DeepLiDAR: Deep Surface Normal Guided Depth Prediction for Outdoor Scene from Sparse LiDAR Data and Single Color Image (Supplementary Materials)

Jiaxiong Qiu[1*] Zhaopeng Cui[2*] Yinda Zhang[3*]
Xingdi Zhang[1] Shuaicheng Liu[1,4†] Bing Zeng[1] Marc Pollefeys[2,5]
[1] UESTC [2]ETH Zürich [3]Google [4]Megvii Technology [5]Microsoft

In this supplementary materials, we provide more details about our synthetic data, network architecture, ablation study, and qualitative results on both indoor and outdoor data.

1. Synthetic Data

Due to the lack of high-quality surface normal ground truth for real data, we generate synthetic data to pre-train the surface normal estimation network (i.e. a DCU).

1.1. CARLA

CARLA is an open-source simulator for the research of autonomous driving. It provides virtual engine that allows cars to run with customized control strategy in virtual environment as a way to test autonomous driving system. The virtual environment can be edited in Unreal Engine 4, and the simulator can be run through python API. More details can be found at its github repository[1].

1.2. Data Generation

We use CARLA to generate synthetic training data from two official provided virtual scenes, named Town01 and Town02. Trees with inaccurate surface normals are removed to keep the ground truth of surface normals clean and correct. To increase the diversity of data, we also manually create three more scenes with different road maps, illumination, types of vehicles and buildings, and vehicle density and distribution (e.g., heavy traffic and parking).

We render in total 50,000 synthetic data, including color image, surface normal, and LiDAR sparse depth. The rendering process takes two days on a single machine with a 4-core CPU. Virtual cameras are picked as 150ms per frame on a driving car on road controlled by the authors to fully explore the maps. The color images and surface normals are rendered from the Unreal Engine. For the sparse depth, we run the LiDAR simulation in CARLA, which produces sparse 3D points from a spinning LiDAR. We project the sparse 3D points into the image coordinates for the sparse depth. The camera is placed at the center on top of the car. The camera intrinsics are set as $f_x = f_y = 512, c_x = c_y = 256$.

Fig. 1 shows some examples of the synthetic data. As can be seen, our data shows high diversity with various vehicles and buildings, shadows and lighting, and roadside objects like road lamps and traffic signs. The ground-truth surface normals and depth are also clean and accurate.

1.3. Benefits

Accurate dense depth is required to generate accurate dense surface normals, while it is non-trivial to collect dense depth for the outdoor scene. As a result, large scale surface normal ground truth for real images does not exist, and the normals converted from the depth of real scenes usually contain quite a lot of noises. Therefore, we use our synthetic data to pre-train the deep completion unit for surface normal prediction.

We evaluate the angular error of the estimated normals on KITTI validation set. The mean errors with and without pre-training on synthetic data are 20.8° and 25.6° respectively. The pre-training on the synthetic data significantly improves the surface normal estimation (reducing 20% of the angular error). This observation is consistent with Zhang *et al.* [14].

