagger$ with $X < -f$, and let $W'$ be the 3D image of the virtual world, created by the lens, lying to the right of the lens. In particular, if $X' = (X', Y', Z') \in W'$, then $X' > f$. Here, $f$ is the focal length of the lens. Consider a plane of sensors (called a focal plane) lying inside the camera. Assume that this is parallel to the lens, so it is a plane defined by the equation $X = F$, with $F > 0$. The symbol of $F$ is chosen to imply that this is the distance of the focal plane from the lens, but it is not the same as the focal length of the lens, which is $f$.

**Observation.** An image $I_L$ of the world $W$ formed from rays passing through a point $C_L$ in the focal plane is the same as a pin-hole camera image of the virtual world $W'$ with projection centre $C_L$ and the same focal plane. The same statement is true for the image $I_R$ formed from rays passing through $C_R$.

**Proof.** Refer to Fig. 2. Consider a world point $X$, mapped by the lens to a virtual world point $X'$. This means that if $C$ is any point on the lens, then the ray $XC$ is refracted by the lens to the ray $CX'$. In particular, this is true for the point $C_L$ lying on the lens. Let the point $Y$ be the point where the ray $C_L X'$ (extended if necessary) meets the focal plane. Then $Y$ is the point where $X$ is imaged in $I_L$. This is because a ray traverses the path $XC_L Y$, passing through $X'$ on the way. On the other hand, looked at in terms of the pinhole camera with centre $C_L$, the ray from $C_L$ to $X'$ meets the focal plane at $Y$. This means that $Y$ is the image of the point $X'$ in the pinhole image taken from $C_L$.

Therefore, if the two regions were single points $C_L$ and $C_R$, then the two images $I_L$ and $I_R$ are exactly a stereo pair of images of the virtual world $W'$. Although the two points $C_L$ and $C_R$ are necessarily very close together, the virtual world is also very small and very close to the camera lens, so there will be appreciable disparity.

\[ X' = (X', Y', Z') = \frac{f}{f+X}(X, Y, Z). \]  

This point is then projected from each point $C \in A$ onto the focal plane $X = F$. The projection rays, through $X'$ from points in $A$ form a cone with cross-section $A'$ and vertex $X'$. This cone meets the focal plane in a further cross-section $A''$. Since the lens and focal plane are parallel, $A$ and $A''$ will be similar regions, in the sense that there is a

\[1\text{All vectors are column vectors.} \]
similarity transform relating \( A \) and \( A' \). Even more, this is simply a scaling and translation (where the scale may be positive or negative). In particular, a point \( C \in A \) maps to a point \( C' = sC + t \). Here, \( t \) is a 2-dimensional offset and \( s \) is a scale. For a given point \( X \), the values of \( s \) and \( t \) are constant, not dependent on the particular point \( C \) chosen, but they vary according to the point \( X \) chosen.

Refer to Fig. 4. In particular, by similar triangles [14]

\[
C' = T(C) = (1 - s)C + sX',
\]

where \( s = F/d' \).

**Image coordinates.** We start with an RGB-D image, so each pixel in an image comes with an associated depth. For simplicity, we assume that all distances (including depth) are measured in pixel coordinates.

We make the following assumptions. A pixel with coordinates \( (y, z) \) corresponds to a ray in space defined by \(-d(1, y/f, z/f)\) for varying \( d \). In particular, since each point in the image comes with a depth, we are assuming that the point imaged at point \((y, z)\) has 3D coordinates \((-d, -dy/f, -dz/f)\). This corresponds to a pinhole camera with focal plane given by \( X = f \) and camera centre (pinhole) at the origin.

