Blurry-Edges: Photon-Limited Depth Estimation from Defocused Boundaries

Supplementary Material

5. Calculation of illuminance

To obtain the equivalent illuminance in Tab. 1, we first calculate the corresponding photon level α by solving:

$$\frac{\sqrt{\alpha+\sigma^2}}{\alpha}\times 255=\mathrm{SD}_{\mathrm{LSB}}. \tag{24}$$

Then compute the solid angle of a pixel Ω_{pix}:

$$\Omega_{\mathrm{pix}}=\frac{A_{\mathrm{pix}}}{f^2}, \tag{25}$$

where $A_{\mathrm{pix}}=\left(5.93\times10^{-6}\right)^2$ m^2 is the area of one pixel. P_{pix} is the power received by one pixel:

$$P_{\mathrm{pix}}=\frac{\alpha}{t\cdot\mathrm{QE}}\cdot\frac{hc}{\lambda_G}, \tag{26}$$

where $t=1/200$ s is the exposure time, $\lambda_G=532$ nm is the wavelength of green light, $\mathrm{QE}=0.73$ is the quantum efficiency at $\lambda_G=532$ nm, $h=6.626\times10^{-34}$ J·s is the Planck's constant, $c=3\times10^8$ m/s is the speed of light. A_{aperture} is the area of the aperture:

$$A_{\mathrm{aperture}}=\pi\Sigma^2. \tag{27}$$

$f/\#$ is the f-number:

$$f/\#=\frac{f}{2\Sigma}, \tag{28}$$

and we choose $f/5.6$ in our calculation. $K_m=683$ lm/W is the maximum possible value of photopic luminous efficacy of radiation. $V(\lambda_G)=0.83$ is the photopic luminous efficiency function at λ_G. Finally, the illuminance is computed via:

$$\begin{aligned}E_{\mathrm{lux}}=&\frac{2\pi}{\Omega_{\mathrm{pix}}}\cdot\frac{P_{\mathrm{pix}}}{A_{\mathrm{aperture}}}\\=&\frac{8\left(f/\#\right)^2}{A_{\mathrm{pix}}}\cdot\frac{\alpha}{t\cdot\mathrm{QE}}\cdot\frac{hc}{\lambda_G}\cdot K_m\cdot V(\lambda_G)\end{aligned}. \tag{29}$$

6. Further discussion on the image model

6.1. Image rendering

The captured image $I(\boldsymbol{x})$ theoretically results from Gaussian convolutions of defocus (Eq. (1)) and smooth textures smoothness (Eq. (4)), applied independently to a piecewise step function $\bar{Q}(\boldsymbol{x})$, requiring two convolutions:

$$\begin{aligned}I(\boldsymbol{x})=&\bar{Q}(\boldsymbol{x})*k\left(\boldsymbol{x},\xi\right)*k\left(\boldsymbol{x},\sigma(z)\right)\\=&\bar{Q}(\boldsymbol{x})*\left(k\left(\boldsymbol{x},\xi\right)*k\left(\boldsymbol{x},\sigma(z)\right)\right)\\=&\bar{Q}(\boldsymbol{x})*k\left(\boldsymbol{x},\sqrt{\sigma(z)^2+\xi^2}\right)\end{aligned}. \tag{30}$$

which leads to Eq. (5).

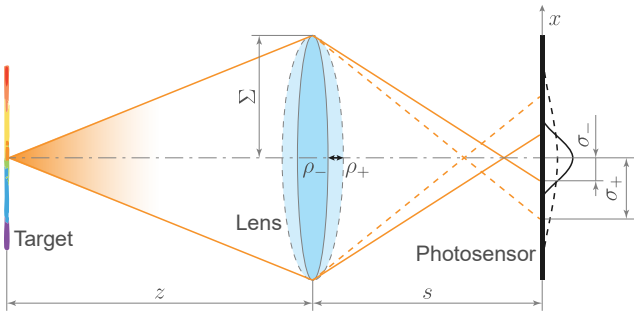

Figure 10. Image formation model. Consider a deformable lens that can change its optical power from ρ_- to ρ_+. The point spread function of a fixed target changes its width from σ_- to σ_+ according to the thin-lens law.

6.2. Image formation model

To complement the mathematical derivation of our DfD equation and visualize key terms used in depth estimation, the thin lens model with a deformable lens is shown in Fig. 10.

7. Derivation of distance maps

We first introduce $d_{it}(\boldsymbol{x};\boldsymbol{\Psi})$, $t=1,2$, the signed distance map to the starting ($t=1$) or ending ($t=2$) edge of the ith wedge (Fig. 11d-e):

$$d_{it}(\boldsymbol{x};\boldsymbol{\Psi})=\begin{cases}r_{it}(\boldsymbol{x};\boldsymbol{\Psi}) & \text{if } a_{it}(\boldsymbol{x};\boldsymbol{\Psi})\geq 0,\\ \left[2H\left(r_{it}(\boldsymbol{x};\boldsymbol{\Psi})\right)-1\right]\cdot v_{it}(\boldsymbol{x};\boldsymbol{\Psi}) & \text{otherwise,}\end{cases} \tag{31}$$

where $r_{it}(\boldsymbol{x};\boldsymbol{\Psi})$ is the signed distance map in the radial direction of the edge (Fig. 11a):

$$r_{it}(\boldsymbol{x};\boldsymbol{\Psi})=-(x-x_i)\sin(\theta_{it})+(y-y_i)\cos(\theta_{it}), \tag{32}$$

$a_{it}(\boldsymbol{x};\boldsymbol{\Psi})$ is the signed distance map in the axial direction of the edge (Fig. 11b):

$$a_{it}(\boldsymbol{x};\boldsymbol{\Psi})=(x-x_i)\cos(\theta_{it})+(y-y_i)\sin(\theta_{it}), \tag{33}$$

and $v_{it}(\boldsymbol{x};\boldsymbol{\Psi})$ is the unsigned scaled distance map to the vertex $\boldsymbol{p}_i$ (Fig. 11c):

$$v_{it}(\boldsymbol{x};\boldsymbol{\Psi})=\sqrt{\left(r_{it}(\boldsymbol{x};\boldsymbol{\Psi})\right)^2+w^2\cdot\left(a_{it}(\boldsymbol{x};\boldsymbol{\Psi})\right)^2}, \tag{34}$$

where w is a scale factor.