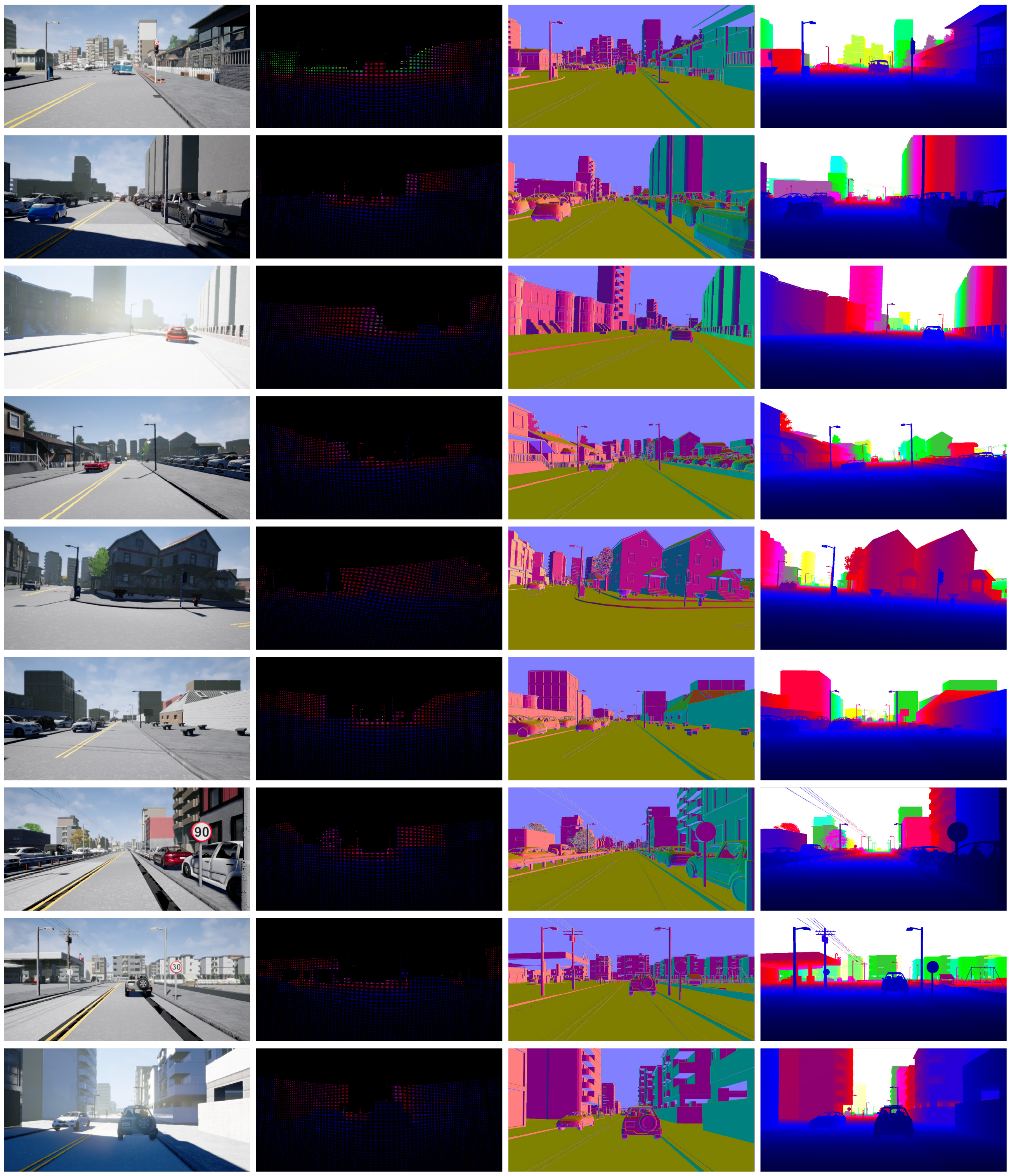
2. Network Architecture

In this section, we introduce the detailed network architectures for deep completion unit and attention based integration.

2.1. Deep Completion Unit

We use deep completion units (DCU) to produce dense output from a dense color image and a sparse depth. It contains two encoders consisting of ResNet blocks, and one

*indicates equal contributions.
†indicates corresponding author.
[1]https://github.com/carla-simulator/carla



(a) RGB image (b) Sparse depth (c) Surface normal (d) Dense depth

Figure 1. **Examples of our synthetic data.** We use Carla [3] to generate 50,000 synthetic data for outdoor on-road scene. Besides RGB image (a), dense depth (d), and surface normal (c), we also run its LiDAR simulator to generate sparse depth (b). For better visualization, the sparse depth samples are enlarged twice.