A point \( X = -d(1, y/f, z/f) \) at depth \( d \) will be mapped to a point \( X' = d'(1, y/f, z/f) \), where \( 1/d + 1/d' = 1/f \), or

\[
d' = fd/(d - f).
\]

Let \( C = (0, Y_0, Z_0) \) be a point in \( A \) (lying on the lens plane \( X = 0 \)). The line from \( C \) through \( X' = d'(1, y/f, z/f) \) is expressed as \((1 - s)C + sX'\) for varying values of \( s \). This line meets the plane \( X = F \) when \( s = F/d' \). The coordinates of this point are therefore

\[
T(X, C) = \frac{d'}{d}C + \frac{F}{d'}X'
= \frac{d'}{d}C + F(1, y/f, z/f).
\]

We are only interested in the \((Y, Z)\)-coordinates, in which case we have

\[
T(X, C) = \frac{d'}{d}(Y_0, Z_0) + F(y/f, z/f).
\]

Here, the transformation function \( T(X, C) \) for the left and right image has been defined with points \( C_L \) and \( C_R \) in \( A_L \) and \( A_R \) respectively.

### 3.3. DP simulator

Given a sharp image \( I \in \mathbb{R}^{H \times W} \) and its associated depth map \( d \in \mathbb{R}^{H \times W} \), we can simulate a DP image pair \( (I_L, I_R) \) as follows. Here, \( H \) and \( W \) are the image height and width, and it is also assumed that focal length \( f \) and focal-plane-sensor distance \( F \) are known (measured in units of pixel-size).

For each pixel \((y, z)\) in \( I \), as shown in Fig. 4 the intensity of a pixel is spread over regions \( R_L \) and \( R_R \), in the left and right view of the DP pair. Each region contains a set of points \( p \), containing \(|R| \) pixels, and the intensity \( I(y, z) \) of the pixel \((y, z)\) is spread out evenly over this set of pixels. Now running over all pixels \( I(y, z) \), and summing, a pair of dual-pixel images are created. This is essentially blurring with depth-dependent blur kernel.

This operation is expensive computationally, since it requires looping over each pixel \((y, z)\) in the image, as well as each pixel in the region \( R \), thus 4 levels of looping.

We utilize the concept of ‘integral image’ to speed up the computation, so that it has complexity independent of the size of the region \( R \), thus of order \( \mathcal{O}(n) \) where \( n \) is the number of pixels. In this description, we assume that the left and right halves of the aperture are approximated by rectangles, though with some amount of extra computation, more general shapes may be accommodated.

Given a pixel \((y, z)\) of the RGB-D image at a given depth, its corresponding light rays will pass through every point of \( A_L \). To compute the blur-region \( R_L \), one needs only to compute the destinations of the rays passing through the four corners of \( A_L \), which will be denoted by \( p_{tl} = (y_{tl}, z_{tl}) \), \( p_{tr} = (y_{tr}, z_{tr}) \), \( p_{bl} = (y_{bl}, z_{bl}) \) and \( p_{br} = (y_{br}, z_{br}) \) in the left image \( I_L \). The positions of these four points are given by Eq. (5). Here, the subscript \( tl, tr, bl \) and \( br \) denotes top-left, top-right, bottom-left and bottom-right, respectively.

We create a differential mask \( I_L \in \mathbb{R}^{H \times W} \) for the region \( R_L \) defined by

\[
I_L(p) = I(y, z)/|R_L|,
\]

\[
I_L(p_{tl}) = -I(y, z)/|R_L|,
I_L(p_{br}) = I(y, z)/|R_L|.
\]

These values are summed for the regions \( R_L \) corresponding to all points \((y, z)\) in the image, to create the differential image. Finally, we integrate the differential image, and the
left/right view of the DP image pair is given by

\[
\mathbf{B}_{L,R} \text{ = } \mathcal{T}(\mathcal{I}_{L,R}),
\]

where \(\mathcal{T}(\cdot)\) denotes the integral process. The integration process [35] is also closely related to ‘summed area tables’ in graphics [5].

Our simulator allows the vision community to collect large amounts of DP data with ground truth, opening the door to accurately benchmark DP-based methods. Our simulator can also be used to supervise the learning process.

With the DP model and the simulator, we build our DDDNet considering both stereo and defocus cues.

