PlaneDepth: Self-supervised Depth Estimation via Orthogonal Planes

Supplementary Material

A. Introduction

We first provide additional details of our method in Appendix B, specifically regarding the resizing cropping transformation and the self-distillation loss. Furthermore, we provide implementation details in Appendix C. Additionally, we report more experimental results and examine the influence of different training strategies in Appendix D.

B. Additional Method Details

B.1. Resizing Cropping Transformation

Resizing and cropping data augmentation is expected to only modify the camera intrinsics while keeping the world coordinates unchanged. However, the networks must predict the depth and relative pose of the original images for all augmented inputs, which results in the disruption of a crucial monocular cue - the closer an object is, the lager its relative size [1]. We therefore assume that all input images are captured by the same camera system, and discuss the effect of the resizing cropping augmentation on the world coordinates.

Figure 1. Image is cropped at coordinates (p_x, p_y) using a scale factor of f_s, and then resized to the original resolution.

Given the center pixel coordinates $\mathbf{P}_c = (p_x, p_y)$ and the cropping scale factor f_s, the pixel coordinates $\mathbf{P} = (x, y)$ and depth D of the original image after resizing and cropping are modified as follows:

$$\widetilde{\mathbf{P}} = \frac{\mathbf{P} - \mathbf{P}_c}{f_s} + \frac{\mathbf{S}}{2} \qquad \widetilde{D} = f_s D, \tag{1}$$

where $\mathbf{S} = (W, H)$ is the size of the image. The adjustment in depth is based on the assumption that when the relative size of an object increases by a factor of f_s, its depth decreases by the same factor, f_s.

Hence, the rectified coordinates $\tilde{\mathbf{w}}$ obtained from augmented images and the original world coordinates w can be expressed as:

$$\tilde{\mathbf{w}} = \widetilde{D}\mathbf{K}^{-1}\begin{bmatrix}\tilde{x}\\ \tilde{y}\\ 1\end{bmatrix} \qquad \mathbf{w} = D\mathbf{K}^{-1}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}. \tag{2}$$

The transformation $\mathbf{R_C}$ of world coordinates from $\mathbf{w}$ to $\tilde{\mathbf{w}}$ is given by:

$$\tilde{\mathbf{w}} = \mathbf{R_C}\mathbf{w} \qquad \mathbf{R_C} = \begin{bmatrix}1 & & \frac{c_x - p_x}{f_x}\\ & 1 & \frac{c_y - p_y}{f_y}\\ & & f_s\end{bmatrix}, \tag{3}$$

where (f_x, f_y) represents the focal length and (c_x, c_y) denotes the principal point of the camera.

Given the equations of the transformed plane and the original plane:

$$\tilde{\mathbf{n}}^T\mathbf{R_C}\mathbf{w} - \tilde{\delta} = 0 \qquad \mathbf{n}^T\mathbf{w} - \delta = 0, \tag{4}$$

the rectified normal $\tilde{\mathbf{n}}$ and distance $\tilde{\delta}$ can be computed as:

$$\tilde{\mathbf{n}} = \frac{{\mathbf{R_C}}^{-T}}{||{\mathbf{R_C}}^{-T}\mathbf{n}||}\mathbf{n} \qquad \tilde{\delta} = \frac{\delta}{||{\mathbf{R_C}}^{-T}\mathbf{n}||}. \tag{5}$$

Therefore, given the original vertical plane normal $\mathbf{n}^v = [0\ 0\ 1]^T$ and ground plane normal $\mathbf{n}^g = [0\ 1\ 0]^T$, the rectified vertical plane normal $\tilde{\mathbf{n}}^v$ and ground plane normal $\tilde{\mathbf{n}}^g$ are:

$$\tilde{\mathbf{n}}^v = \frac{{\mathbf{R_C}}^{-T}\mathbf{n}^v}{||{\mathbf{R_C}}^{-T}\mathbf{n}||} = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{6}$$

$$\tilde{\mathbf{n}}^g = \frac{{\mathbf{R_C}}^{-T}\mathbf{n}^g}{||{\mathbf{R_C}}^{-T}\mathbf{n}||} = \frac{1}{\sqrt{1 + (\frac{p_y - c_y}{f_y f_s})^2}}\begin{bmatrix}0\\1\\\frac{p_y - c_y}{f_y f_s}\end{bmatrix}. \tag{7}$$

Consequently, the normal of the planes $\tilde{\mathbf{n}}^v$ and $\tilde{\mathbf{n}}^g$ after transformation are no longer orthogonal.