Name	Layer Description	Output Tensor Dim.
Input	image	H×W×3
	concat(sparse input, mask)	H×W×2
	Image Encoder	
Res1_1	3×3 Res.Block, 32 channels, stride 1	H×W×32
Res1_2	3×3 Res.Block, 64 channels, stride 1	H×W×64
Res2_1	3×3 Res.Block, 128 channels, stride 2	$\frac{1}{2}$H×$\frac{1}{2}$W×128
Res2_2	3×3 Res.Block, 128 channels, stride 1	$\frac{1}{2}$H×$\frac{1}{2}$W×128
Res3_1	3×3 Res.Block, 256 channels, stride 2	$\frac{1}{4}$H×$\frac{1}{4}$W×256
Res3_2	3×3 Res.Block, 256 channels, stride 1	$\frac{1}{4}$H×$\frac{1}{4}$W×256
Res4_1	3×3 Res.Block, 256 channels, stride 2	$\frac{1}{8}$H×$\frac{1}{8}$W×256
Res4_2	3×3 Res.Block, 256 channels, stride 1	$\frac{1}{8}$H×$\frac{1}{8}$W×256
Res5_1	3×3 Res.Block, 512 channels, stride 2	$\frac{1}{16}$H×$\frac{1}{16}$W×512
Res5_2	3×3 Res.Block, 512 channels, stride 1	$\frac{1}{16}$H×$\frac{1}{16}$W×512
	Sparse Encoder	
ResS1	3×3 Res.Block, 32 channels, stride 1	H×W×32
ResS2	3×3 Res.Block, 97 channels, stride 1	H×W×97
ResS3	3×3 Res.Block, 193 channels, stride 2	$\frac{1}{2}$H×$\frac{1}{2}$W×193
ResS4	3×3 Res.Block, 385 channels, stride 2	$\frac{1}{4}$H×$\frac{1}{4}$W×385
ResS5	3×3 Res.Block, 513 channels, stride 2	$\frac{1}{8}$H×$\frac{1}{8}$W×513
ResS6	3×3 Res.Block, 512 channels, stride 2	$\frac{1}{16}$H×$\frac{1}{16}$W×512
	Decoder	
sum_1	**Sum**(Res5_2, ResS6)	$\frac{1}{16}$H×$\frac{1}{16}$W×512
up_1	Up-projection Block	$\frac{1}{8}$H×$\frac{1}{8}$W×257
sum_2	**Sum**(concat(Res4_2,up_1), ResS5)	$\frac{1}{8}$H×$\frac{1}{8}$W×513
up_2	Up-projection Block	$\frac{1}{4}$H×$\frac{1}{4}$W×129
sum_3	**Sum**(concat(Res3_2,up_2), ResS4)	$\frac{1}{4}$H×$\frac{1}{4}$W×385
up_3	Up-projection Block	$\frac{1}{2}$H×$\frac{1}{2}$W×65
sum_4	**Sum**(concat(Res2_2,up_3), ResS3)	$\frac{1}{2}$H×$\frac{1}{2}$W×193
up_4	Up-projection Block	H×W×33
sum_5	**Sum**(concat(Res1_2,up_4), ResS2)	H×W×97
convD5	3×3 conv, 1 channels	H×W×1
Output	dense output	H×W×1

Table 1. **Detailed architectures of the deep completion unit.** Here we take the depth estimation as an example.The deep completion units for the normal estimation share similar architectures. All the code will be released upon acceptance.

decoder consisting of up-projection blocks [8]. The architectures of the specific components in DCU for the depth estimation are listed in Tab. 1. The detailed structures of ResNet and up-projection blocks we used are shown in Fig. 2.

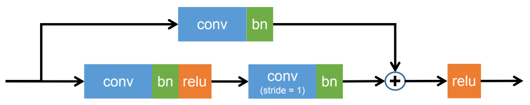
2.2. Attention Based Integration

We use an attention mechanism to integrate the depths recovered from two pathways. We feed the feature map of the last layer in each pathway into an attention block shown in Fig. 2 (c) to predict score maps. The first two convolution layers of each path in the attention block have 97 channels. The last convolution layer produces 1 channel feature map, which is then fed into the softmax layer to produce the normalized attention map.