### 4. DP-based Depth and Deblur Network

#### 4.1. Network architecture.

The input of our DDDNet is the left and right view of a DP image pair \(\mathbf{B}_{L,R}\). The output of our DDDNet is the estimated inverse depth map \(\hat{\mathbf{D}}\) and the deblurred image \(\hat{\mathbf{I}}\). We use ground-truth latent sharp image \(\mathbf{I}\) and inverse depth map \(\mathbf{D}\) for training.

The pipeline of DDDNet is shown in Fig. 5. It consists of two components: DepthNet \(g(\cdot)\) with parameters \(\hat{G}\) and DeblurNet \(f(\cdot)\) with parameters \(\hat{F}\). The DepthNet is based on [4] and the DeblurNet is based on the multi-patch network [42]. Note that our approach is independent of the choices of \(f(\cdot)\) and \(g(\cdot)\) (e.g., multi-scale and multi-patch architectures). We start with a coarse inverse depth map \(\mathbf{D}\) estimation by the DepthNet, where \(\mathbf{D} = g(\mathbf{B}_{L,R}; \hat{G})\). Then, we combine the coarse and blurred inverse depth map with \(\mathbf{B}_{L,R}\), and feed it to the DeblurNet. The DeblurNet is an encoder-decoder network, and \(\{\mathbf{I}, \mathbf{D}\} = f(\mathbf{B}_{L,R}, \mathbf{D}; \hat{F})\).

#### 4.2. Loss functions.

We use a combination of an image restoration loss, depth loss, and an image reblur loss. The final loss is a sum of the three losses.

\[
\mathcal{L} = \mathcal{L}_{\text{res}} + \mathcal{L}_{d} + \mathcal{L}_{\text{reb}},
\]

where \(\mathcal{L}_{\text{res}}\) is the image restoration loss, \(\mathcal{L}_{d}\) is the depth loss, and \(\mathcal{L}_{\text{reb}}\) is the image reblur loss.

### 4.2. Loss functions.

- **Image restoration loss** \(\mathcal{L}_{\text{res}}\). This loss ensures the deblurred image is similar to the target image, and is given by

\[
\mathcal{L}_{\text{res}} = \frac{1}{N} \sum_{y,z} \| \mathbf{I}(y,z) - \hat{\mathbf{I}}(y,z) \| ,
\]

where \(N\) is the number of pixels and \(\| \cdot \|\) is the \(\ell_2\) norm.

- **Depth loss** \(\mathcal{L}_{d}\). We adopt the smooth \(\ell_1\) loss. Smooth \(\ell_1\) loss \(\mathcal{S}(\cdot)\) is widely used in a regression task, for its robustness and low sensitivity to outliers [11]. The \(\mathcal{L}\) is defined as

\[
\mathcal{L}_{d} = \frac{1}{N} \sum_{y,z} \mathcal{S}(\hat{\mathbf{D}}(y,z) - \mathbf{D}(y,z)) .
\]

- **Image reblur loss** \(\mathcal{L}_{\text{reb}}\). The reblur loss penalizes the differences between the input blurred images and the reblurred images (from the DP simulator). The reblur loss explicitly enforces the restored image and inverse depth map to lie on the manifold of the ground-truth image and inverse depth map, subject to our DP model. The \(\mathcal{L}_{\text{reb}}\) is given by