Furthermore, two views which are resized and cropped by the same parameters have the relationship as:

$$\mathbf{R_C}\mathbf{w}^r = \widetilde{\mathbf{R}}\mathbf{R_C}\mathbf{w} + \widetilde{\mathbf{t}} \qquad \mathbf{w}^r = \mathbf{R}\mathbf{w} + \mathbf{t}, \tag{8}$$

where $\mathbf{R}$ and $\mathbf{t}$ denote the rotation and translation, respectively. Therefore, the modified relative pose can be expressed as:

$$\widetilde{\mathbf{R}} = \mathbf{R_C}\mathbf{R}{\mathbf{R_C}}^{-1} \qquad \tilde{\mathbf{t}} = \mathbf{R_C}\mathbf{t} \tag{9}$$

B.2. Self-distillation Loss

Taking only the left view as input, we now provide the details of filtering the loss of the occluded pixels in the right view in self-distillation strategy.

Similar to [6], we obtain the right view occlusion mask $\mathbf{M}^{\mathrm{R}}$ by using:

$$\mathbf{M}^{\mathrm{R}} = \min(\sum_{i=0}^{N-1} \mathcal{W}_{\mathrm{L}\to\mathrm{R}}(\pi_i, d_i), 1), \tag{10}$$

where $\mathcal{W}(\cdot, d)$ represents the warping function using disparity d, and π_i denotes the predicted Laplace weight of the i-th plane. We define the occlusion-aware mixture Laplace loss and perceptual loss as:

$$\mathcal{L}_{\text{MLL}}^{\text{M}} = \mathbf{M}^{\text{R}} \odot (-\log \sum_{i=0}^{N-1} \frac{\hat{\pi}_i e^{\frac{-\frac{1}{3}||\mathbf{I}_r - \hat{\mathbf{I}}_i||_1}{\hat{\sigma}_i}}}{2\hat{\sigma}_i}) \tag{11}$$

$$\mathcal{L}_{\text{pc}}^{\text{M}} = ||\phi_l(\mathbf{I}_r) - \phi_l(\mathbf{M}^R \odot \hat{\mathbf{I}}_r + (1 - \mathbf{M}^R) \odot \mathbf{I}_r)||_2^2, \tag{12}$$

where $\hat{\pi}_i,\ \hat{\sigma}_i,\ \hat{\mathbf{I}}_i$ are the weight, scale and image warped by the i-th plane, respectively. $\mathbf{I}_r$ is the right view image, $\hat{\mathbf{I}}_r$ is the synthesised right view obtained by compositing $\hat{\mathbf{I}}_i$, and ϕ_l is the first l maxpool layers of a VGG19 [11] model pre-trained on the ImageNet [2].

Therefore, our final loss in the self-distillation stage is:

$$\mathcal{L}_{\text{distill}} = \mathcal{L}_{\text{MLL}}^{M} + \lambda_1 \mathcal{L}_{\text{pc}}^{M} + \lambda_2 \mathcal{L}_{\text{ds}} + \lambda_3 \mathcal{L}_{\text{sd}} \tag{13}$$

which is averaged over pixel, view and batch. λ_1, λ_2 and λ_3 are the weight hyper-parameters.

C. Implement Details

C.1. Training Details

We implement our network using PyTorch [9] and train it using Adam [8] with $\beta_1 = 0.5, \beta_2 = 0.999$. Our default data augmentations consist of resizing using a random scaling factor between 0.75 and 1.5, random cropping using size 640×192, and random gamma, brightness and color augmentations. In the first stage, we train our model with $\mathcal{L}$ for 50 epochs using a batch size of 8, where half of input images are obtained by horizontal flipping the other half. Our initialization learning rate is 1×10^{-4}, which is halved at 30 and 40 epochs, respectively. We then fine-tune the network with another epoch at high resolution (1280×384) without resizing and cropping data augmentation. Subsequently, we train another 10 epochs using the self-distillation loss $\mathcal{L}_{\text{distill}}$ with a batch size of 4 at high resolution, still without resizing and cropping. or self-distillation, we use an initialization learning rate of 2×10^{-5}, which is halved at 5 epoch. We set the minimum and maximum disparity to $d_{\text{min}} = 2$ and $d_{\text{max}} = 300$, and the minimum and maximum camera heights to $h_{\text{min}} = 1$ and $h_{\text{max}} = 2$. Additionally, we set the hyper-parameters of our loss function to $\lambda_1 = 0.1,\ \lambda_2 = 0.04,\ \lambda_3 = 1$.