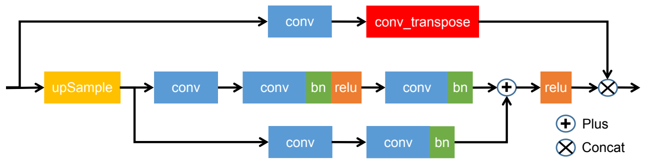
3. Model Analysis

3.1. Deep Completion Unit

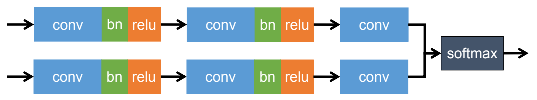
In Sec. 4.2 of the main submission, we show that replacing our DCU with traditional early fusion, i.e., the color image and spare depth are concatenated and then fed as input, causes about 43mm error increase (in RMSE). Here we



(a) ResNet block

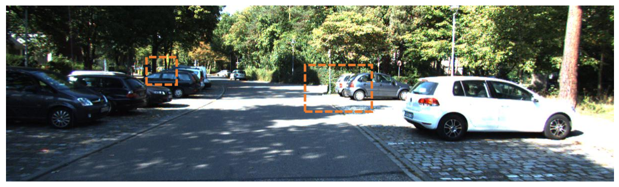


(b) Up-projection block

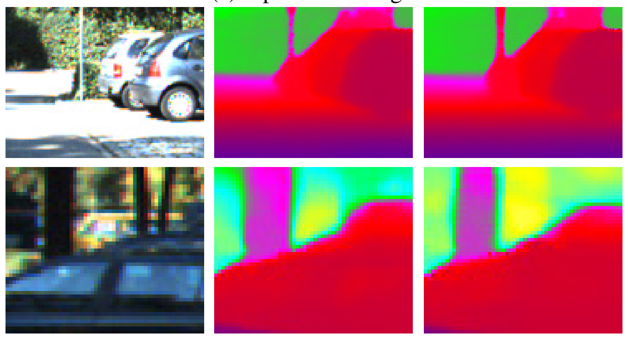


(c) Attention block

Figure 2. **Detailed structures of ResNet, up-projection, and attention blocks we used.**



(a) Input RGB image



(b) Cropped image (c) Result w/o DCU (d) Result with DCU

Figure 3. **Effect of deep completion unit.** We show qualitative results of the depth prediction from models trained without and with DCU. (b-d) show zoom-in views of the color image and the output dense depth for the marked regions in (a). The model with DCU produces sharper boundaries and preserves thin structures.

show additional evaluations.

We first show a qualitative comparison to the above early fusion strategy in Fig. 3. The model without DCU is easily affected by the ad-hoc feature, such as the bushes behind the light pole, which yields noisy depth for thin structures. Comparatively, the model trained with DCU learns to preserve the completeness of thin structures, which is consistent with the color image, such as the tree trunk.

We then keep the late fusion architecture in DCU but replace the summation with concatenation in the decoder, and re-train the whole system keeping all the other parts of the network unchanged. The RMSE again increases about 10mm. Not only just the performance is worse, the concate-

