Supplementary Material for: "PLADE-Net: Towards Pixel-Level Accuracy for Self-Supervised Single-View Depth Estimation with Neural Positional Encoding and Distilled Matting Loss"

Juan Luis Gonzalez Bello
juanluisgb@kaist.ac.kr

Munchurl Kim
mkimee@kaist.ac.kr

Korea Advanced Institute of Science and Technology

1. Introduction

This supplementary material provides additional details on our network architectures (on both single-view and stereo-view input versions), the total loss functions, and Neural Positional Encoding. We also present further results on the KITTI [4], CityScapes [2], and Make3D [13] datasets. Additionally, we provide a video, "Video01", to show that our PLADE-Net can generate very sharp and consistent depth estimates from single view inputs.

2. More Details on Network Architectures

For the sake of completeness and reproducibility, we provide the detailed layer descriptions of our PLADE-Net and PLADE-NetS networks in Tables 1 and 2, respectively.

2.1. Detailed PLADE-Net Architecture

Table 1 describes our PLADE-Net architecture. Other than its input and output layers, our PLADE-Net adopts a fairly simple auto-encoder backbone with residual blocks and skip connections as shown in Table 1.

2.2. Detailed PLADE-NetS Architecture

Table 2 depicts our stereo input variant, the PLADE-NetS. To incorporate right-view features, our PLADE-NetS shares the left-view encoder filter weights to process the right view input image. Then, left and right deep features are combined by simple concatenation in the bottleneck, as described in the Conv6 encoder layer in Table 2. The decoder side is the same as in our PLADE-Net, which means that the multi-scale right-view features are simply ignored in the decoder side, as described in Table 2.

3. More Details on Loss Functions

We provide more details on the total loss functions used to train our PLADE-Net following the two-stage training strategy in [8].

3.1. First Training Stage Total Loss

The total loss function (l_{s1}) in the first stage of training is a combination of $l_1 = ||\mathbf{I}'_R - \mathbf{I}_R||_1$, perceptual [11] ($l_p$), and disparity smoothness (l_{ds}) losses, as given by

$$l_{s1} = ||\mathbf{I}'_R - \mathbf{I}_R||_1 + \alpha_p l_p + \alpha_{ds} l_{ds}, \tag{1}$$

where α_p and α_{ds} are empirically set to 0.01 and 0.0004, respectively, to balance their contributions. Each term is described next.

Perceptual loss. Perceptual loss [11] (l_p) enforces similarity between the synthesized right-view image ($\mathbf{I}'_R$) and the GT right-view ($\mathbf{I}_R$) in the deep feature space of a pre-trained CNN. Perceptual loss has shown to be effective for image reconstructions [11, 12, 7] as it penalizes the relationships between the target pixel and the surrounding pixels instead of measuring the error at the single-pixel location. These relationships are given by the receptive fields and learned filter parameters of the deep network layers. The receptive fields grow in size as the CNN extracts deeper features. The perceptual loss is then given by:

$$l_p = \alpha_p \sum_{l=1}^{3} ||\phi^l(\mathbf{I}'_R) - \phi^l(\mathbf{I}_R)||_2^2, \tag{2}$$

where $\phi^l(\cdot)$ denotes the first 3 maxpool layers from the pre-trained VGG19 [14] on the ImageNet classification task.

Why is L1 norm used in photometric reconstructions and L2 norm used in perceptual loss? l_1 loss is used in photometric error as it imposes a "sharper" reconstruction, which can lead to sparser solutions than l_2. l_2 is applied in the deep feature space where it penalizes larger errors, which can significantly affect the photometric reconstructions.

Edge-preserving smoothness loss. We follow the previous works of [5, 6, 10, 8] and adopt an edge-aware smoothness loss to guide our PLADE-Net in producing sharp but smooth depth estimates as given by:

$$l^L_{ds} = ||\partial_x \mathbf{D}'_L \odot e^{-\gamma|\partial_x \mathbf{I}_L|}||_1 + ||\partial_y \mathbf{D}'_L \odot e^{-\gamma|\partial_y \mathbf{I}_L|}||_1 \tag{3}$$