Then the signed distance map of the ith wedge $d_i(\boldsymbol{x};\boldsymbol{\Psi})$ is computed via:

$$d_i(\boldsymbol{x};\boldsymbol{\Psi})=\min\left(|d_{i1}(\boldsymbol{x};\boldsymbol{\Psi})|,|d_{i2}(\boldsymbol{x};\boldsymbol{\Psi})|\right)\cdot\chi_i(\boldsymbol{x};\boldsymbol{\Psi}), \tag{35}$$

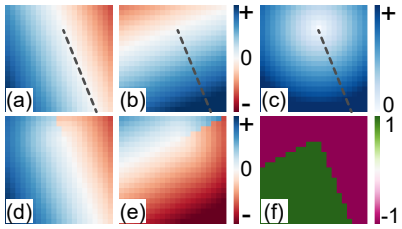

Figure 11. Additional visualizations from the sample Blurry-Edges representation in Fig. 3. (a-b) The signed distance maps in the radial and axial directions to the starting edge of the bottom wedge, $r_{11}(\boldsymbol{x};\boldsymbol{\Psi})$ and $a_{11}(\boldsymbol{x};\boldsymbol{\Psi})$, respectively. The location of the related boundary is noted as the dotted line. (c) The unsigned scaled distance map to the vertex $\boldsymbol{p}_1$, $v_{11}(\boldsymbol{x};\boldsymbol{\Psi})$, in this sample $w=1$. The location of the related boundary is noted as the dotted line. (d-e) The signed distance map to the starting and ending edges of the bottom wedge, $d_{11}(\boldsymbol{x};\boldsymbol{\Psi})$ and $d_{12}(\boldsymbol{x};\boldsymbol{\Psi})$, where $d_{11}(\boldsymbol{x};\boldsymbol{\Psi})$ is calculated through the map in (a-c). (f) The indicator function of the bottom wedge, $\chi_1(\boldsymbol{x};\boldsymbol{\Psi})$.

where $\chi_i(\boldsymbol{x};\boldsymbol{\Psi})$ is a indicator function indicating whether pixel $\boldsymbol{x}$ is inside of the ith wedge:

$$\chi_i(\boldsymbol{x};\boldsymbol{\Psi})=\begin{cases}1 & \text{if } \theta_{i1}\le \arctan 2(\boldsymbol{x}-\boldsymbol{p}_i)\le\theta_{i2},\\ -1 & \text{otherwise.}\end{cases} \tag{36}$$

Finally, the unsigned distance map $u(\boldsymbol{x};\boldsymbol{\Psi})$, mentioned in Eq. (9), is calculated by:

$$\begin{aligned}u(\boldsymbol{x};\boldsymbol{\Psi}) = \min\big(&|d_i(\boldsymbol{x};\boldsymbol{\Psi})|,\\ &|d_{i+1}(\boldsymbol{x};\boldsymbol{\Psi})|,\cdots,|d_l(\boldsymbol{x};\boldsymbol{\Psi})|\big),\\ &\text{when } M_i(\boldsymbol{x})=1,\end{aligned} \tag{37}$$

where $M_i(\boldsymbol{x})=1$ is the mask for the unoccluded ith wedge:

$$M_i(\boldsymbol{x})=H(d_i(\boldsymbol{x};\boldsymbol{\Psi}))\prod_{j=i+1}^{l}\left[1-H(d_j(\boldsymbol{x};\boldsymbol{\Psi}))\right]. \tag{38}$$

8. Details of implementation

8.1. Local stage architecture

Table 4 lists the hyperparameters of the convolutional neural network (CNN) of the local stage shown in Fig. 4. We adopt the Smish function as the activation function for each layer [S1]:

$$\text{Smish}(x)=x\cdot\tanh\left[\ln(1+\text{sigmoid}(x))\right], \tag{39}$$

which we find to be more stable and accurate in our experiments.

Layer	Specification	Output
Conv2d	$7\times7,64$	$21\times21\times64$
MaxPool2d	3×3	$11\times11\times64$
ResBlock	$\begin{bmatrix}3\times3,96\\3\times3,96\end{bmatrix}$	$11\times11\times96$
MaxPool2d	3×3	$6\times6\times96$
ResBlock	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}$	$6\times6\times256$
ResBlock	$\begin{bmatrix}3\times3,384\\3\times3,384\end{bmatrix}$	$6\times6\times386$
ResBlock	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}$	$6\times6\times256$
MaxPool2d	2×2	$3\times3\times256$
Linear	-	1024
Linear	-	10

Table 4. CNN architecture of the local stage.

Item	Value
Dimension of each feature vector	128
Number of sub-encoder-layers	8
Number of heads in multi-head attention	8
Dimension of the feedforward network model	256

Table 5. Transformer Encoder details in the global stage. It takes in the Blurry-Edges representation parameters and does not have access to the input image pair in the inference.

