Bilateral Propagation Network for Depth Completion

Supplementary Material

This supplementary material provides additional information to complement the main paper. It includes implementation details of the proposed method in Sec. A, network architecture details in Sec. B, additional training details in Sec. C, descriptions of the adopted evaluation metrics in Sec. D, and further experimental results in Section E.

A. Additional Method Details

A.1. Inverse Projection Implementation

Our method utilizes depth features in both prior encoding (Sec. 3.2.3) and multi-modal fusion (Sec. 3.3) of the main paper. These depth features are obtained by inverse projecting the depth map into camera space. This inverse projection technique has been proven beneficial for extracting 3D cues in a previous study [4]. Specifically, the depth feature map D is transformed from a single-channel depth map to a three-channel feature map, with each pixel coordinate (x, y) represented by (X, Y, Z), and can be written as:

$$\begin{aligned} X_{x,y} &= \tfrac{x-c_x}{f_x} D_{x,y}, \\ Y_{x,y} &= \tfrac{y-c_y}{f_y} D_{x,y}, \\ Z_{x,y} &= D_{x,y}. \end{aligned} \tag{1}$$

Here, c_x, c_y, f_x, f_y are intrinsic parameters of a camera. Our method employs a coarse-to-fine manner for depth estimation, where depth maps are generated at multiple scales. Thus we correspondingly adjust the intrinsic parameters used for depth feature generation at different scales. Specially, for a scale s.

$$\begin{aligned} c_x^s &= \frac{c_x}{2^s}, c_y^s = \frac{c_y}{2^s}, \\ f_x^s &= \frac{f_x}{2^s}, f_y^s = \frac{f_y}{2^s}. \end{aligned} \tag{2}$$

A.2. Image Encoding Implementation

The proposed bilateral propagation module is arranged in a multi-scale scheme, with prior encodings from the corresponding resolution. The image encoding $\mathcal{I}$ in scale s can be written as:

$$\mathcal{I}^s = \begin{cases} \mathbb{C}([\mathbf{I}^s, \mathbb{D}([\mathbf{F}^{s+1}, \mathbf{D}^{s+1}])]), & 0 \le s < 5, \\ \mathbf{I}^s, & s = 5. \end{cases} \tag{3}$$

Here, for the lowest resolution with $s = 5$, we directly adopt the image feature $\mathbf{I}^s$ as image encoding. Otherwise, to make image encoding representative, we concatenate the multi-modal fused feature $\mathbf{F}^{s+1}$ with depth feature $\mathbf{D}^{s+1}$ in scale $s+1$. The depth feature $\mathbf{D}^{s+1}$ is achieved by inverse projecting estimated depth D^{s+1} to camera space. Then we utilize deconvolution operation $\mathbb{D}$ to upsample the concatenated feature map to scale s. Finally, we concatenate the upsampled feature with $\mathbf{I}^s$, and adopt an extra convolution operation $\mathbb{C}$ to produce the image encoding $\mathcal{I}^s$.

A.3. Weighted Pooling Implementation

As explained in Sec. 3.2.3 of the main paper, we employ weighted pooling to downsample the sparse depth map. For a pixel i at under scale s, the downsampled sparse depth map can be represented as:

$$S_i^s = \frac{\sum\limits_{j=\mathcal{N}_s(i)} w_j^s S_j}{\sum\limits_{j=\mathcal{N}_s(i)} w_j^s \mathbb{I}(S_j) + \epsilon}, \tag{4}$$

The weight map w is estimated from image content and generated using an exponential layer to ensure positivity. Thus, Eq. (4) can be explicitly formalized as

$$S_i^s = \frac{\sum\limits_{j=\mathcal{N}_s(i)} e^{\hat{w}_j^s} S_j}{\sum\limits_{j=\mathcal{N}_s(i)} e^{\hat{w}_j^s} \mathbb{I}(S_j) + \epsilon}, \tag{5}$$

where, $\hat{w}$ is the generated weight map before exponential transform. Directly implementing Eq. (5) may have numerical risk on weights generation and gradients calculation. In practice, we adopt an equivalent transformation that

$$\begin{aligned} S_i^s &= \frac{\sum\limits_{j=\mathcal{N}_s(i)} e^{\breve{w}_j^s} S_j}{\sum\limits_{j=\mathcal{N}_s(i)} e^{\breve{w}_j^s} \mathbb{I}(S_j) + \epsilon}, \\ \breve{w}_j^s &= \hat{w}_j^s - \max_{j=\mathcal{N}_s(i)} \hat{w}_j^s. \end{aligned} \tag{6}$$

Here, by reducing the maximum value in $\mathcal{N}_s(i)$ for each pixel j, $\breve{w}_j^s$ is less equal than 0, avoiding the potential numerical stability issue in implementation.