Outputs	Layer descriptions	Inputs	Channels	Feature sizes
$\mathbf{I}_L$	Input image	-	3	H×W
$\mathbf{P}$	Raw (x, y) pixel locations	-	2	H×W
$\mathbf{F}_{npe}$	1×1Conv, ELU, 1×1Conv, ELU	$\mathbf{P}$	8	H×W
Conv0	3×3Conv, ELU, ResBlock	$\mathbf{I}_L$	64	H×W
Conv0_{LR}	3×3Conv, ELU, ResBlock	$\mathbf{I}_L^{(1/2)}$	64	H/2×W/2
Conv1	3×3Conv(s2), ELU, ResBlock	Concat(Conv0, $\mathbf{F}_{npe}$)	128	H/2×W/2
Conv2	3×3Conv(s2), ELU, ResBlock	Concat(Conv1, $\mathbf{F}_{npe}$, Conv0_{LR})	256	H/4×W/4
Conv3	3×3Conv(s2), ELU, ResBlock	Concat(Conv2, $\mathbf{F}_{npe}$)	256	H/8×W/8
Conv4	3×3Conv(s2), ELU, ResBlock	Concat(Conv3, $\mathbf{F}_{npe}$)	256	H/16×W/16
Conv5	3×3Conv(s2), ELU, ResBlock	Concat(Conv4, $\mathbf{F}_{npe}$)	256	H/32×W/32
Conv6	3×3Conv(s2), ELU, ResBlock	Concat(Conv5, $\mathbf{F}_{npe}$)	256	H/64×W/64
Dec6	Nearest, 3×3Conv, ELU	Conv6	128	Conv5
iConv6	3×3Conv, ELU	Concat(Dec6, Conv5)	256	Conv5
Dec5	Nearest, 3×3Conv, ELU	iConv6	128	Conv4
iConv5	3×3Conv, ELU	Concat(Dec5, Conv4)	256	Conv4
Dec4	Nearest, 3×3Conv, ELU	iConv5	128	Conv3
iConv4	3×3Conv, ELU	Concat(Dec4, Conv3)	256	Conv3
Dec3	Nearest, 3×3Conv, ELU	iConv4	128	Conv2
iConv3	3×3Conv, ELU	Concat(Dec3, Conv2)	256	Conv2
Dec2	Nearest, 3×3Conv, ELU	iConv3	128	Conv1
iConv2	3×3Conv, ELU	Concat(Dec2, Conv1)	128	Conv1
Dec1	Nearest, 3×3Conv, ELU	iConv2	64	Conv0
$\mathbf{D}_L^L$	3×3Conv	Concat(Dec1, Conv0)	N	H×W
$\mathbf{D}_L^{PR}$	$\sigma(\{g(\mathbf{D}_{L_n}^L, d_n)\}_0^N)$	$\mathbf{D}_L^L$, d_n	N	H×W
$\mathbf{I}'_R$	$\sum_{n=0}^N g\left(\mathbf{I}_L, d_n\right) \odot \mathbf{D}_{L_n}^{PR}$,	$\mathbf{I}_L$, $\mathbf{D}_L^{PR}$, d_n	3	H×W
$\mathbf{D}_L^{PL}$	$\sigma(\mathbf{D}_L^L)$	$\mathbf{D}_L^L$	N	H×W
$\mathbf{D}'_L$	$\sum_{n=0}^N d_n \mathbf{D}_{L_n}^{PL}$	$\mathbf{D}_L^{PL}$, d_n	1	H×W
ResBlock	ELU(3×3Conv(ELU(3×3Conv(X))) + X)	X	X	X
d_n	$d_n = d_{max} e^{\ln d_{max}/d_{min}(n/N-1)}$	n: Channel number	1	1

Table 1. Detailed network architecture of our PLADE-Net. s2: Stride of 2. Nearest: Nearest up-scaling to the size of corresponding encoder layer (skip connection). ELU: Exponential Linear Unit. $\mathbf{D}_L^L$: Left disparity logit volume. $\mathbf{D}_L^{PR}$: Right-from-left disparity probability volume. $\mathbf{I}'_R$: Synthetic right view. $\mathbf{D}_L^{PR}$: Left disparity probability volume. $\mathbf{D}'_L$: Left disparity estimate. d_n: Exponential disparity quantization level. X denotes any feature map input to the ResBlock.

where $\gamma = 2$ regulates the amount of edge preservation [8] and $\odot$ is the Hadamard product.

3.2. Second Training Stage Total Loss

The total loss l_{s2} for the second training stage adds our novel distilled matting Laplacian loss (l_{dm}), a deep corr-l_1 loss (l_{dc}), and the mirror loss (l_m) in [8] to an occlusions free l_{s1} loss. l_{s2} is defined by:

$$l_{s2} = l_{s1} + l_m + \alpha_{dm} l_{dm} + \alpha_{dc} l_{dc}, \tag{4}$$

where $\alpha_{dm} = 0.25$ and $\alpha_{dc} = 0.01$ are empirically set to weight the contributions of the distilled matting Laplacian and the deep corr-l_1 losses, respectively. Our proposed deep correlation (l_{dc}) and distilled matting Laplacian (l_{dm}) losses are described in detail in our main paper.