8.2. Global stage architecture

Table 5 lists the architecture of the Transformer Encoder of the global stage. In this stage, each parameter pair, $\boldsymbol{\Psi}^{\boldsymbol{m}}_{\pm}$, is projected into a feature vector $\boldsymbol{v}^{\boldsymbol{m}}\in\mathbb{R}^v$ and added with a positional encoding vector $\boldsymbol{E}^{\boldsymbol{m}}=\begin{bmatrix}E^{(m,n,1)} & \cdots & E^{(m,n,v)}\end{bmatrix}^\top\in\mathbb{R}^v$. The 2D positional encoding vector follows the design by Zhang and Liu [S2]:

$$E^{(m,n,q)}=\begin{cases}\sin\left(\frac{m}{10000^{\frac{2q}{v}}}\right) & \text{if } q \text{ is even and } q\le\frac{v}{2},\\ \cos\left(\frac{m}{10000^{\frac{2q+2}{v}}}\right) & \text{if } q \text{ is odd and } q\le\frac{v}{2},\\ \sin\left(\frac{n}{10000^{\frac{2q}{v}-1}}\right) & \text{if } q \text{ is even and } q>\frac{v}{2},\\ \cos\left(\frac{n}{10000^{\frac{2q+2}{v}-1}}\right) & \text{if } q \text{ is odd and } q>\frac{v}{2}.\end{cases} \tag{40}$$

8.3. Loss functions for local and global stages training

As shown in Eqs. (21) and (22), we use comprehensive loss functions to train the CNN of the local stage and the Transformer Encoder of the global stage, respectively. For local stage training, three terms l_{1-3} regularize the Blurry-Edges

prediction as following:

$$
\begin{aligned}
l_1 &= \left\| c\left(\boldsymbol{x}; \boldsymbol{\Psi}_{\pm}^{\boldsymbol{m}}\right) - P_{\text{clean},\pm}^{\boldsymbol{m}}\left(\boldsymbol{x}\right) \right\|^2 && \text{(color error)}, \\
l_2 &= \left\| c'\left(\boldsymbol{x}; \boldsymbol{\Psi}_{\pm}^{\boldsymbol{m}}\right) - P_{\text{clean},\pm}^{\boldsymbol{m},\prime}\left(\boldsymbol{x}\right) \right\|^2 && \text{(smoothness error)}, \\
l_3 &= b\left(\boldsymbol{x}; \boldsymbol{\Psi}_{\pm}^{\boldsymbol{m}}, \delta\right) \cdot u^{\boldsymbol{m}}\left(\boldsymbol{x}\right) && \text{(boundary localization)},
\end{aligned} \tag{41}
$$

where the terms $P_{\text{clean},\pm}^{\boldsymbol{m}}\left(\boldsymbol{x}\right)$ and $P_{\text{clean},\pm}^{\boldsymbol{m},\prime}\left(\boldsymbol{x}\right)$ indicates the noiseless image patch and its derivative map from Sobel filtering, and $u^{\boldsymbol{m}}$ represents the unsigned distance map to the nearest true boundaries in the patch. For global stage training, seven terms g_{1-7} comprehensively penalize the Blurry-Edges prediction as detailed below:

$$
\begin{aligned}
g_1 &= \left\| c\left(\boldsymbol{x}; \{\boldsymbol{\Omega}^{\boldsymbol{m}}, \boldsymbol{\eta}^{\boldsymbol{m}}\}\right) - P_{\text{clean}}^{\boldsymbol{m}}\left(\boldsymbol{x}\right) \right\|^2 && \text{(color error)}, \\
g_2 &= \left\| c\left(\boldsymbol{x}; \{\boldsymbol{\Omega}^{\boldsymbol{m}}, \boldsymbol{\eta}^{\boldsymbol{m}}\}\right) - C^{\boldsymbol{m}}\left(\boldsymbol{x}\right) \right\|^2 && \text{(color consistency)}, \\
g_3 &= \left\| b\left(\boldsymbol{x}; \boldsymbol{\Omega}^{\boldsymbol{m}}, \delta\right) - B^{\boldsymbol{m}}\left(\boldsymbol{x}\right) \right\|^2 && \text{(boundary consistency)}, \\
g_4 &= \left\| c'\left(\boldsymbol{x}; \{\boldsymbol{\Omega}^{\boldsymbol{m}}, \boldsymbol{\eta}^{\boldsymbol{m}}\}\right) - P_{\text{clean}}^{\boldsymbol{m},\prime}\left(\boldsymbol{x}\right) \right\|^2 && \text{(smoothness error)}, \\
g_5 &= \left\| c'\left(\boldsymbol{x}; \{\boldsymbol{\Omega}^{\boldsymbol{m}}, \boldsymbol{\eta}^{\boldsymbol{m}}\}\right) - C^{\boldsymbol{m},\prime}\left(\boldsymbol{x}\right) \right\|^2 && \text{(smoothness consistency)}, \\
g_6 &= b\left(\boldsymbol{x}; \boldsymbol{\Omega}^{\boldsymbol{m}}, \delta\right) \cdot u^{\boldsymbol{m}}\left(\boldsymbol{x}\right) && \text{(boundary localization)}, \\
g_7 &= \left\| \sum_{i=1}^{l} H\left(b_i\left(\boldsymbol{x}; \boldsymbol{\Omega}^{\boldsymbol{m}}, \delta\right) - \tau\right) \cdot z_i^{\boldsymbol{m}} - Z^{*,\boldsymbol{m}}\left(\boldsymbol{x}\right) \right\|^2 && \text{(depth error)},
\end{aligned} \tag{42}
$$

where $Z^{*,\boldsymbol{m}}\left(\boldsymbol{x}\right)$ denotes the ground truth depth map of the patch.

8.4. Weight scheduling

We observe that dynamically varying the weights of each term in the loss functions, β_{1-3} and γ_{1-7} (Eq. (21) and Eq. (22) in the main paper), benefit the convergence of both the local and global stages. In addition to the parameters listed in Sec. 4.1, we list the full details of the training parameters here.