B. Network Architecture

The overview of our BP-Net is depicted in Fig. 2 of the main paper. We show the detailed architecture in Tab. 1 with images of 320×256 as input. Here, symbols are consistent with the main paper, *e.g.* D'^5 denotes the propagated depth map from bilateral propagation module in scale 5. Note that only main operators are listed in this table, and some trivial operations, *e.g.* converting S^4 to $\mathcal{S}^4$ by inverse projection, are omitted for clarity.

Output	Input	Operator	Output Size
$\mathbf{I}^0$	I	Basic2D + ResBlock $\times 2$	(32, 256, 320)
$\mathbf{I}^1$	$\mathbf{I}^0$	ResBlock $\times 2$	(64, 128, 160)
$\mathbf{I}^2$	$\mathbf{I}^1$	ResBlock $\times 2$	(128, 64, 80)
$\mathbf{I}^3$	$\mathbf{I}^2$	ResBlock $\times 2$	(256, 32, 40)
$\mathbf{I}^4$	$\mathbf{I}^3$	ResBlock $\times 2$	(256, 16, 20)
$\mathbf{I}^5$	$\mathbf{I}^4$	ResBlock $\times 2$	(256, 8, 10)
S^0	S	Identity	(1, 256, 320)
S^1	$\mathbf{I}^1, S$	Weighted Pooling	(1, 128, 160)
S^2	$\mathbf{I}^2, S$	Weighted Pooling	(1, 64, 80)
S^3	$\mathbf{I}^3, S$	Weighted Pooling	(1, 32, 40)
S^4	$\mathbf{I}^4, S$	Weighted Pooling	(1, 16, 20)
S^5	$\mathbf{I}^5, S$	Weighted Pooling	(1, 8, 10)
$\mathcal{I}^5$	$\mathbf{I}^5$	Identity	(256, 8, 10)
D'^5	$\mathcal{I}^5, \mathcal{S}^5, \mathcal{O}^5$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 8, 10)
$\mathbf{F}^5$	$\mathcal{I}^5, \mathcal{S}^5$	*MF.* Basic2D + ResBlock $\times 2 \times 1$	(256, 8, 10)
D''^5	$\mathbf{F}^5, D'^5$	Conv + Add	(1, 8, 10)
D^5	S^5, D''^5	*Post.* Conv $\times 3 \times 2$	(1, 8, 10)
$\mathcal{I}^4$	$\mathbf{I}^4, \mathbf{F}^5, D^5$	Deconv + Conv	(256, 16, 20)
D'^4	$\mathcal{I}^4, \mathcal{S}^4, \mathcal{O}^4$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 16, 20)
$\mathbf{F}^4$	$\mathcal{I}^4, \mathcal{S}^4$	*MF.* Basic2D + ResBlock $\times 2 \times 2$	(256, 16, 20)
D''^4	$\mathbf{F}^4, D'^4$	Conv + Add	(1, 16, 20)
D^4	S^4, D''^4	*Post.* Conv $\times 3 \times 4$	(1, 16, 20)
$\mathcal{I}^3$	$\mathbf{I}^3, \mathbf{F}^4, D^4$	Deconv + Conv	(256, 32, 40)
D'^3	$\mathcal{I}^3, \mathcal{S}^3, \mathcal{O}^3$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 32, 40)
$\mathbf{F}^3$	$\mathcal{I}^3, \mathcal{S}^3$	*MF.* Basic2D + ResBlock $\times 2 \times 3$	(256, 32, 40)
D''^3	$\mathbf{F}^3, D'^3$	Conv + Add	(1, 32, 40)
D^3	S^3, D''^3	*Post.* Conv $\times 3 \times 6$	(1, 32, 40)
$\mathcal{I}^2$	$\mathbf{I}^2, \mathbf{F}^3, D^3$	Deconv + Conv	(128, 64, 80)
D'^2	$\mathcal{I}^2, \mathcal{S}^2, \mathcal{O}^2$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 64, 80)
$\mathbf{F}^2$	$\mathcal{I}^2, \mathcal{S}^2$	*MF.* Basic2D + ResBlock $\times 2 \times 4$	(128, 64, 80)
D''^2	$\mathbf{F}^2, D'^2$	Conv + Add	(1, 64, 80)
D^2	S^2, D''^2	*Post.* Conv $\times 3 \times 8$	(1, 64, 80)
$\mathcal{I}^1$	$\mathbf{I}^1, \mathbf{F}^2, D^2$	Deconv + Conv	(64, 128, 160)
D'^1	$\mathcal{I}^1, \mathcal{S}^1, \mathcal{O}^1$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 128, 160)
$\mathbf{F}^1$	$\mathcal{I}^1, \mathcal{S}^1$	*MF.* Basic2D + ResBlock $\times 2 \times 5$	(64, 128, 160)
D''^1	$\mathbf{F}^1, D'^1$	Conv + Add	(1, 128, 160)
D^1	S^1, D''^1	*Post.* Conv $\times 3 \times 10$	(1, 128, 160)
$\mathcal{I}^0$	$\mathbf{I}^0, \mathbf{F}^1, D^1$	Deconv + Conv	(32, 256, 320)
D'^0	$\mathcal{I}^0, \mathcal{S}^0, \mathcal{O}^0$	*Pre.* (Linear + BN + GeLU) $\times 4$	(1, 256, 320)
$\mathbf{F}^0$	$\mathcal{I}^0, \mathcal{S}^0$	*MF.* Basic2D + ResBlock $\times 2 \times 6$	(32, 256, 320)
D''^0	$\mathbf{F}^0, D'^0$	Conv + Add	(1, 256, 320)
D^0	S^0, D''^0	*Post.* Conv $\times 3 \times 12$	(1, 256, 320)