Occlusions-Free l_{s1}. The l_1 and l_p components of l_{s1} are weighted by the right-view occlusion mask $\mathbf{O}_R$ during the second stage of training to remove the occluded contents which are visible in the right view but hidden in the left view. $\mathbf{O}_R$ is computed by the Mirror Occlusion Module proposed in [8]. $0 \leq \mathbf{O}_R \leq 1$ is active low on the occluded regions. The occlusions-free first stage loss is then given by:

$$l_{s1} = ||\mathbf{O}^R \odot (\mathbf{I}'_R - \mathbf{I}_R)||_1 + \alpha_p l_p^{O_R} + \alpha_{ds} l_{ds}, \tag{5}$$

where is $l_p^{O_R}$ is the occlusions-free perceptual loss weighted by $\mathbf{O}_R$, and is given by:

$$l_p^{O_R} = \sum_{l=1}^{3} ||\phi^l(\mathbf{O}^R \odot \mathbf{I}'_R + (1 - \mathbf{O}^R) \odot \mathbf{I}_R) - \phi^l(\mathbf{I}_R)||_2^2, \tag{6}$$

where $\mathbf{O}^R$ combines the non-occluded contents of $\mathbf{I}'_R$ with the occluded contents of $\mathbf{I}_R$. Note that the smoothness loss remains the same during both training stages, with the difference that $\alpha_{ds} = 0.0016$ during the second step of training.

Mirror loss. This term helps in removing the occlusion artifacts to the left side of the objects in the scene by supervising the left-occluded contents with a mirrored disparity

Outputs	Layer descriptions	Inputs	Channels	Feature sizes
$\mathbf{I}_L$	Input left view	-	3	H×W
$\mathbf{P}$	Raw (x, y) pixel locations	-	2	H×W
$\mathbf{F}_{npe}$	1×1Conv, ELU, 1×1Conv, ELU	$\mathbf{P}$	8	H×W
Conv0	3×3Conv, ELU, ResBlock	$\mathbf{I}_L$	64	H×W
Conv0_{LR}	3×3Conv, ELU, ResBlock	$\mathbf{I}_L^{(1/2)}$	64	H/2×W/2
Conv1	3×3Conv(s2), ELU, ResBlock	Concat(Conv0, $\mathbf{F}_{npe}$)	128	H/2×W/2
Conv2	3×3Conv(s2), ELU, ResBlock	Concat(Conv1, $\mathbf{F}_{npe}$, Conv0_{LR})	256	H/4×W/4
Conv3	3×3Conv(s2), ELU, ResBlock	Concat(Conv2, $\mathbf{F}_{npe}$)	256	H/8×W/8
Conv4	3×3Conv(s2), ELU, ResBlock	Concat(Conv3, $\mathbf{F}_{npe}$)	256	H/16×W/16
Conv5	3×3Conv(s2), ELU, ResBlock	Concat(Conv4, $\mathbf{F}_{npe}$)	256	H/32×W/32
$\mathbf{I}_R$	Input right view	-	3	H×W
Conv0r	3×3Conv, ELU, ResBlock	$\mathbf{I}_R$	64	H×W
Conv1r	3×3Conv(s2), ELU, ResBlock	Concat(Conv0r, $\mathbf{F}_{npe}$)	128	H/2×W/2
Conv2r	3×3Conv(s2), ELU, ResBlock	Concat(Conv1r, $\mathbf{F}_{npe}$, Conv0_{LR})	256	H/4×W/4
Conv3r	3×3Conv(s2), ELU, ResBlock	Concat(Conv2r, $\mathbf{F}_{npe}$)	256	H/8×W/8
Conv4r	3×3Conv(s2), ELU, ResBlock	Concat(Conv3r, $\mathbf{F}_{npe}$)	256	H/16×W/16
Conv5r	3×3Conv(s2), ELU, ResBlock	Concat(Conv4r, $\mathbf{F}_{npe}$)	256	H/32×W/32
Conv6	3×3Conv(s2), ELU, ResBlock	Concat(Conv5, Conv5r, $\mathbf{F}_{npe}$)	256	H/64×W/64
Dec6	Nearest, 3×3Conv, ELU	Conv6	128	Conv5
iConv6	3×3Conv, ELU	Concat(Dec6, Conv5)	256	Conv5
Dec5	Nearest, 3×3Conv, ELU	iConv6	128	Conv4
iConv5	3×3Conv, ELU	Concat(Dec5, Conv4)	256	Conv4
Dec4	Nearest, 3×3Conv, ELU	iConv5	128	Conv3
iConv4	3×3Conv, ELU	Concat(Dec4, Conv3)	256	Conv3
Dec3	Nearest, 3×3Conv, ELU	iConv4	128	Conv2
iConv3	3×3Conv, ELU	Concat(Dec3, Conv2)	256	Conv2
Dec2	Nearest, 3×3Conv, ELU	iConv3	128	Conv1
iConv2	3×3Conv, ELU	Concat(Dec2, Conv1)	128	Conv1
Dec1	Nearest, 3×3Conv, ELU	iConv2	64	Conv0
$\mathbf{D}_L^L$	3×3Conv	Concat(Dec1, Conv0)	N	H×W
$\mathbf{D}_L^{PR}$	$\sigma(\{g(\mathbf{D}_{Ln}^L, d_n)\}_0^N)$	$\mathbf{D}_L^L, d_n$	N	H×W
$\mathbf{I}'_R$	$\sum_{n=0}^N g\left(\mathbf{I}_L, d_n\right) \odot \mathbf{D}_{Ln}^{PR}$,	$\mathbf{I}_L, \mathbf{D}_L^{PR}, d_n$	3	H×W
$\mathbf{D}_L^{PL}$	$\sigma(\mathbf{D}_L^L)$	$\mathbf{D}_L^L$	N	H×W
$\mathbf{D}'_L$	$\sum_{n=0}^N d_n \mathbf{D}_{Ln}^{PL}$	$\mathbf{D}_L^{PL}, d_n$	1	H×W
ResBlock	ELU(3×3Conv(ELU(3×3Conv(X))) + X)	X	X	X
d_n	$d_n = d_{max} e^{\ln d_{max}/d_{min}(n/N-1)}$	n: Channel number	1	1