We use $\lambda = 5 \times 10^{-3}$ in Eq. (14) and $w = 1$ in Eq. (34) for both local and global stages. We apply dynamic loss function weights in Eq. (21) and Eq. (22) for local and global stages respectively. This strategy improves the stability and accuracy of the training. For each epoch, the weights are updated using linear interpolation, and the weight values are shown in Fig. 12 and Fig. 13.

8.5. Processing large images using blocks

For large images, we divide them into blocks, as shown in Fig. 14. After estimating the depth for the patches within each block, the model combines them to generate the final depth maps. The number of blocks n_b is calculated via:

$$
n_b = \left\lceil \frac{l_I - l_b}{s_b} + 1 \right\rceil, \tag{43}
$$

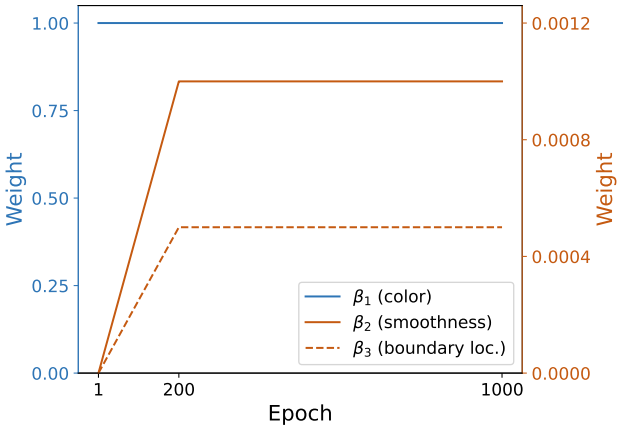

Figure 12. Weight scheduling in the local stage training. There is one dynamic phase in the beginning. β_2 (for smoothness error) and β_3 (for boundary localization error) gradually increase to the final values in the first 200 epochs.

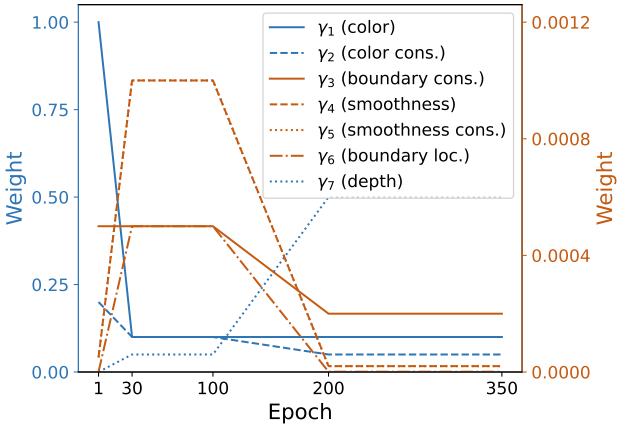

Figure 13. Weight scheduling in the global stage training. There are two dynamic phases, 1–30 and 100–200 epochs individually. In the beginning, γ_1 (for color error) and γ_2 (for color consistency error) dominate the loss function, and γ_3 (for boundary consistency error) regularizes the consistency more at this phase. Then γ_4 (for smoothness error), γ_5 (for smoothness consistency error, overlapped with γ_4), and γ_6 (for boundary localization error) increase to refine the global color and boundary map. Finally, γ_7 (for depth error) leads the loss function to penalize the depth estimation.

where l_I is the image side length, l_b is the block side length, and s_b is the block stride that is obtained through:

$$
s_b = l_b - l_p + \left(1 - 2n_p\right) s_p, \tag{44}
$$

where l_p is the patch side length, n_p is the number of marginal patches removed along the side length dimension, and s_p is the patch stride. When $\left(2n_p + 1\right) s_p > l_p$, the overlapped areas of blocks ensure that all patches are op-

timized with respect to all neighboring patches, mitigating the discontinuities between blocks.

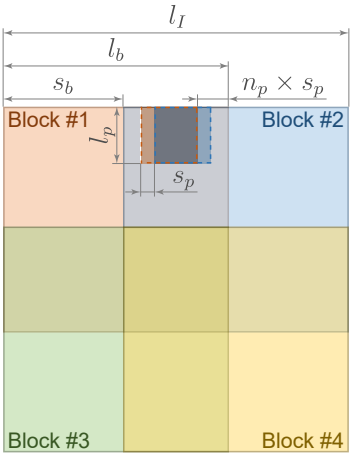

Figure 14. Performing on large image using blocks. In this example, there are 2×2 blocks to cover the image. The final upper-right patch of orange block and the final upper-left patch of the blue block are indicated by the dark orange square and dark blue square, respectively

9. Improvement on DEReD training

We observe that DEReD [32] struggles to be trained with the original loss function due to the lack of textures in our basic shape training set. To address this, we add two regularization terms: one for the ℓ^2 norm of the depth map and the other for the ℓ^2 norm of the depth map's first derivative.

10. Additional results

10.1. Depth estimation metrics

Here, we provide the computation of each metric used to quantify the depth estimation accuracy in the main paper.

Given an estimated depth map $Z(\boldsymbol{x})$ and the ground truth depth map $Z^*(\boldsymbol{x})$, the RMSE and AbsRel metrics are calculated via:

$$\text{RMSE} = \sqrt{\frac{\sum_{\boldsymbol{x}} \|Z(\boldsymbol{x}) - Z^*(\boldsymbol{x})\|^2}{\left|\hat{Z}(\boldsymbol{x})\right|}} \tag{45}$$

and

$$\text{AbsRel} = \frac{\sum_{\boldsymbol{x}} \frac{\|Z(\boldsymbol{x}) - Z^*(\boldsymbol{x})\|}{Z^*(\boldsymbol{x})}}{\left|\hat{Z}(\boldsymbol{x})\right|}, \tag{46}$$

where $|\cdot|$ is the cardinality operator returning the number of pixels. For the δ-threshold, we use the normalized maps of $Z(\boldsymbol{x})$ and $Z^*(\boldsymbol{x})$ with the range $[0, 1]$, $\hat{Z}(\boldsymbol{x})$ and $\hat{Z}^*(\boldsymbol{x})$ respectively:

$$\begin{aligned}
\hat{Z}(\boldsymbol{x}) &= \frac{Z(\boldsymbol{x}) - Z_{\min}}{Z_{\max} - Z_{\min}}, \\
\hat{Z}^*(\boldsymbol{x}) &= \frac{Z^*(\boldsymbol{x}) - Z_{\min}}{Z_{\max} - Z_{\min}}, \\
\delta i &= \frac{\left|\max\left(\frac{\hat{Z}(\boldsymbol{x})}{\hat{Z}^*(\boldsymbol{x})}, \frac{\hat{Z}^*(\boldsymbol{x})}{\hat{Z}(\boldsymbol{x})}\right) < \tau_n^i\right|}{\left|\hat{Z}(\boldsymbol{x})\right|}, i = 1, 2, 3,
\end{aligned} \tag{47}$$

where $Z_{\min}$ and $Z_{\max}$ are the minimum and maximum of the working range, and $\tau_n = 1.25$ is the threshold.

10.2. Ablation study on weight scheduling

We conducted an ablation study on different variations of weights in the loss function for the training of the global stage. Based on the configuration in Fig. 13, we mainly vary the scheduling of γ_2 (corresponding to color consistency error) and γ_7 (corresponding to depth error) to explore whether increasing the weights of the depth error can improve the performance, as shown in Tab. 6. The results are in Tab. 7. As our task requires accurate depth prediction at the correct boundary positions, a balanced weight between the depth error and the boundary error (Config 1) leads to the highest performance.

Key epoch	Config 1 (base)		Config 2		Config 3	
	γ_2	γ_7	γ_2	γ_7	γ_2	γ_7
1	0.2	0.0001	0.2	0.0001	0.1	0.0001
30	0.1	0.05	0.1	0.1	0.05	0.1
100	0.05	0.5	0.05	0.8	0.02	0.8
350	0.05	0.5	0.05	0.8	0.02	0.8

Table 6. Configurations of different weights for the ablation study. *Config 1* is the base configuration in Fig. 13, and *Config 2* increases the weight for the depth error, while *Config 3* additionally decreases the weight for the color consistency error.

Config	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$	RMSE (cm)↓	AbsRel (cm)↓
1	**0.720**	**0.840**	**0.895**	**5.281**	**3.295**
2	0.680	0.820	0.882	5.702	3.657
3	0.671	0.823	0.884	5.768	3.797

Table 7. Depth estimation accuracy for different configurations on the synthetic testing set. *Config 1* leads to the highest accuracy on all metrics.

10.3. Evaluation with sparse masks

In the main paper, we evaluate PhaseCam3D [44], DefocusNet [23], DFV-DFF [47], and DEReD [32] with dense

	Method	Venue'Year	# images	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$	RMSE (cm) $\downarrow$	AbsRel (cm) $\downarrow$
Sparse	PhaseCam3D [44]	ICCP'2019	2	0.408	0.669	0.805	9.115	7.715
	DefocusNet [23]	CVPR'2020	5	0.633	0.821	0.894	6.132	4.707
	DFV-DFF [47]	CVPR'2022	5	0.486	0.747	0.862	8.312	6.815
	DEReD [32]	CVPR'2023	5	0.508	0.754	0.859	7.799	6.214

Table 8. Depth prediction accuracy of competing learn-based algorithms on the synthetic testing set after applying the same mask as ours for sparse depth map evaluation. According to Tab. 3, our method achieves the highest performance.

	Method	Venue'Year	# images	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$	RMSE (cm) $\downarrow$	AbsRel (cm) $\downarrow$
Sparse	Focal Track [8]	ICCV'2017	2	0.503	0.709	0.857	7.794	6.103
	Tang *et al*. [38]	CVPR'2017	2	**0.726**	0.832	**0.902**	5.991	3.537
	Ours	-	2	0.714	**0.836**	0.886	**5.007**	**3.255**

Table 9. Depth prediction accuracy on synthetic large image testing set. Only competing methods capable of directly handling large images are shown. Our model adopts 3×3 blocks for this test.

depth maps. For fairness, we re-evaluate these algorithms by masking out the same pixels as ours. See numbers in Tab. 8. Compared with the results of *Ours* in Tab. 3, our method universally performs the best.

10.4. Additional results on synthetic images

Figure 15 shows additional results on synthetic testing set. Compared to other algorithms, our method predicts the depth more accurately, especially when the images have abundant textures.

10.5. Additional results on synthetic large images

We evaluate our model on synthetic large images using 3×3 blocks. The quantitative comparison is shown in Table 9, where Focal Track [8] and Tang *et al*. [38] can process large images directly. Figure 16 presents a qualitative comparison on synthetic large images. Compared to other algorithms, our method predicts the depth more accurately.

10.6. Additional results on real-world images

We build a prototype camera system that is similar to the setup in Guo *et al*. [8], including an Optotune EL-16-40-TC-VIS-5D-C to change the optical power and a FLIR Grasshopper 3 GS3-U3-23S6C-C camera.

As the optical parameters of the physical system are different from the ones set for the synthetic data, we perform a linear mapping to calibrate depth estimation:

$$Z_{\text{output}} = \omega_0 + \omega_1 Z, \tag{48}$$

where Z is the predicted depth value from our model, and Z_{output} is the calibrated depth value that should match the actual object depths. We determine the parameters ω_0 and ω_1 using the following approach. We use a linear slide mounted with a front-parallel texture pad, as shown in Fig. 17. By moving the texture pad to different true distances Z^*, we obtain a series of mapping from the mean predicted depth value from our model $\{\bar{Z}_i\}$ to the corresponding true depth $\{Z_i^*\}$. We solve the following linear regression problem:

$$\begin{bmatrix} \vdots \\ Z_i^* \\ \vdots \end{bmatrix} = \omega_0 + \omega_1 \begin{bmatrix} \vdots \\ \bar{Z}_i \\ \vdots \end{bmatrix}. \tag{49}$$

The same mapping is applied to all other methods.