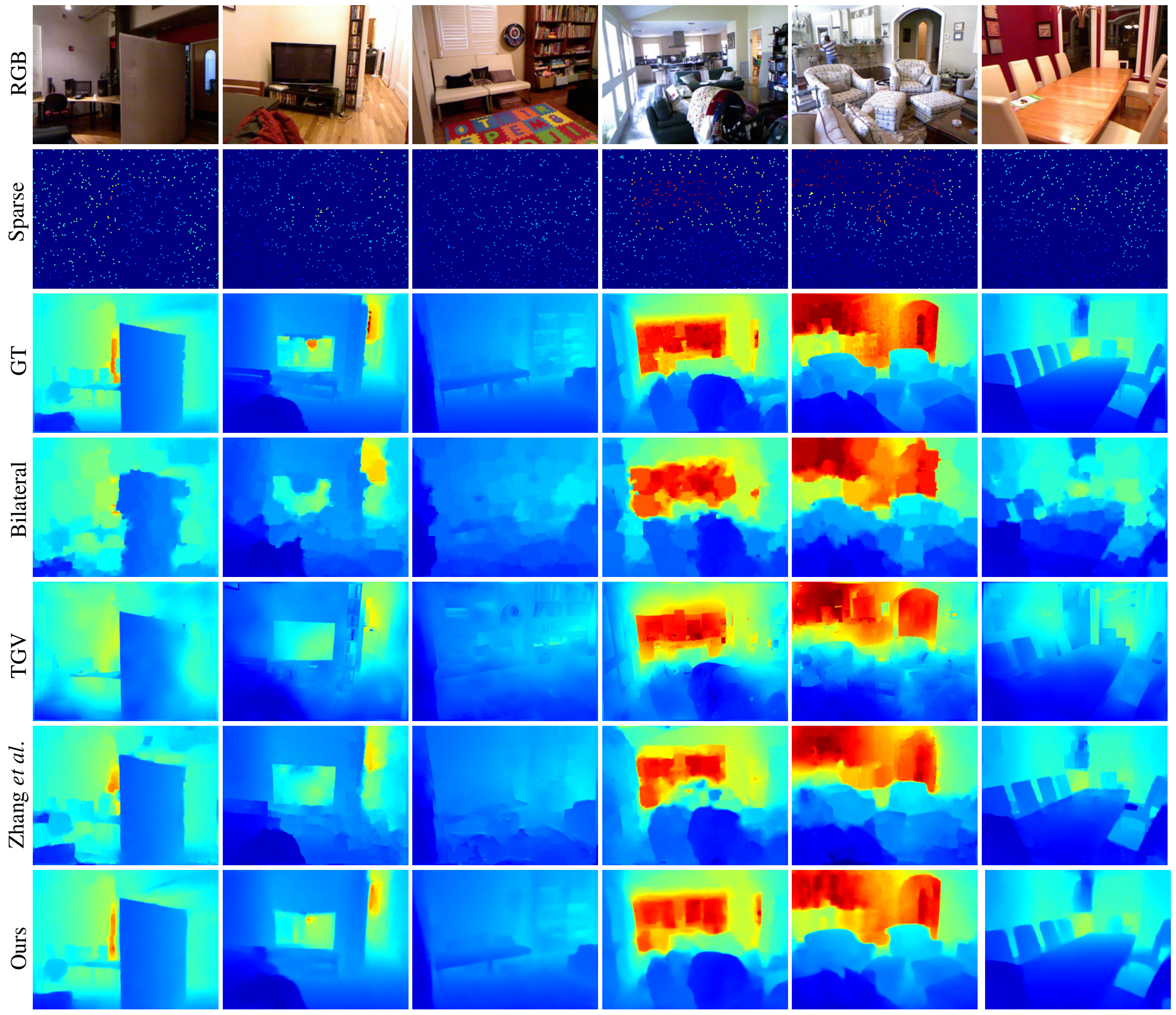


Figure 4. **Qualitative results on NYU v2 validation set [10].** From top to bottom are input RGB image, sparse depth samples (500), ground truth dense depth, results of Bilateral [10], TGV [5], Zhang *et al.* [13], and our method. For better visualization, the sparse depth samples are enlarged twice.

	RMSE	MAE	iRMSE	iMAE
CSPN [2]	1019.64	279.46	2.93	1.15
Spade-RGBsD [7]	917.64	234.81	**2.17**	**0.95**
HMS-Net [6]	841.78	253.47	2.73	1.13
MSFF-Net [12]	836.69	241.54	2.63	1.07
NConv-CNN [4]	829.98	233.26	2.60	1.03
Sparse-to-Dense [9]	814.73	249.95	2.80	1.21
Ours with inverse L_1 loss	767.34	229.06	2.40	1.08
Ours with L_2 loss	**758.38**	**226.50**	2.56	1.15

Table 2. **Performance of depth completion on KITTI test set [11].**

nation alternative also costs more memory during training and testing.