Table 2. Detailed network architecture of our proposed PLADE-NetS. Encoder weights are shared.

estimate $\mathbf{D}'_{Lmr}$ [8]. $\mathbf{D}'_{Lmr}$ is obtained from a horizontally flipped version of the input image $\mathbf{I}'_L$ by a fixed copy of the PLADE-Net with only the first stage of training. The mirror loss is weighted by the left occlusion mask $\mathbf{O}_L$, and is given by:

$$l_m = (1/max(\mathbf{D}'_{Lmr}))||(1 - \mathbf{O}^L) \odot (\mathbf{D}'_L - \mathbf{D}'_{Lmr})||_1 \quad (7)$$

where $max(\mathbf{D}'_{Lmr})$ is the maximum disparity value in $\mathbf{D}'_{Lmr}$ that normalizes the mirror loss.

4. More Details on Neural Positional Encoding

Positional encoding (PE) has been used in natural language processing (NLP) to provide the models with means of understanding the location of a word relative to the text in which it appears [16]. In "Attention Is All You Need" [16], positions are encoded by several sinusoids with varying frequencies, converting each position into a 512-element vector. In the case of image processing, adopting the positional encoding in [16] is un-permissible, as adding 512 channels to each CNN layer would be computationally too expensive. For this reason, we propose to use neural positional encoding (NPE) instead, which shows good performance while adding only a few channels to each convolutional stage in our PLADE-Net's encoder. We explored the PE in [16] with eight sinusoids (or eight channels) and observed poor performance with no improvements in some metrics and even deterioration in others, as shown in Table 3. Such performance difference shows that our proposed NPE is more suitable for image processing, where computing budgets limit the number of positional features.

Ref	Methods	PP	Sup	Data	#Par	abs rel	sq rel	rmse	$\text{rmse}_{\log}$	δ^1	δ^2	δ^3
Improved Eigen Test Split [15]												
our	PLADE-Net without PE		S	K+CS	15	0.075	0.317	2.990	0.111	0.938	0.989	0.997
our	PLADE-Net with [16]'s PE		S	K+CS	15	**0.070**	0.298	3.100	0.111	0.935	0.989	**0.998**
our	PLADE-Net (narrow)		S	K+CS	10	0.072	0.308	3.064	0.112	0.936	0.990	**0.998**
our	PLADE-Net (wide)		S	K+CS	29	**0.070**	0.295	2.961	0.108	0.941	**0.991**	**0.998**
our	PLADE-Net		S	K+CS	15	**0.070**	**0.291**	**2.910**	**0.107**	**0.942**	0.990	**0.998**

Table 3. Evaluations on the improved KITTI Eigen test split [15] for models with [16]'s positional encoding, less (narrow), and more (wide) network parameters in the first stage of training. S indicates training from stereo. Metrics are the lower the better and the higher the better. **Best** and second best metrics. Results capped to 80m.

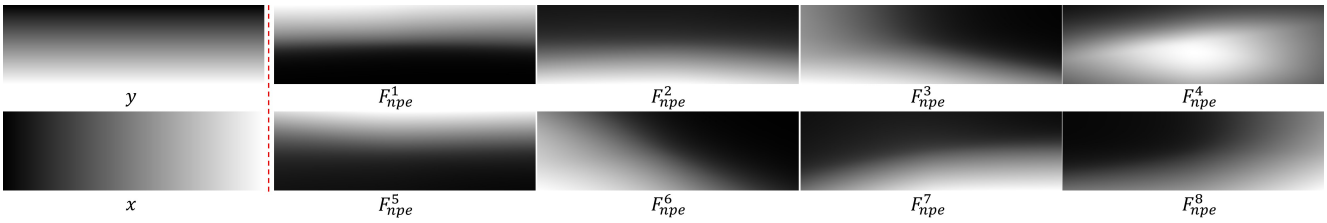

Figure 1. Visualization of estimated Neural Positional Encodings (NPE).