We collect real data in an indoor environment with normal illumination, using two shutter speeds, 10011.37 µs and 4395.25 µs, denoted by *SS*+ and *SS*++. The results in Fig. 9 are with the shutter speed *SS*++. Additional depth estimation results on real captured data are shown in Fig. 18. Our method achieves the best performance in almost all scenes. Although Tang *et al*. [38] performs slightly better than ours in the first scene at *SS*+, ours outperforms it at *SS*++ and better retains the image structure.

11. Densification of depth maps

11.1. Dense depth maps from Blurry-Edges

The proposed method aims to generate sparse depth maps along boundaries. However, we also experiment with generating dense depth maps by utilizing the Blurry-Edges representation. We compute a dense depth map using the following equation:

$$Z(\boldsymbol{x}) = \frac{\sum_{P_{\pm}^{\boldsymbol{m}} \ni \boldsymbol{x}} \sum_{i=1}^{l} M_i(\boldsymbol{x}) \cdot z_i^{\boldsymbol{m}}}{\sum_{P_{\pm}^{\boldsymbol{m}} \ni \boldsymbol{x}} \sum_{i=1}^{l} M_i(\boldsymbol{x})}. \tag{50}$$

We retrain the same global stage architecture for dense depth maps generation with an additional loss term g_8 in the

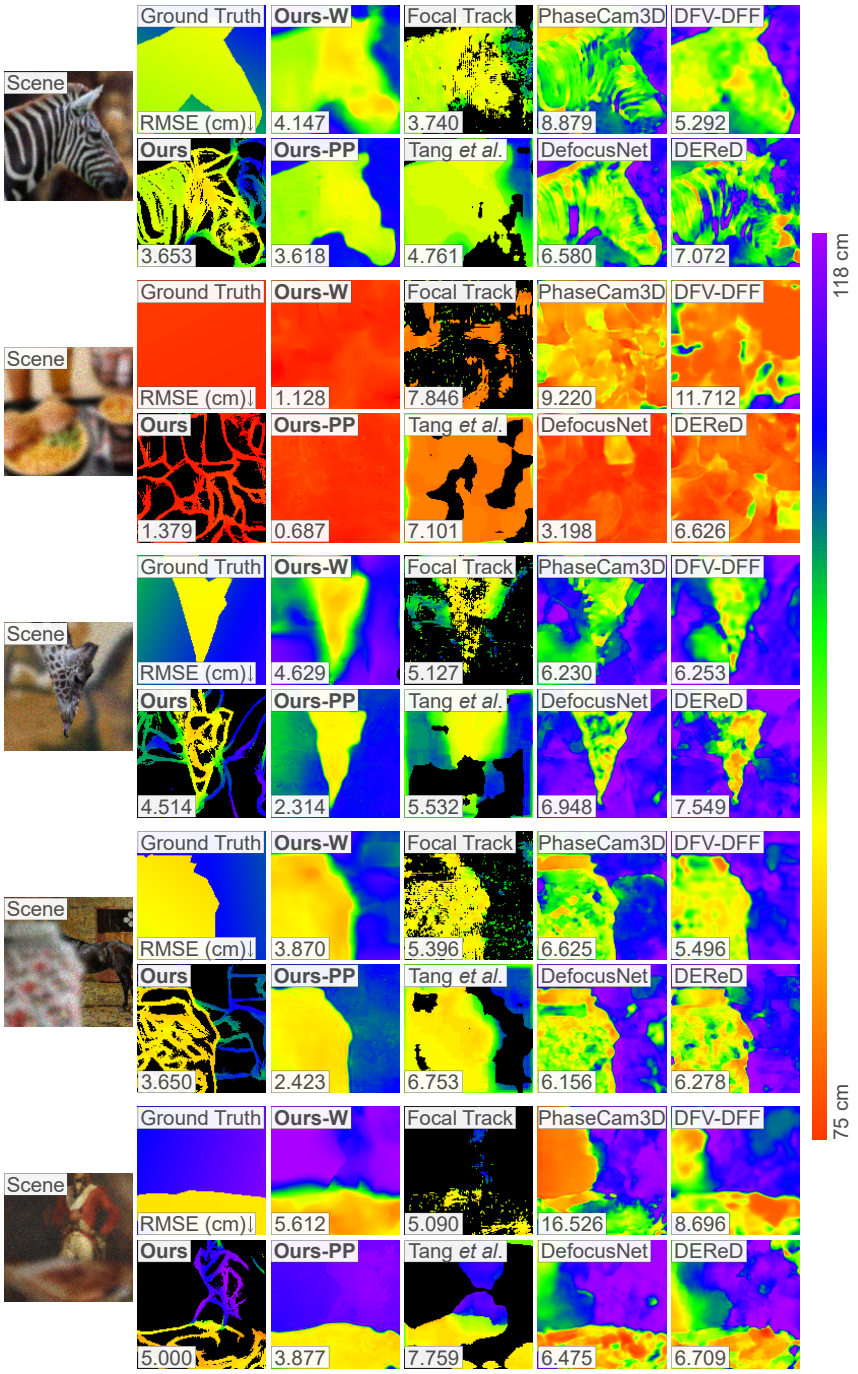

Figure 15. Depth estimation on the synthetic testing set.

objective function Eq. (22) with the weight γ_8:

$$g_8 = \left\| \sum_{i=1}^{l} M_i(\boldsymbol{x}) \cdot z_i^{\boldsymbol{m}} - Z^{*,\boldsymbol{m}}(\boldsymbol{x}) \right\|^2. \quad (51)$$

The loss function penalizes the prediction error of the dense depth map compared to the ground truth. According to Fig. 15, the visual quality of the dense depth map depends on the distribution of boundaries in the images; the denser the boundaries are, the higher the quality of the depth maps.