3.2. Evaluation Metric and Loss Function

We evaluate the effect of training with L_2 and L_1 losses. In our paper, we train our model with L_2 loss, following [11]. Inspired by Spade-RGBsD [7] and NConv-CNN [4], we also train our model using the inverse L_1 loss on the depth. Its performances on the test set (from the KITTI official testing server) is shown in Tab. 2. As can be seen, training with the inverse L_1 improves the iRMSE and iMAE, but yields slightly worse RMSE and MAE which are still better than the other methods.

4. Qualitative Results

4.1. Evaluation in Indoor Scenes

We show the qualitative comparison in indoor environments in Fig. 4. We compare our method with NYUv2 dataset [10] to some related methods including bilateral filter using color images (Bilateral) [10], optimization using total variance (TGV) [5], and deep depth completion method for indoor scenes [13]. As can be seen, our method performs the best among all these methods. Bilateral [10] fails when the input depth is not dense enough, which makes sense as it is mostly used to fill in small areas with missing data. TGV [5] is sensitive to bright textures but overlooks dark textures, which yields noisy flat surfaces and over-smooths depth boundaries (e.g., the boundaries of the door and chairs). Zhang *et al*. [13] produces in general good results but is sensitive to the boundary estimation, therefore the results contains noisy boundaries with blocky artifacts.

4.2. More Qualitative Results in Outdoor Scenes

We also show more qualitative results on KITTI validation set [11] in Fig. 5 and Fig. 6. Similar to our main submission, we compare our method to Bilateral [10], Fast (the fast bilateral solver) [1], TGV [5], and Zhang *et al*. [13]. From the highlighted regions, we can see that our method consistently performs better than these methods. From the left examples in both Fig. 5 and Fig. 6, we can see that the mixed LiDAR signals on some near objects (e.g., tree trunk, bicyclist, and traffic sign) cause obvious artifacts in the final dense depth estimation for all the other methods, while our method solves this problem by learning a confidence mask within the network. Moreover, we can also see that our method also outperforms the other methods in distant areas and recovers more details of the objects in the distance. We refer the reader to the highlighted regions for details.

References

[1] J. T. Barron and B. Poole. The fast bilateral solver. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 617–632, 2016.

[2] X. Cheng, P. Wang, and R. Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 108–125, 2018.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proc. of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[4] A. Eldesokey, M. Felsberg, and F. S. Khan. Propagating confidences through cnns for sparse data regression. In *The British Machine Vision Conference (BMVC)*, 2018.

[5] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 993–1000, 2013.

[6] Z. Huang, J. Fan, S. Yi, X. Wang, and H. Li. Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *arXiv preprint arXiv:1808.08685*, 2018.

[7] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, and F. Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *Proc. of International Conf. on 3D Vision (3DV)*, pages 52–60, 2018.

[8] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *Proc. of International Conf. on 3D Vision (3DV)*, pages 239–248, 2016.

[9] F. Ma, G. V. Cavalheiro, and S. Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. *arXiv preprint arXiv:1807.00275*, 2018.

[10] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 746–760, 2012.

[11] J. Uhrig, N. Schneider, L. Schneidre, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. In *Proc. of International Conf. on 3D Vision (3DV)*, 2017.

[12] B. Wang, Y. Feng, and H. Liu. Multi-scale features fusion from sparse lidar data and single image for depth completion. *Electronics Letters*, 2018.

[13] Y. Zhang and T. Funkhouser. Deep depth completion of a single rgb-d image. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 175–185, 2018.

[14] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5287–5295, 2017.