Table 1. Detailed Architecture of BP-Net.

C. Additional Training Details

When training on KITTI dataset, we randomly crop image to 256×1216 for training. Following previous works [11, 15], we adopt random horizontal flip, color jitter, and normalization as data augmentation. When training on NYUv2 dataset, we follow data augmentation in previous works [9, 15], including random horizontal flip, random crop, random rotation, random resize, color jitter and normalization. We apply data augmentation on color image and sparse depth map, and adjust the camera intrinsic parameters correspondingly.

D. Details on Evaluation Metrics

We verify our method on both indoor and outdoor scenes with standard evaluation metrics. For indoor scene, root mean squared error (RMSE), mean absolute relative error (REL), and δ_θ are chosen as evaluation metrics. For outdoor scene, the standard evaluation metrics are root mean squared error (RMSE), mean absolute error (MAE), root mean squared error of the inverse depth (iRMSE), and mean absolute error of the inverse depth (iMAE). These evaluation metrics are firstly calculated on each sample and then averaged among samples. And for each sample, they can be written as:

$$\begin{aligned} RMSE &= \left(\tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}(D_i^{gt} - D_i)^2\right)^{\frac{1}{2}}, \\ iRMSE &= \left(\tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}\left(\tfrac{1}{D_i^{gt}} - \tfrac{1}{D_i}\right)^2\right)^{\frac{1}{2}}, \\ MAE &= \tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}\left|D_i^{gt} - D_i\right|, \\ iMAE &= \tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}\left|\tfrac{1}{D_i^{gt}} - \tfrac{1}{D_i}\right|, \\ REL &= \tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}\frac{\left|D_i^{gt} - D_i\right|}{D_i^{gt}}, \\ \delta_\theta &= \tfrac{1}{n}\textstyle\sum_{i\in\mathcal{P}_v}\left|\left\{max\left(\tfrac{D_i^{gt}}{D_i}, \tfrac{D_i}{D_i^{gt}}\right) < \theta\right\}\right|. \end{aligned} \tag{7}$$

Here, $\mathcal{P}_v$ is the set of pixels with valid ground truth, and $n = |\mathcal{P}_v|$ is the size of the set.

E. Additional Experimental Results

Due to space limitation, we only show limited comparison results in Tab. 1 and Fig. 5 of the main paper. Here, we list more performance evaluation on outdoor scene and indoor scene in Tab. 2 and Tab. 3 respectively. We also show more qualitative results on outdoor scene and indoor scene in Fig. 1 and Fig. 2 respectively.