11.2. Dense depth maps through post-processing

We use a U-Net [30] as a densifier that takes in the sparse depth maps and outputs the dense depth maps. For the training, the loss function optimizes the parameters through ℓ^2

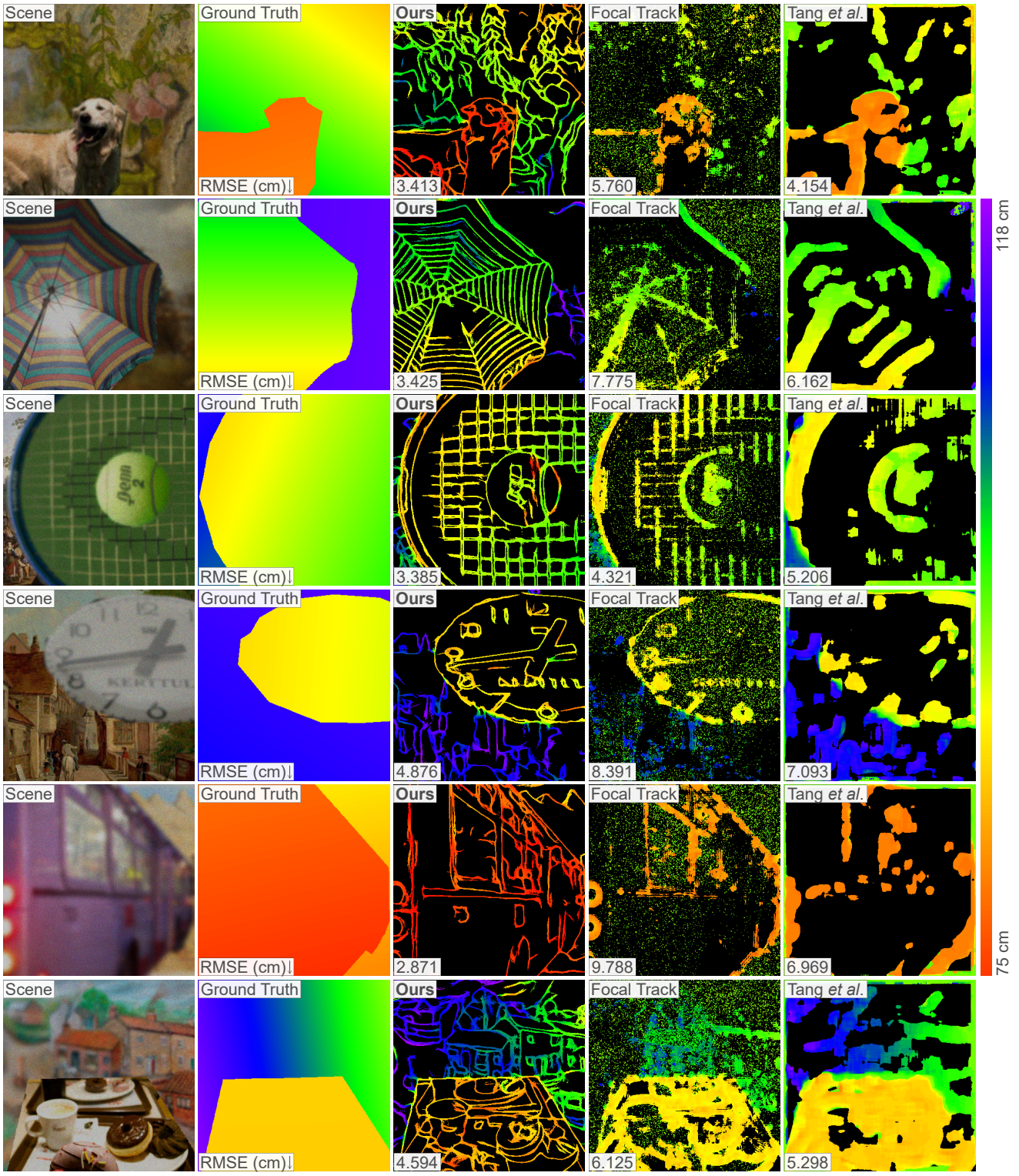

Figure 16. Depth estimation on synthetic large images. Our method performs the best visual quality.

norm on the depth map and ℓ^2 norm on its first derivative:

$$\begin{aligned}\mathcal{L}_{\text{densify}} =& \nu_1 \mathbb{E}_Z \left(\left\| Z\left(\boldsymbol{x}\right) - Z^*\left(\boldsymbol{x}\right) \right\|^2 \right) \\ &+ \nu_2 \mathbb{E}_Z \left(\left\| Z'\left(\boldsymbol{x}\right) - Z^{*,\prime}\left(\boldsymbol{x}\right) \right\|^2 \right),\end{aligned} \tag{52}$$

where $\mathbb{E}_Z$ denotes the expectation over all dense depth maps, and ν_i's are the weight coefficients. Figure 15 shows that the sparse depth map by our method is a sufficient output, and can be easily densified.