\[
\mathcal{L}_{\text{reb}} = \frac{1}{N} \sum_{y,z} \| \mathbf{B}_{L,R}(y,z) - \hat{\mathbf{B}}_{L,R}(y,z) \| .
\]
Table 1. Quantitative analysis of deblurring results on the DPD-blur dataset [1], Our-syn dataset, and Our-real dataset. The best results are shown in bold. Here, ‘Ourswb’ and ‘Oursreb’ denotes our model trained without and with the reblur loss, respectively. The results demonstrate that our model improves significantly with the reblur loss. Besides ‘PSNR’ (in dB), to explicitly illustrate the improved quality of deblurred images, we use a metric ‘RMSE_rel’ (RMSE/255, in %) to further explain ‘what 1dB improvement means’. Compared with the second-best method, the relative improvement of our method in intensity value for the three datasets is 13%, 19%, and 12%, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>MSE_rel ↓</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>MSE_rel ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBDB [17]</td>
<td>24.82</td>
<td>0.801</td>
<td>0.826</td>
<td>32.17/33.21</td>
<td>0.948/0.956</td>
<td>0.826/0.842</td>
</tr>
<tr>
<td>DMENet [19]</td>
<td>23.93</td>
<td>0.812</td>
<td>0.862</td>
<td>31.45</td>
<td>0.939</td>
<td>0.826</td>
</tr>
<tr>
<td>DPDNet [1]</td>
<td>25.53</td>
<td>0.926</td>
<td>0.529</td>
<td>24.82</td>
<td>0.826</td>
<td>0.826</td>
</tr>
<tr>
<td>Ourswb/Ourreb</td>
<td>26.15/26.76</td>
<td>0.933/0.949</td>
<td>5.29</td>
<td>4.93/4.59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Quantitative analysis of depth estimation on the DPD-disp dataset [29]. As the DPD-disp dataset only provides testing data, we directly use our model (trained on Our-syn dataset, namely ‘Ours’) without fine-tuning and achieves the second best. Then, we use our reblur loss to fine-tune the model (without using the GT depth map). Our model with the reblur loss ‘Oursreb (fine-tune)’ achieves competitive results, indicating that our simulated DP outputs resemble those from a real DP sensor.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>MSE_rel ↓</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>MSE_rel ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours-syn</td>
<td>29.1</td>
<td>0.905</td>
<td>0.806</td>
<td>23.91/24.03</td>
<td>0.826/0.850</td>
<td>0.826/0.850</td>
</tr>
<tr>
<td>Ours-real</td>
<td>26.3</td>
<td>0.877</td>
<td>0.808</td>
<td>24.82</td>
<td>0.825</td>
<td>0.826</td>
</tr>
<tr>
<td>Ours</td>
<td>29.1</td>
<td>0.905</td>
<td>0.806</td>
<td>23.91/24.03</td>
<td>0.826/0.850</td>
<td>0.826/0.850</td>
</tr>
</tbody>
</table>

5. Experiment

5.1. Experimental setup

Real dataset. We evaluate our method on three real DP datasets, namely, the Defocus Depth estimation dataset (DPD-disp) [29], Defocus Deblur Dual-Pixel dataset (DPD-blur) [1], and our new-collected dataset (Our-real). DPD-disp dataset provides DP images with depth maps. The ground-truth (GT) depth map is computed by applying the well-established depth-from-defocus technique. DPD-blur provides a collection of image pairs captured by Canon 5D IV with all-in-focus and out-of-focus. In testing, we down-sampled the frames to 50% of their original size (1120 × 1680 pixels). Compared with the DPD-blur dataset, Our-real is collected by Canon with a variety of aperture sizes, varying from f/4 to f/22. Each all-in-focus image is associated with several out-of-focus blurred images, yielding a diversity dataset. Our-real contains 150 scenes, including both indoor and outdoor scenes, captured under a variety of lighting conditions.

Synthetic dataset. Our synthetic dataset is generated using the proposed DP simulator. We use the NYU depth dataset [24] as the input to our simulator, as it provides RGB images with depths. Giving different camera parameters, we simulate 5000 image pairs for training and 500 image pairs for testing.

Evaluation metrics. Given a DP-pair, our method jointly estimates a depth map and a deblurred image. We use standard metrics to evaluate the quality of estimated depth map and restored image separately [7, 28]. For depth map, we use absolute relative error ‘AbsRel’, square relative error ‘SqRel’, root mean square error ‘RMSE’ and its log scale ‘RMSE_log’, and the δ lower relative error of δl = 1.25i for i ∈ 1, 2, 3. For restored image, we adopt the peak signal-to-noise ratio ‘PSNR’, structural similarity ‘SSIM’, and RMSE relative error ‘RMSE_rel’. For the DPD-disp dataset, we follow DPDisp [29] to use the affine invariant version of MAE (‘AI(1)’), RMSE (‘AI(w)’), and Spearman’s rank correlation (1 − |ρs|) for evaluation.