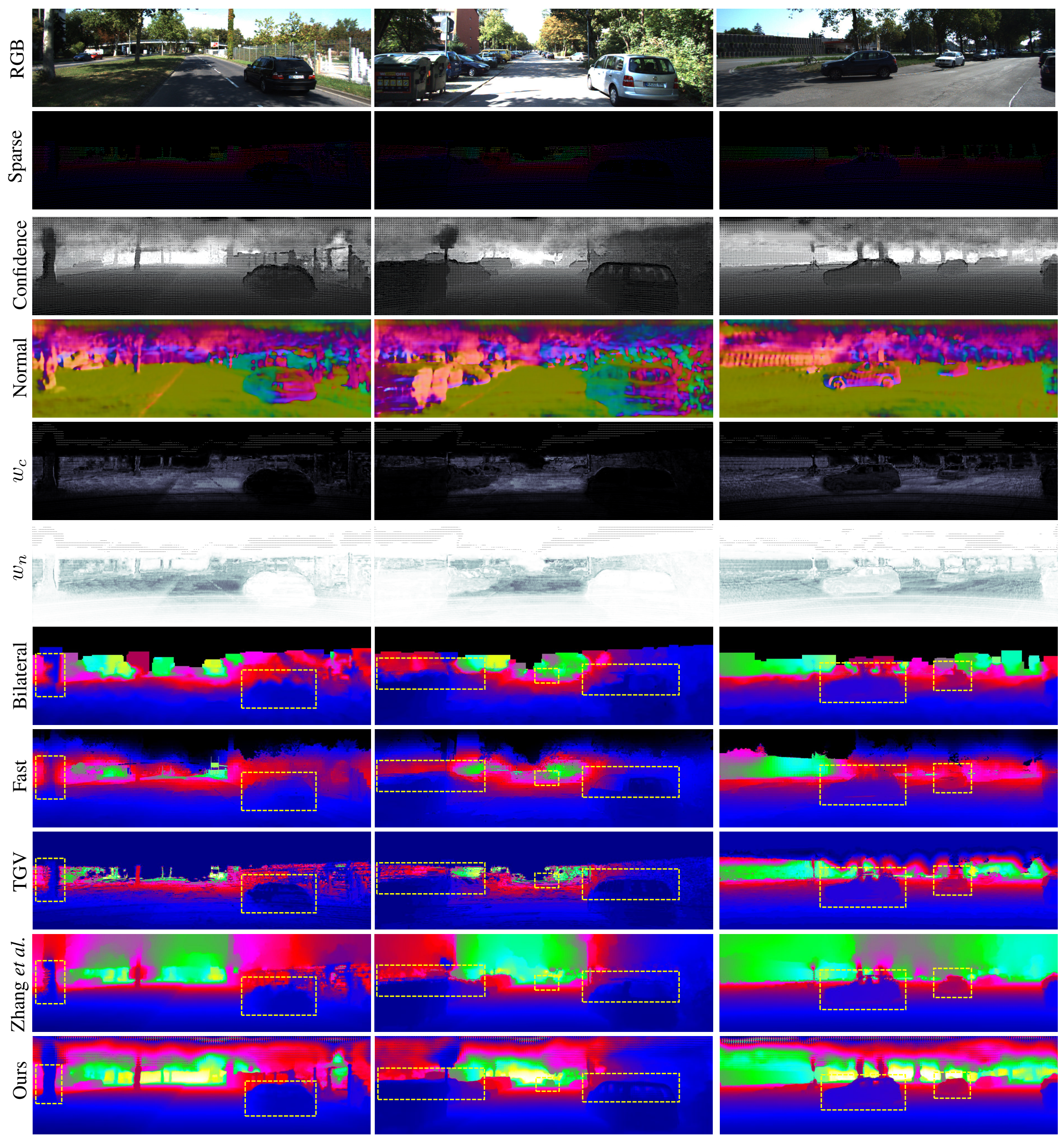


Figure 5. **More qualitative results on KITTI validation set.** From top to bottom are RGB image input, sparse depth input, confidence mask, estimated surface normals, attention map for color pathway, attention map for normal pathway, results of Bilateral [10], Fast [1], TGV [5], Zhang *et al*. [13], and our method. We mark some regions in the results to highlight the difference across methods.