References

[1] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *ICCV*, pages 10023–10032, 2019. 5

[2] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *IEEE TPAMI*, 42(10):2361–2379, 2019. 5

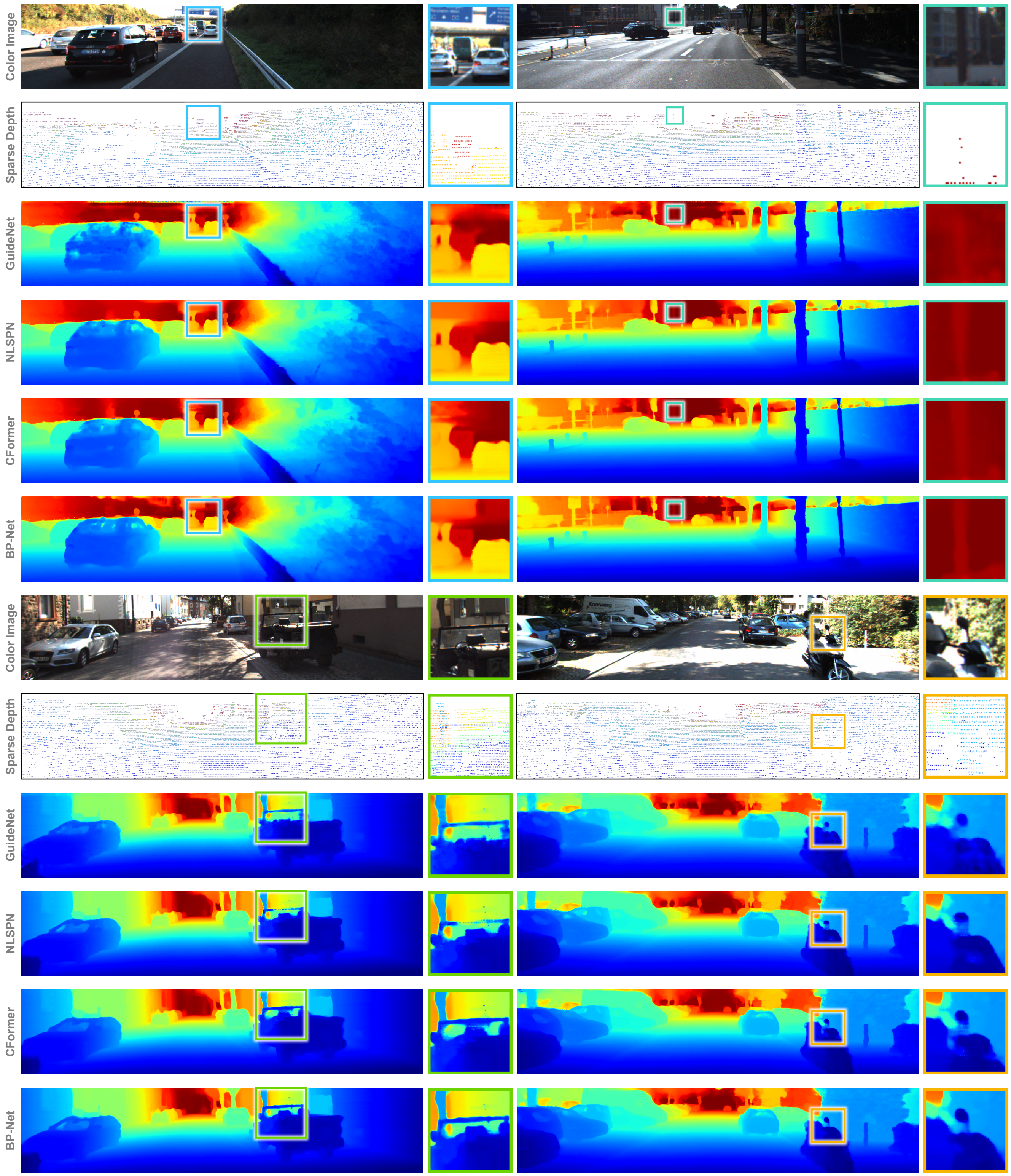

Figure 1. **Additional qualitative results on KITTI dataset.**

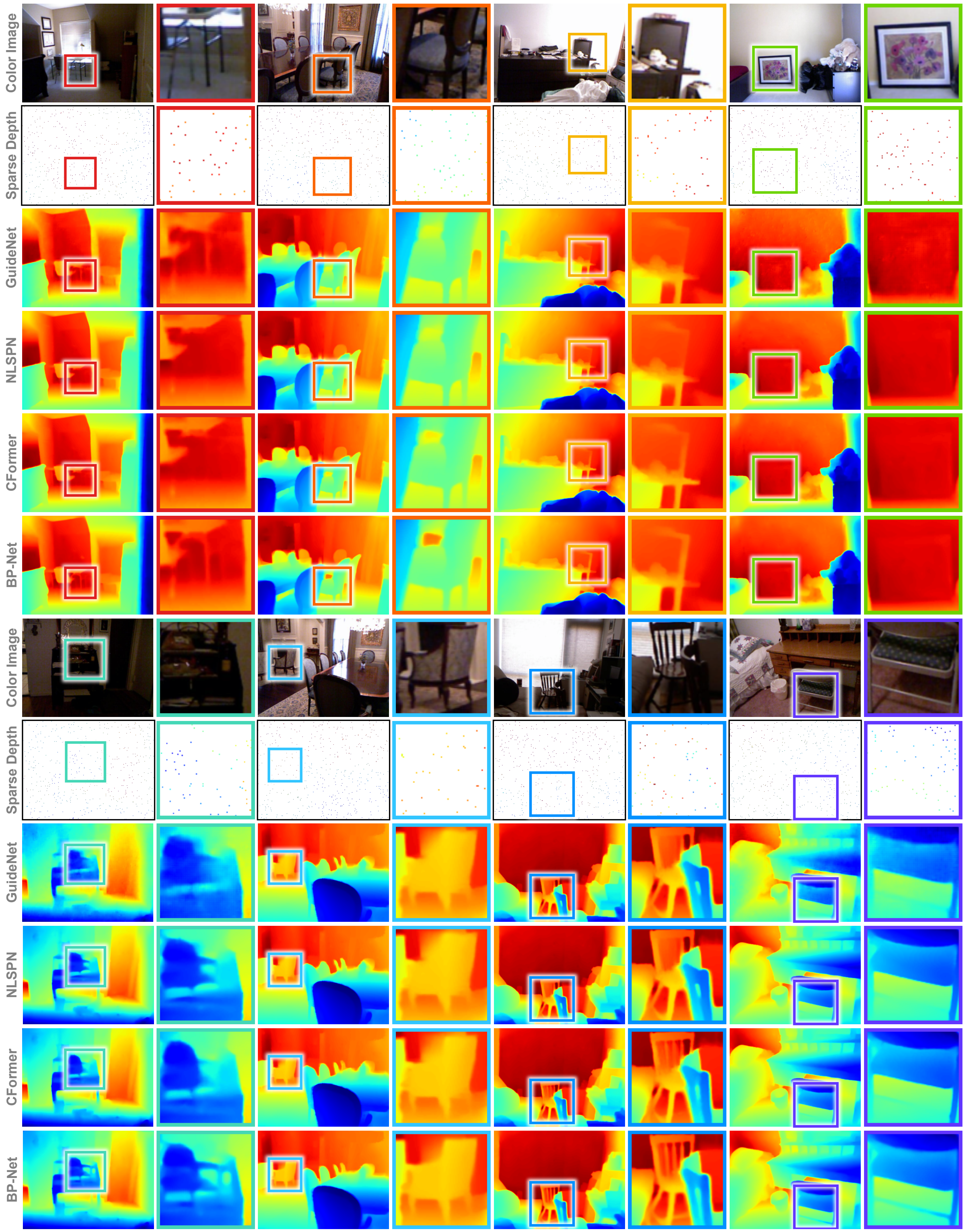

Figure 2. **Additional qualitative results on NYUv2 dataset.**

	RMSE↓ (mm)	MAE↓ (mm)	iRMSE↓ ($1/km$)	iMAE↓ ($1/km$)
S2D [8]	814.73	249.95	2.80	1.21
CSPN [2]	1019.64	279.46	2.93	1.15
DeepLiDAR [10]	758.38	226.50	2.56	1.15
FuseNet [1]	752.88	221.19	2.34	1.14
CSPN++ [3]	743.69	209.28	2.07	0.90
GuideNet [11]	736.24	218.83	2.25	0.99
FCFR [4]	735.81	217.15	2.20	0.98
ACMNet [16]	744.91	206.09	2.08	0.90
NLSPN [9]	741.68	199.59	1.99	0.84
PENet [4]	730.08	210.55	2.17	0.94
RigNet [14]	712.66	203.25	2.08	0.90
DySPN [6]	709.12	192.71	1.88	0.82
BEV@DC [17]	697.44	189.44	1.83	0.82
CFormer [15]	708.87	203.45	2.01	0.88
LRRU [12]	696.51	**189.96**	1.87	**0.81**
BP-Net	**684.90**	194.69	**1.82**	0.84

Table 2. **Performance on KITTI dataset.** Results are evaluated by the KITTI testing server and ranked by the RMSE (in mm). The best result under each criterion is in **bold**.