Baseline methods. For depth map, we compare with state-of-the-art methods (BTs [20], Monodepth2 [12]), and stereo (AnyNet [37]), and DP (SDoF [36], DPDisp [29]) based depth estimation methods. For defocus deblurring, we compared with EBDB [17], DMENet [19], and DPDNet [1]. All methods are evaluated on each dataset independently. All learning methods are fine-tuned for each dataset (except DPD-disp, as no training data is provided). To fine-tune DMENet [19] on the DPD-blur and Our-real dataset, we use BTS [20] to estimate a coarse depth map to train, as the two datasets do not provide ground-truth depth maps. The same coarse depth maps are also used to train our network.

Implementation details. Our network is trained from scratch using the Adam optimizer [18] with a learning rate of 10−3 and a batch size of 10. Our model is trained on a single NVIDIA Titan XP GPU.

5.2. Results and discussions

We compare our results with baselines on depth estimation and image deblurring on 4 (including real and synthetic) datasets.

Deblurring results. We show the quantitative and qualitative comparisons for deblurring in Table 1, Fig. 7 and Fig. 8, respectively. In Table 1, we achieve competitive results compared with the state-of-the-art methods [17, 19, 1] on both DPD-blur, Our-syn, and Our-real dataset. Here, ‘Ourswb’ and ‘Oursreb’ denote results without and with the reblur loss, respectively. The relative improvement is around 10%, demonstrating that using our DP-model based reblur loss can significantly improve the deblurring perfor-
Table 3. Quantitative analysis of deblurring and depth estimation results on our synthetic dataset. The synthetic dataset is generated using our DP simulator. Here, we evaluate the intermediate outputs of DDDNet (see Fig. 5) step by step. ‘Oursb’ denotes the coarse depth map $\hat{D}_c$ from DepthNet, ‘Ourswb’ denotes the depth map $\hat{D}$ from DeblurNet, and ‘Oursreb’ denotes the final results with the reblur loss. BTS achieves lower in ‘RMSE’ but higher in ‘RMSE\_log’, and the reason is that BTS gets a number of zeros in their results.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Method</th>
<th>abs_rel</th>
<th>sq_rel</th>
<th>rmse</th>
<th>rmse_log</th>
<th>$a &lt; 1.25 \uparrow$</th>
<th>$a &lt; 1.25^2 \uparrow$</th>
<th>$a &lt; 1.25^3 \uparrow$</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monocular</td>
<td>BTS [20]</td>
<td>0.377</td>
<td>0.195</td>
<td>0.255</td>
<td>0.484</td>
<td>0.554</td>
<td>0.741</td>
<td>0.994</td>
<td>-</td>
</tr>
<tr>
<td>Stereo</td>
<td>AnyNet [37]</td>
<td>0.128</td>
<td>0.117</td>
<td>0.731</td>
<td>0.168</td>
<td>0.828</td>
<td>0.923</td>
<td>0.998</td>
<td>-</td>
</tr>
<tr>
<td>Dual-pixel</td>
<td>DPdisp [29]</td>
<td>0.328</td>
<td>0.479</td>
<td>1.252</td>
<td>0.332</td>
<td>0.438</td>
<td>0.804</td>
<td>0.965</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Oursb</td>
<td>0.149</td>
<td>0.364</td>
<td>1.222</td>
<td>0.224</td>
<td>0.743</td>
<td>0.930</td>
<td>0.978</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Ourswb</td>
<td>0.091</td>
<td>0.082</td>
<td>0.599</td>
<td>0.123</td>
<td>0.918</td>
<td>0.993</td>
<td>0.998</td>
<td>32.171</td>
</tr>
<tr>
<td></td>
<td>Oursreb</td>
<td>0.083</td>
<td>0.052</td>
<td>0.461</td>
<td>0.111</td>
<td>0.936</td>
<td>0.998</td>
<td>1.000</td>
<td>33.218</td>
</tr>
</tbody>
</table>