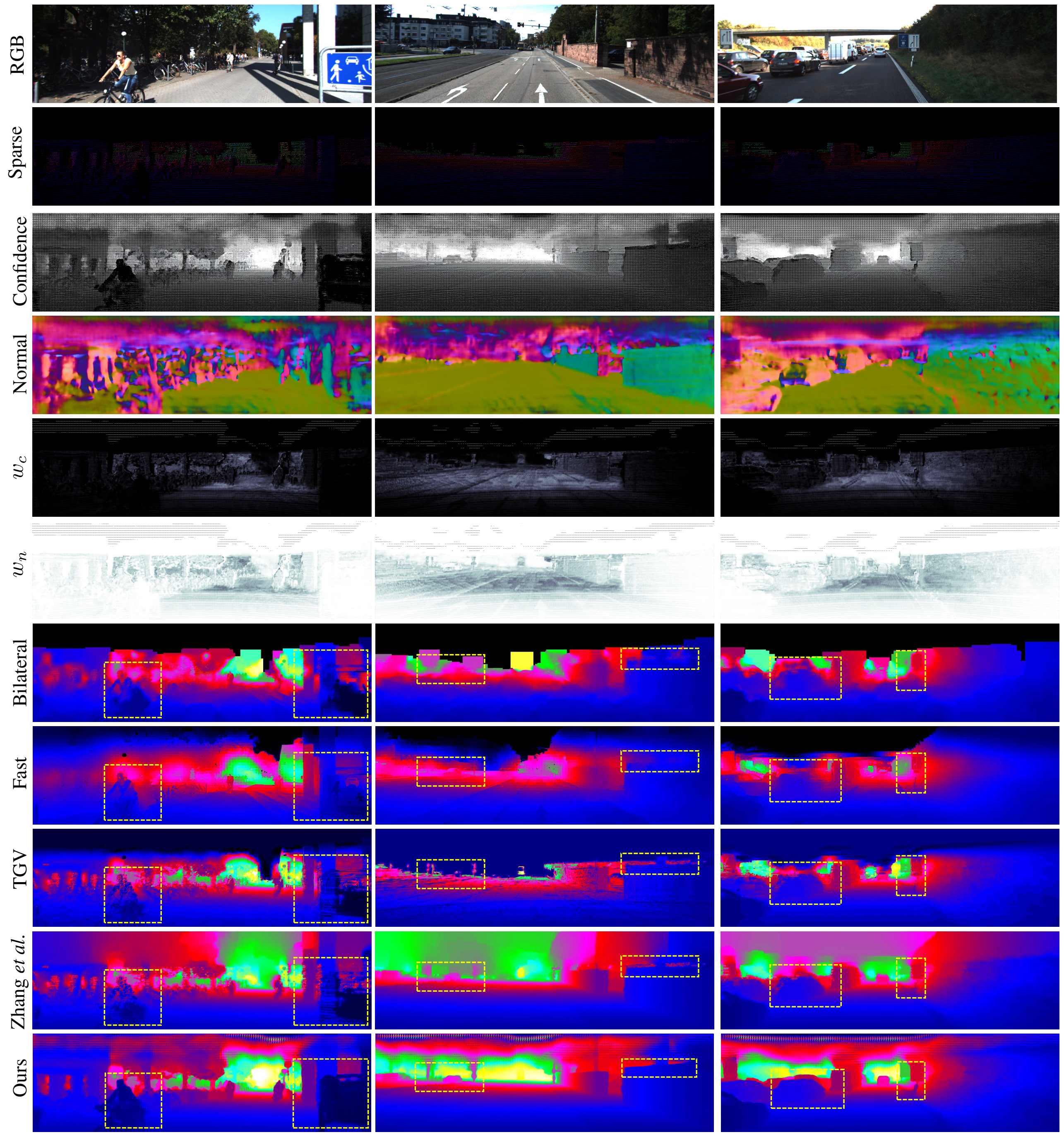


Figure 6. **More qualitative results on KITTI validation set.** From top to bottom are RGB image input, sparse depth input, confidence mask, estimated surface normals, attention map for color pathway, attention map for normal pathway, results of Bilateral [10], Fast [1], TGV [5], Zhang *et al*. [13], and our method. We mark some regions in the results to highlight the difference across methods.