4.1. Visualization of Estimated Positional Features

We visualize the estimated positional features versus the raw (x, y) pixel locations in figure 1 and notice interesting properties. While some features generate activations similar to the Y-axis (features 1, 2, and 3), no feature map seems to directly represent the X-axis. Interestingly, feature F^4_{npe} depicts a gradient circle near the center of the image, suggesting a means of learning the relative position of the input pixels versus the lens or projection distortions discussed in Section 3.2 of our main paper.

5. More Results on the KITTI dataset

We provide additional results on the KITTI Eigen test split [3] in Figure 2. As can be observed, our PLADE-Net consistently generates sharper, more detailed, and more accurate depth estimates than the previous SOTA [17, 10, 8].

5.1. Varying the Number of Network Parameters

For the sake of completeness, we explored wider and narrower versions of our PLADE-Net by varying the number of learnable filters in its auto-encoder backbone. Table 3 shows the performance of the narrower and wider versions of our PLADE-Net, with 10M and 29M parameters respectively, for the first stage of training. Our PLADE-Net (narrow) has half the number of filter channels in the Conv2 and Conv3 layers in Table 1, and does not perform as good as our base PLADE-Net. On the other hand, our PLADE-Net (wide) has the double of number of parameters in the Conv5 and Conv6 layers, and yields no improvements with respect our base PLADE-Net. Such behaviour confirms that, for the single view depth estimation task, shallow feature extraction plays a critical role.

5.2. Single-View Versus Stereo

We compare the depth estimates of the single-view and stereo-view input versions of our PLADE-Net on the KITTI Eigen test split [3] in Figure 4. As can be noted, both the PLADE-Net and PLADE-NetS generate detailed depth estimates, achieving 95% and 98.9% in δ^1 accuracy, respectively. Interestingly, the single-view estimates of our PLADE-Net, with lower quantitative accuracy than the PLADE-NetS, appear to generate more consistent depths in certain regions, like the windows, the van, and the traffic signs in Figure 4 rows 4, 6, and 8, respectively. It is worth mentioning that future research works could explore training the PLADE-Net and the PLADE-NetS simultaneously to exploit each network's strong points to self-supervise the other.

5.3. Textured Point Clouds

In some cases, it is hard to visualize the subtle differences between the depth estimates when presented as 2D maps. To provide further evidence of the superiority of our method versus the previous SOTA, we provide novel views synthesized from textured point clouds obtained from the PLADE-Net's depth estimates in Figure 3. It is noted in Figure 3 that in comparison with the previous SOTA of FALnet [8], our PLADE-Net manages to keep various 3D geometries properly: the door frame and the van's rear bumper in the first column; the building structure and car geometries in the second column; the traffic lights in the third column; and the building and pedestrian geometries in the last column.