Figure 7. Examples of deblurring results on the real DPD-blur dataset [1] (top two rows) and our collected real DP dataset (bottom three rows). (c) The distribution of the intensity error (‘RMSE\_rel’). It shows the percentile of the number of pixels below an error ratio. The lower the curve, the better. For space limitations, we only display deblurred images from the second-best baseline [29] for comparison. (Best viewed in colour on screen.)

Performance of our DDDNet. Fig. 7 and Fig. 8 show the qualitative and quantitative comparisons on the three datasets.

**Depth results.** We first provide depth estimation results on the DPD-disp dataset. The comparisons are shown in Table 2 and Fig. 6. As no training data is available, we directly test our model (trained on Our-syn dataset) on their testing set without fine-tuning and achieve the second best performance. Furthermore, note that our DP-model based reblur loss can be used as a ‘self-supervision’ signal to adapt our DDDNet to this test dataset, we can further fine-tune our model trained on Our-syn dataset using our reblur loss, without using GT depth map. Our model with reblur loss ‘Ours (fine-tune)’ achieves competitive results compared with the state-of-the-art methods [20, 12, 36, 29].

We then provide depth comparisons on Our-syn dataset. The results are shown in Table 3 and Fig. 8. In Table 3, we evaluate the outputs of our DDDNet step by step, it shows that, with our joint optimization of depth estimation and deblurring, we improve the quality of both the depth map and the deblurred image. Also, with our DP-model based reblur
loss, we further improve our performance. For example, for
the 'abs_rel' metric, the relative error for the coarse depth
map as the output of the DepthNet (see Fig. 5), the Deblur-
Net without reblur loss and the DeblurNet with reblur loss
is 0.149, 0.091 and 0.083, respectively.

To further illustrate the superiority of our results, we re-
construct the scene to other views with the estimated sharp
image and depth map. The reconstructed videos can be
found in our project page.

**Effectiveness of our simulated DP images.** To further
show the usefulness of our simulated DP images, our model
trained on Our-syn dataset is directly used to test on the
DPD-blur dataset. The result is 20.28dB/0.650/9.68% in
PSNR/SSIM/RMSE_{rel} without any fine-tuning. Af-
ter fine-tuning, our model trained on Our-syn get re-
markable performance, with PSNR/SSIM/RMSE_{rel} at
26.92dB/0.864/4.51%, outperforming the performance of
our model (26.76dB/0.891/4.50%) directly trained on the
DPD-blur dataset. To further show the transfer ability of
our model trained using simulated DP images, we only use
a half amount of images from the DPD-blur dataset to finetune our model trained on Our-syn dataset, obtain-
ing a good performance, with PSNR/SSIM/RMSE_{rel} at
26.52dB/0.822/4.72% in 20 epoches.

**6. Conclusions**

In this paper, we derive a mathematical DP model for-
mulating the imaging process of the DP camera. We pro-
pose an end-to-end DDDNet (DP-based Depth and Deblur
Network), to jointly estimate a inverse depth map and re-
store a sharp image from a blurred DP image pair. Also,
we present a reblur loss building upon our DP model and
integrate it into our DDDNet. For future work, we plan to
extend our DDDNet to a self-supervised depth and sharp
image estimation network.

**7. Acknowledgment**

This research was supported in part by the (ARC)
fellowship and Discovery Project grant (DE180100628,
DP200102274, CE140100016, DP190102261). Liyuan Pan
would like to thank Dr. Liu Liu for proofreading the arti-
cle and constructive feedback, and Dr. Yiran Zhong for his
support when designing the DepthNet.
References