	RMSE↓ (m)	REL↓	$\delta_{1.25}\uparrow$ (%)	$\delta_{1.25^2}\uparrow$ (%)	$\delta_{1.25^3}\uparrow$ (%)
S2D [8]	0.230	0.044	97.1	99.4	99.8
CSPN [2]	0.117	0.016	99.2	**99.9**	**100.0**
DeepLiDAR [10]	0.115	0.022	99.3	**99.9**	**100.0**
CSPN++ [3]	0.115	–	–	–	–
DepthNormal [13]	0.112	0.018	99.5	**99.9**	**100.0**
GuideNet [11]	0.101	0.015	99.5	**99.9**	**100.0**
FCFR [4]	0.106	0.015	99.5	**99.9**	**100.0**
ACMNet [16]	0.105	0.015	99.4	**99.9**	**100.0**
TWISE [5]	0.097	0.013	**99.6**	**99.9**	**100.0**
NLSPN [9]	0.092	0.012	**99.6**	**99.9**	**100.0**
RigNet [14]	0.090	0.013	**99.6**	**99.9**	**100.0**
DySPN [6]	0.090	0.012	**99.6**	**99.9**	**100.0**
GraphCSPN [7]	0.090	0.012	**99.6**	**99.9**	**100.0**
BEV@DC [17]	**0.089**	0.012	**99.6**	**99.9**	**100.0**
CFormer [15]	0.090	0.012	–	–	–
LRRU [12]	0.091	**0.011**	**99.6**	**99.9**	**100.0**
BP-Net	**0.089**	0.012	**99.6**	**99.9**	**100.0**

Table 3. **Performance on NYUv2 datasets.** For the NYUv2 dataset, authors report their performance on the official test set in their papers. The best result under each criterion is in **bold**.

[3] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *AAAI*, pages 10615–10622, 2020. 5

[4] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *ICRA*, pages 13656–13662. IEEE, 2021. 1, 5

[5] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2583–2592, 2021. 5

[6] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *AAAI*, pages 1638–1646, 2022. 5

[7] Xin Liu, Xiaofei Shao, Bo Wang, Yali Li, and Shengjin Wang. Graphcspn: Geometry-aware depth completion via dynamic gcns. In *ECCV*, pages 90–107. Springer, 2022. 5

[8] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *ICRA*, pages 4796–4803. IEEE, 2018. 5

[9] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *ECCV*, pages 120–136. Springer, 2020. 2, 5

[10] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *CVPR*, pages 3313–3322, 2019. 5

[11] Jie Tang, Fei-Peng Tian, Wei Feng, Jian Li, and Ping Tan. Learning guided convolutional network for depth completion. *IEEE TIP*, 30:1116–1129, 2020. 2, 5

[12] Yufei Wang, Bo Li, Ge Zhang, Qi Liu, Tao Gao, and Yuchao Dai. Lrru: Long-short range recurrent updating networks for depth completion. In *ICCV*, pages 9422–9432, 2023. 5

[13] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *ICCV*, pages 2811–2820, 2019. 5

[14] Zhiqiang Yan, Kun Wang, Xiang Li, Zhenyu Zhang, Jun Li, and Jian Yang. Rignet: Repetitive image guided network for depth completion. In *ECCV*, pages 214–230. Springer, 2022. 5

[15] Youmin Zhang, Xianda Guo, Matteo Poggi, Zheng Zhu, Guan Huang, and Stefano Mattoccia. Completionformer: Depth completion with convolutions and vision transformers. In *CVPR*, pages 18527–18536, 2023. 2, 5

[16] Shanshan Zhao, Mingming Gong, Huan Fu, and Dacheng Tao. Adaptive context-aware multi-modal network for depth completion. *IEEE TIP*, 30:5264–5276, 2021. 5

[17] Wending Zhou, Xu Yan, Yinghong Liao, Yuankai Lin, Jin Huang, Gangming Zhao, Shuguang Cui, and Zhen Li. Bev@dc: Bird's-eye view assisted training for depth completion. In *CVPR*, pages 9233–9242, 2023. 5