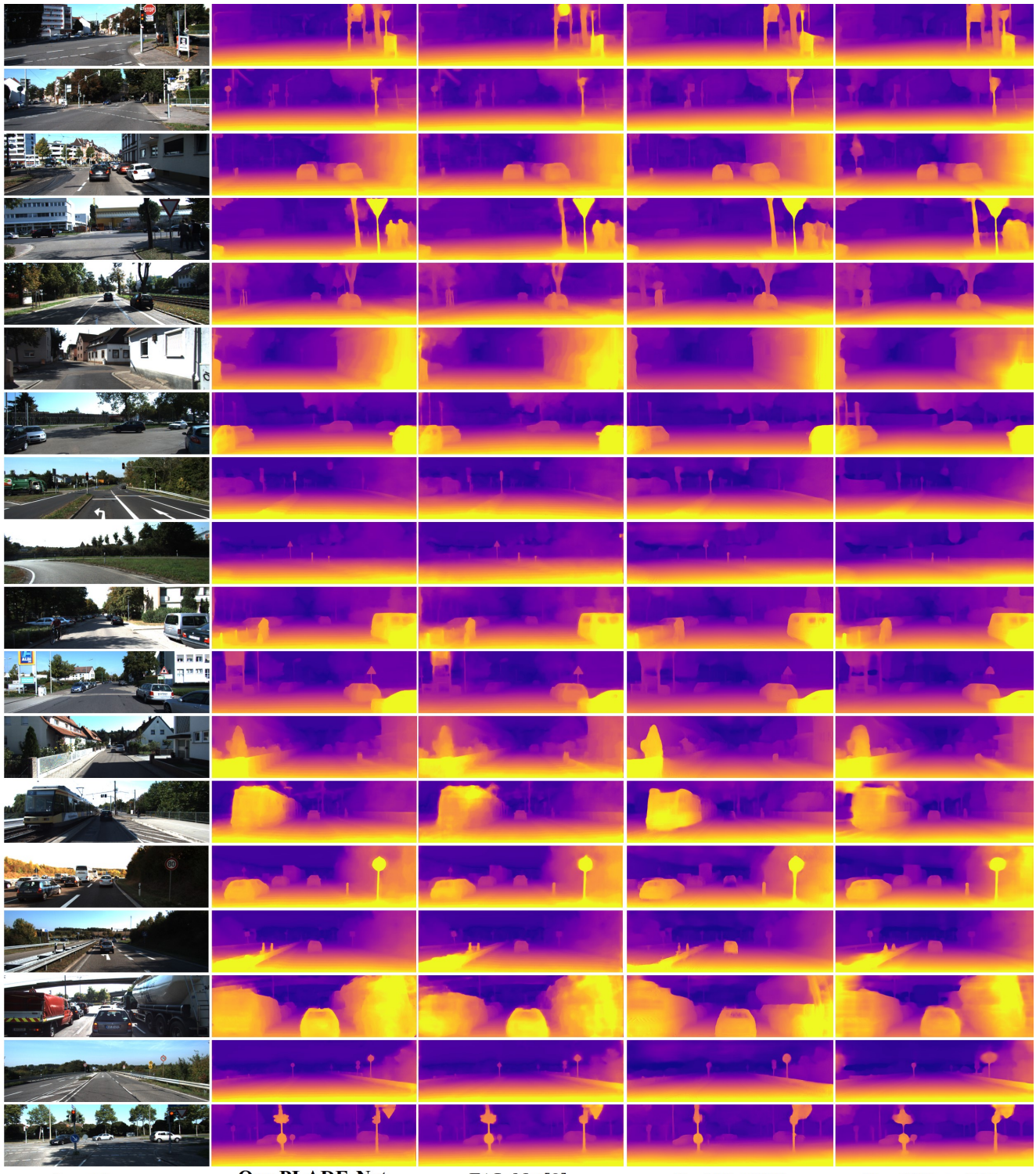

Figure 2. Additional qualitative results on the KITTI [4] dataset. Our Pixel Level Accurate Depth Estimation Network (PLADE-Net) consistently generates sharper and more detailed depth estimates.

Figure 3. Textured point clouds (generated from depth estimates) at novel view-points. From top to bottom: Input images, renders by FAL-net [8], and renders by our PLADE-Net. The camera is translated and rotated to $[0\text{m}, 0.2\text{m}, 3\text{m}, -11°, 0°, 0°]$ in the leftmost two columns, and translated to $[-0.9\text{m}, 0\text{m}, 2\text{m}, 0°, 0°, 0°]$ in the rightmost two columns.

6. More Results on the CityScapes dataset

To further show the generalization capabilities of our PLADE-Net, results of only training on CityScapes (CS) [2] and validating on the KITTI Eigen test split [3] are provided in Table 4 and Figure 4. As can be observed, our method generalizes the best among the competing methods, showing higher accuracy and lower error metrics in Table 4. Our PLADE-Net trained on CityScapes [2] generates very detailed and plausible depths on the KITTI [4] dataset. As can be observed in Figure 4, the depth estimates of our PLADE-Net trained on CityScapes only (CS) are on pair with the PLADE-Net (K+CS) depth estimates, which demonstrates the excellent generalization power of our proposed method.

7. More Results on the Make3D dataset

We provide additional results on the Make3D [13] dataset in Figure 5 and compare with previous existing works [6, 8]. It is clear that our method consistently generates more detailed depth estimates in the previously unseen Make3D [13] dataset.

8. Video01

We provide a video from sequence 18 from the KITTI odometry dataset [4], which is not included in the KITTI Eigen train split [3]. Our PLADE-Net generates a consistent sequence, even when only being fed with individual images. The inference is also very fast and light, taking only an average of 13 milliseconds per full-resolution (1242×375) image on a TitanXP GPU with 2GB of allocated memory. Note that each depth estimate is individually normalized in the range $[0, 1]$ for easier visualization.