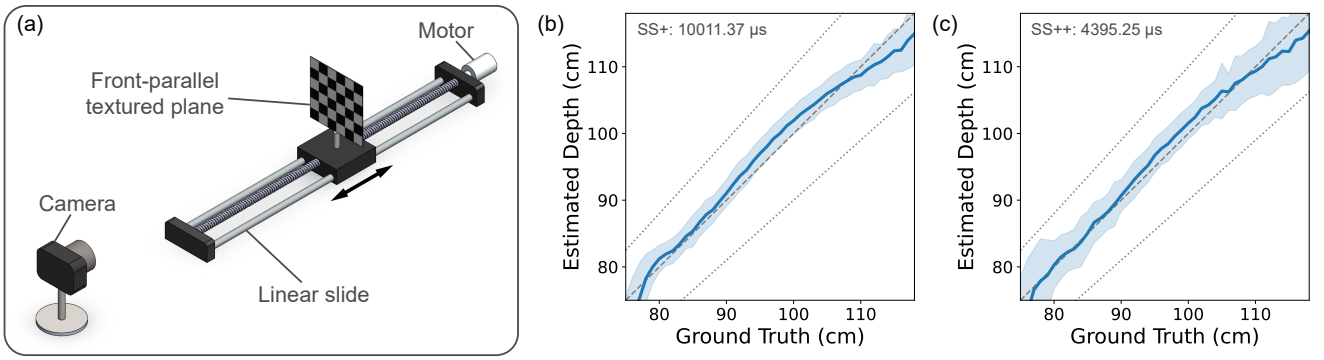

Figure 17. Calibration of the real system. (a) Calibration setup. We place front-parallel textured planes accurately at known distances using a linear slide to collect input images with ground truth depth values. (b-c) Depth prediction accuracy after calibration at shutter speeds, *SS+* and *SS++*. The blue curve shows the mean predicted depth for each true depth. The vertical width of the color band indicates the standard deviation of the depth prediction error, which remains within 10% of the true depth (dotted line) across the working range.

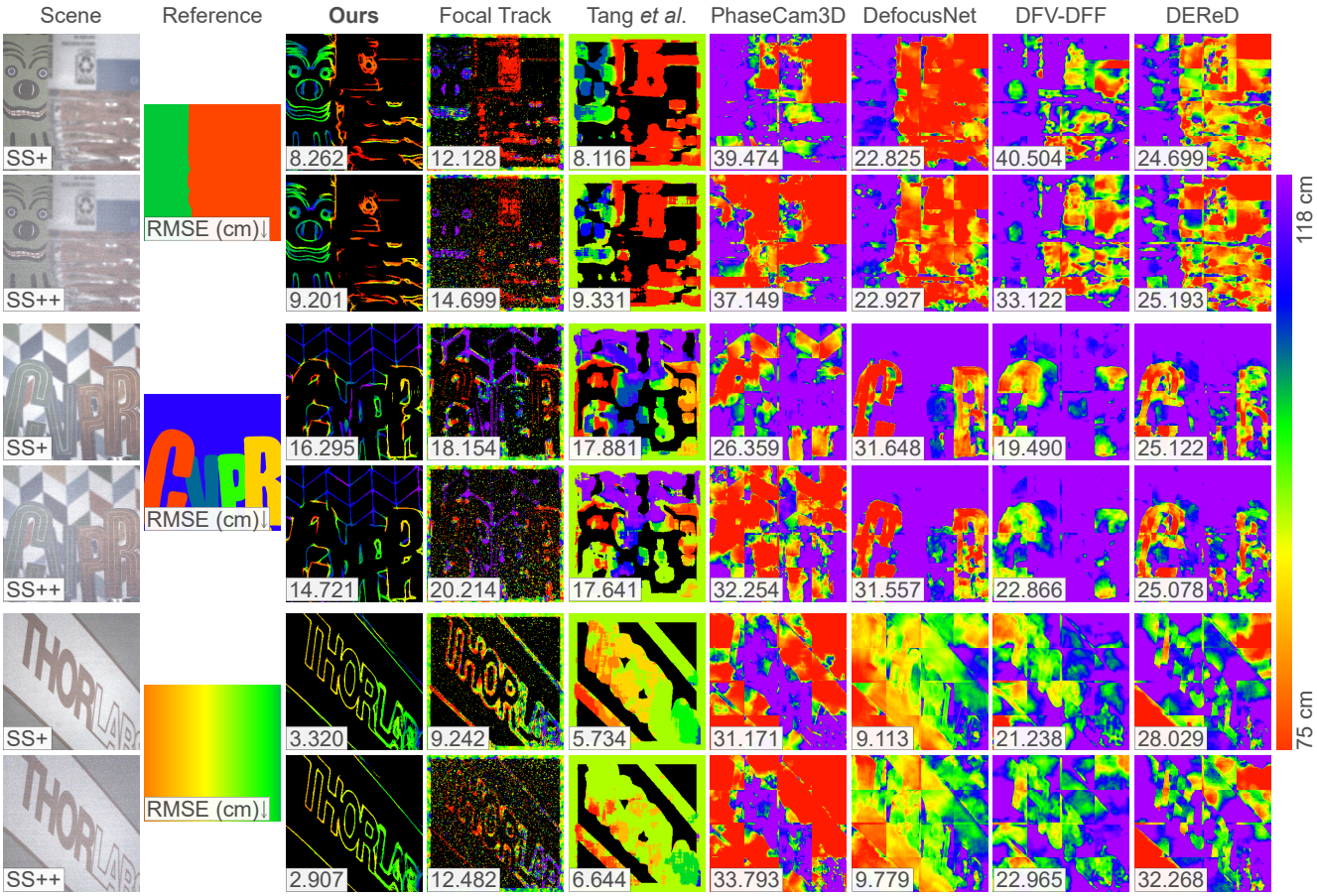

Figure 18. Depth predictions from real-world images. Our method outperforms others in nearly all scenes. While Tang *et al*. [38] slightly surpasses ours in the first scene at *SS+*, our method performs better at *SS++* and maintains the image structure more effectively, proving the robustness to high noise.

References

[S1] Xueliang Wang, Honge Ren, and Achuan Wang. Smish: A novel activation function for deep learning methods. *Electronics*, 11(4):540, 2022. 2

[S2] Zelun Wang and Jyh-Charn Liu. Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition (IJDAR)*, 24(1):63–75, 2021. 2