C.2. Network Architecture

We adopt the U-Net [10] with DenseASPP [12] module as the base network for depth estimation, as proposed by Zhou and Dong [13]. To improve the accuracy of the ground depth and segmentation, we further enhance our PlaneDepth network by incorporating neural positional encoding (NPE). Similar to [4], we simply use the first five blocks of ResNet50 [7] as the encoder, and the five upsampling blocks proposed in [4] as our decoder. Consistent with [13], we insert a DenseASPP module [12] with dilation rates of $r \in \{3, 6, 12, 18, 24\}$ between the first two blocks of the decoder. To assist the network in learning resizing and cropping information, we encode grid $\mathbf{g}$ as an 8-channel feature map using NPE following [5], and concatenate it with the input of the first four decoder blocks.

For our pose estimation network, following Godard *et al*. [4], we use the first five blocks of ResNet18 [7] with double input channels as the encoder to extract two view features. We then employ three convolution layers followed by global average pooling for the decoder. Additionally, we concatenate an 8-channel resizing and cropping feature with the input of the decoder.

C.3. Evaluation Metrics

The metrics used for evaluation are defined as:

1. Abs Rel $= \frac{1}{n}\sum_{i=1}^{n} \frac{|D_i^* - D_i|}{D^*}$
2. Sq Rel $= \frac{1}{n}\sum_{i=1}^{n} \frac{(D_i^* - D_i)^2}{D^*}$
3. RMSE $= (\frac{1}{n}\sum_{i=1}^{n}(D_i^* - D_i)^2)^{1/2}$
4. RMSE log $= (\frac{1}{n}\sum_{i=1}^{n}(\log D_i^* - \log D_i)^2)^{1/2}$
5. $\text{A}_t = \frac{1}{n}|\{D_i | i \leq n, max(\frac{D_i}{D_i^*}, \frac{D_i^*}{D_i}) < 1.25^t\}|$
6. MMP $= \frac{1}{n}\sum_{i=1}^{n}(\max_{0 \leq j \leq N-1} p_j / \sum_{k=0}^{N-1} p_k)$

where D^* and D represent the ground truth and the predicted depth map, respectively. p_i denotes the probability that the pixel belongs to the i-th plane.

D. Additional Experiments

D.1. Ablation of Training Strategy

Without any special data augmentation, the depth estimation network will predict depth based solely on the vertical image position cue [3]. Gonzalez and Kim [6] showed that Resizing and cropping corrupt the vertical image position of object, enabling the network to exploit the relative object size cue. Furthermore, since disparity and relative object size remain consistent, a network that predicts disparity using the relative size cue can naturally adapt to inputs of various resolutions.

Since the vertical image position cue of outdoor scenes is also closely related to depth, exploiting this cue can further improve the performance. Taking advantage of the fact that networks tend to learn the vertical image position, we propose fine-tuning the network for an additional epoch at

Stage1 Resolution	Stage1 RC	Finetune Resolution	Finetune RC	LR performance				HR performance			
				Abs Rel↓	Sq Rel↓	Rmse↓	A1↑	Abs Rel↓	Sq Rel↓	Rmse↓	A1↑
LR		LR		0.103	0.748	4.631	0.878	0.461	5.342	8.675	0.414
HR		HR		0.174	1.284	6.215	0.713	0.090	0.630	4.270	0.898
LR	✓	LR	✓	**0.100**	**0.708**	**4.588**	**0.882**	0.090	0.616	4.188	0.898
LR	✓	HR		**0.100**	0.713	4.653	0.877	**0.086**	**0.581**	**4.094**	**0.906**

Table 1. Comparison of different training strategies in the first training stage and the fine-tune epoch. LR denotes low resolution (640×192), HR represents high resolution (1280×384), and RC indicates resizing and cropping data augmentation. The results show that exploiting both monocular cues in our training strategy significantly improves performance.