References

[1] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8001–8008, 2019. 7

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 1, 6, 7

[3] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014. 4, 6, 7

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 5, 6, 7

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017. 1

[6] Juan Luis Gonzalez and M. Kim. A novel monocular disparity estimation network with domain transformation and ambiguity learning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 474–478, Sep. 2019. 1, 6

[7] Juan Luis Gonzalez and M. Kim. Deep 3d pan via local adaptive "t-shaped" convolutions with global and local adaptive dilations. In *International Conference on Learning Representations*, 2020. 1

[8] Juan Luis GonzalezBello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. In *Advances in Neural Information Process-*

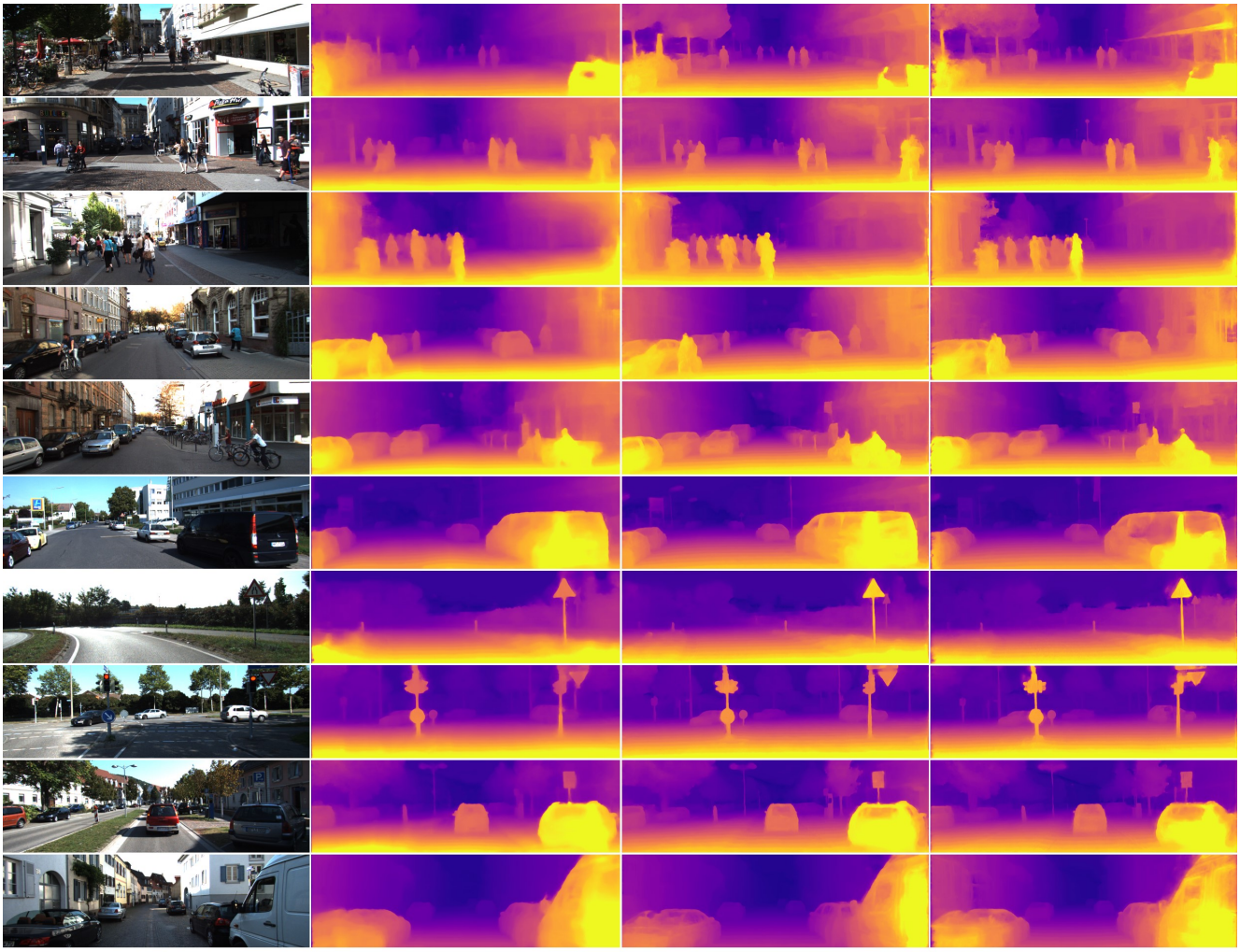

Figure 4. Additional qualitative comparison on the KITTI [4] dataset for our PLADE-Net trained on CityScapes [2] only (CS), our PLADE-Net trained on KITTI and CityScapes concurrently (K+CS), and our PLADE-NetS trained on (K).

Ref	Methods	PP	Sup	Data	#Par	abs rel	sq rel	rmse	$\text{rmse}_{\log}$	δ^1	δ^2	δ^3
Original Eigen Test Split [3]												
[9]	Gordon *et al.*		V	CS	-	0.172	1.370	6.210	0.250	0.754	0.921	0.967
[1]	Casser *et al.*		V	CS	-	0.153	1.109	5.557	0.227	0.796	0.934	0.975
[8]	FAL-net		S	CS	17	<u>0.144</u>	<u>0.871</u>	<u>4.796</u>	<u>0.215</u>	<u>0.811</u>	<u>0.947</u>	<u>0.979</u>
our	PLADE-Net		S	CS	15	**0.135**	**0.785**	**4.596**	**0.205**	**0.828**	**0.956**	**0.982**

Table 4. Evaluations on the KITTI Eigen test split [3] for models trained on CityScapes[2] (CS). S indicates training from stereo.V indicates training from monocular videos. All methods benefit from median-scaling. **Best** and <u>second-best</u> metrics. Methods that use post-processing (PP) are checked ✔. Results capped to 80m.

ing Systems, volume 33, pages 12626–12637. Curran Associates, Inc., 2020. 1, 2, 3, 4, 6, 7

[9] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8977–8986, 2019. 7

[10] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 4

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In

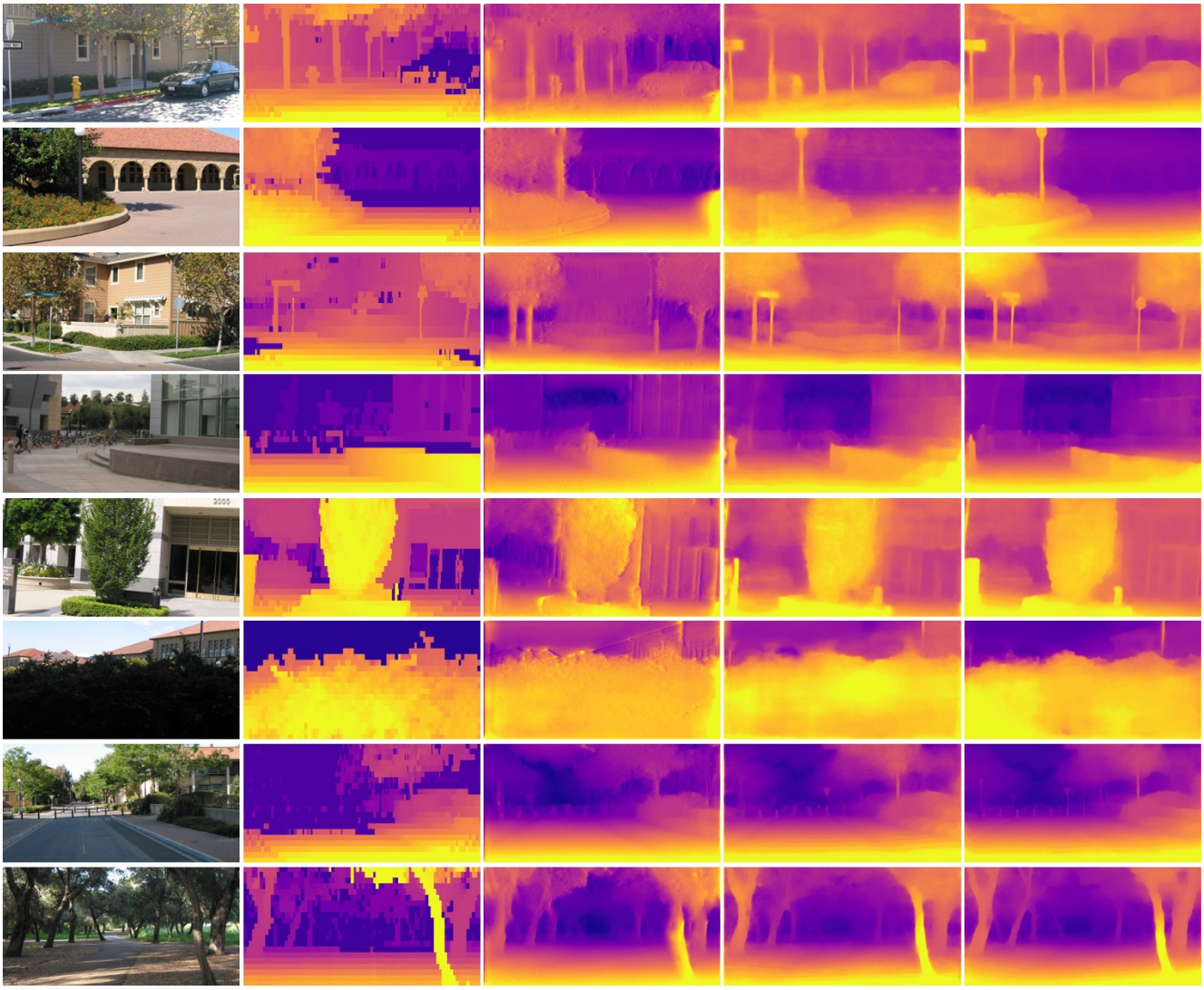

Figure 5. Additional qualitative comparison on the Make3D [13] dataset.

European conference on computer vision, pages 694–711. Springer, 2016. 1

[12] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 261–270, 2017. 1

[13] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008. 1, 6, 8

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[15] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017. 4

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3, 4

[17] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2162–2171, 2019. 4