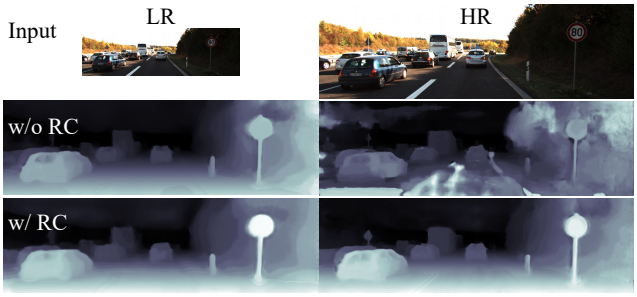

Figure 2. **Visualization of the effect of resizing cropping.** LR and HR are low and high resolution, respectively. RC means training with resizing and cropping. The prediction results of inputs with different resolutions are resized to the same resolution for evaluation. Network trained using resizing and cropping can naturally adjust to inputs with different resolutions.

high resolution without resizing and cropping at the end of training. This fine-tuning step can further introduce vertical image position cue and adapt the network to real images, resulting a performance boost as shown in Tab. 1.

Table 1 shows that networks trained without resizing and cropping are limited to performing only at the resolution of the training data. However, networks trained using resizing and cropping augmentation perform well in both low and high resolutions. Compared with training on HR all the time, our strategy of training on LR and fine-tuning on HR saves a lot of training time and performs better, indicating that the performance improvement is due to the utilization of both cues rather than the increase in training resolution. Compared with training on LR with RC all the time, our method learns the vertical image position cue at high resolution during fine-tuning. This improves performance at HR but affects performance at LR since the vertical image position cue is not universal across different resolutions.

D.2. Monocular Training Results

We conduct experiments with various monocular training settings and show results in Tab. 2. We find that adding monocular supervision without considering appropriate measures significantly decreases performance. However, when using automask [4] and NPE, this negative impact can be mitigated. When stereo supervision is not used, our predefined planes are no longer in suitable positions due

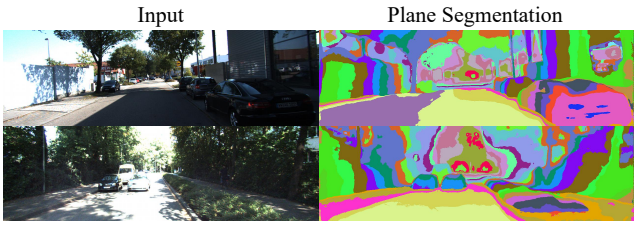

Figure 3. **Visualization of regions modeled by different planes.** The ground is modeled by composing different ground planes, which is necessary since the ground is not always an ideal horizontal plane.

Train	NPE	AM	Abs Rel↓	Sq Rel↓	Rmse↓	A1↑
S			0.089	0.598	4.175	0.900
MS	✓		0.109	1.183	5.034	0.877
MS		✓	0.093	**0.599**	4.203	0.891
MS	✓	✓	**0.092**	0.601	**4.188**	**0.893**
M	✓	✓	0.348	3.842	11.383	0.406
M†		✓	0.113	1.049	4.943	0.859

Table 2. Comparison of different monocular training settings. NPE refers to adding NPE as input to the pose network. AM is automask proposed by [4]. S stands for stereo training using left and right views with a fixed baseline. M denotes monocular training using front and rear frames without camera poses of the left view as reference views. MS stands for both stereo and monocular training. All depth networks use NPE inputs. †: We use the pretrained pose network provided by [4] to solve the scale ambiguity.

to the scale ambiguity, leading to training failure. To address this issue, we try to use the pretrained pose network provided by [4] for predicting poses with suitable scales and achieve good performance. This result confirms the correctness of our resizing cropping transformation of camera pose.

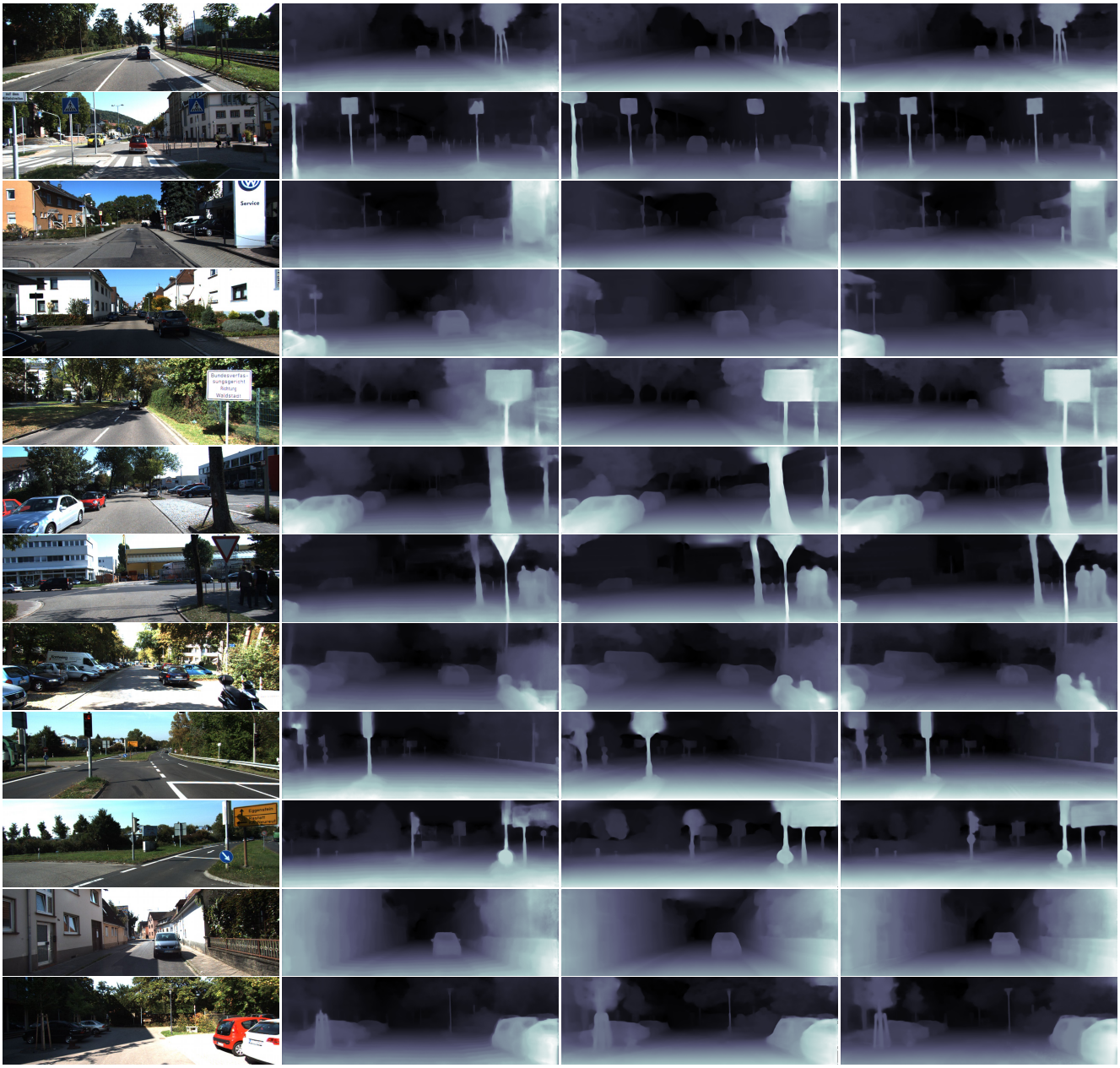

Figure 4. **Additional qualitative results on the KITTI dataset.** Our network predicts smoother depth for the ground while preserving thin structures and sharp object edges with fewer depth artifacts.

References

[1] Juan Luis Gonzalez Bello, Jaeho Moon, and Munchurl Kim. Positional information is all you need: A novel pipeline for self-supervised svde from videos. *arXiv preprint arXiv:2205.08851*, 2022. 1

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[3] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2183–2191, 2019. 2

[4] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 2, 3

[5] Juan Luis Gonzalez and Munchurl Kim. Plade-net: Towards pixel-level accuracy for self-supervised single-view depth estimation with neural positional encoding and dis-

tilled matting loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6851–6860, 2021. 2

[6] Juan Luis GonzalezBello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. *Advances in Neural Information Processing Systems*, 33:12626–12637, 2020. 1, 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[12] Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3684–3692, 2018. 2

[13] Zhengming Zhou and Qiulei Dong. Learning occlusion-aware coarse-to-fine depth map for self-supervised monocular depth estimation. *arXiv preprint arXiv:2203.10925*, 2022. 2