Unsupervised Depth Completion with Calibrated Backprojection Layers SUPPLEMENTARY MATERIALS

Alex Wong
UCLA Vision Lab
alexw@cs.ucla.edu

Stefano Soatto
UCLA Vision Lab
soatto@cs.ucla.edu

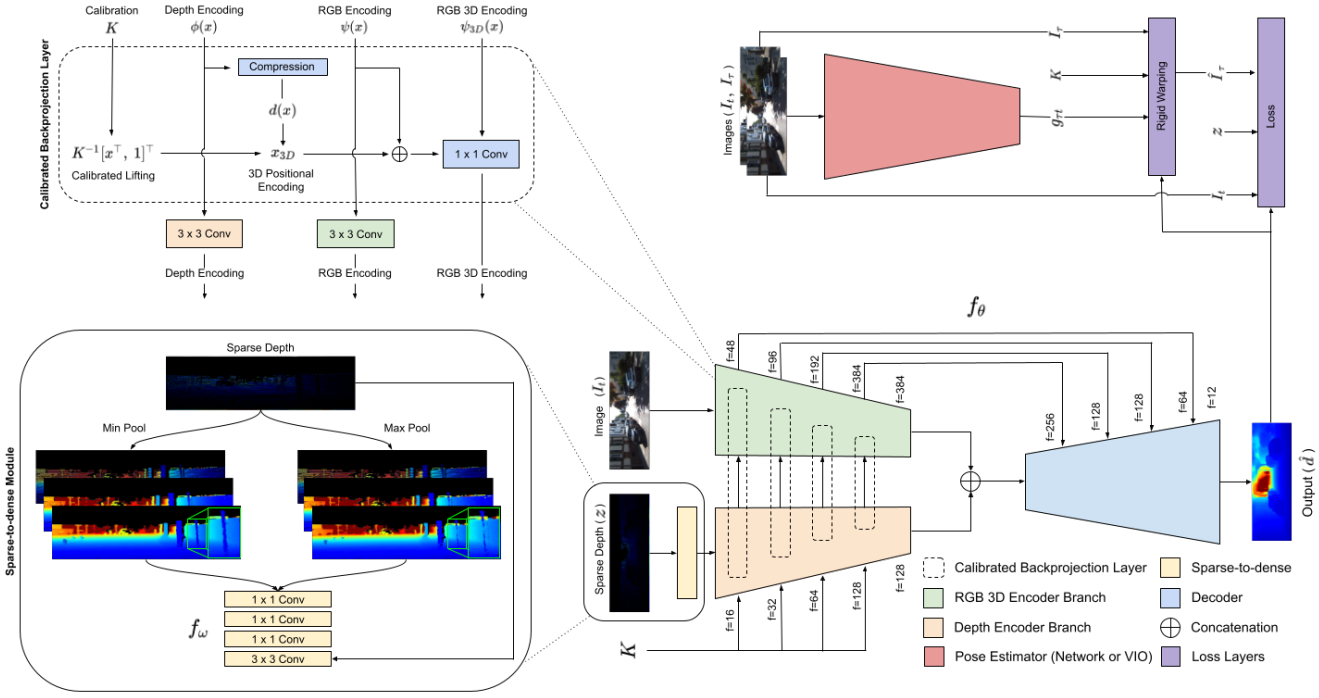

Figure 1: *System diagram during training*. We assume we are given monocular video sequences, synchronized sparse point clouds projected onto the image plane as 2.5D depth maps, and camera calibration. A training sample is therefore (I_t, I_τ, z, K). Sparse depth inputs (z) are fed to our sparse-to-dense module (f_ω) to yield a dense or quasi-dense representation. Along with image (I_t) and camera calibration matrix (K), it is then fed into our depth completion network (f_θ) comprised of calibrated backprojection layers to produce dense depth prediction $\hat{d}$. Relative pose $(g_{\tau t})$ between images I_t and I_τ can be estimated from a VIO or a network. In the case of the latter, pose can be jointly learned with depth. We note that pose is only needed to give the reconstruction $\hat{I}_\tau$ for constructing the loss function and is not needed during inference.

Code available at: https://github.com/alexklwong/calibrated-backprojection-network.

Summary of contents. In Sec. 1, we provide an overview of our full system and more details on our loss function. We also provide the kernel sizes used in our sparse-to-dense module, augmentations used during training and our learning rate schedule to reproduce our results on KITTI [19], VOID [23], and NYUv2 [18]. In Sec. 2, we visualize and compare features learned by our proposed sparse-to-dense module to those from typical convolutional block, and show that our spare-to-dense module yields a much denser representation for the the depth completion network to ingest. In Sec. 3.1, we consider the possibility of miscalibration and examine the sensitivity of our model to changes in intrinsics parameters i.e. *incorrect* calibration. We show that our model is robust to reasonable ranges of calibration error. In

Sec. 3.2, we study the sensitivity of our model to changes in sparse depth input density levels and demonstrate that we are robust even when sparse point cover only 0.15% of the image space. Finally, in Sec. 5, we show that we can beat several supervised methods on KITTI online leaderboard and that we rank 5th amongst *all methods* for the iMAE metric.

1. System Overview

Fig. 1 shows a diagram of our full system. Our model takes an RGB image I, a sparse depth map z, and the camera intrinsics matrix K as input. First, the sparse depth map z is fed into our sparse-to-dense module f_ω to obtain a dense or qusai-dense representation (Sec. 2.1, main text). Then, the depth representation $f_\omega(z)$, RGB image I, and intrinsics K are fed into the depth completion network f_θ, which is comprised of an encoder with calibrated backprojection layer followed by a decoder (Sec. 2.2, main text). Each calibrated backprojection realizes the backprojection process into 3D camera space by performing calibrated lifting of pixel coordinates using K, and projecting the depth representation to 1 dimension and multiplying it with the lifted coordinates – result of which is a 3D positional encoding of the scene structure.

To yield a unified depth and RGB representation, the 3D positional encoding from the depth branch is passed laterally to the RGB branch to enable association between each RGB feature and its 3D position. By doing so, we introduce 3D structure as an architectural inductive bias, which allows the network to reason about "close" points in the 2D image topology that are actually far in 3D scene topology. The RGB 3D representation is finally fed through the decoder to produce the final depth prediction $\hat{d}$.

1.1. Loss Function

To train our model, we assume the availability of previous and next RGB frames I_τ of the given image I or I_t (to denote the current time frame) where $\tau \in T \doteq \{t-1, t+1\}$. During training, we estimate the relative pose $g_{\tau t}$ between images at time t and τ. Using I_τ, K and $g_{\tau t}$, we can create the reconstruction $\hat{I}_t$ of I_t via reprojection (Eqn. 4, main text) to enable an unsupervised loss (Eqn. 6-9, main text), which include a photometric reconstruction term, a sparse depth reconstruction term and a local smoothness term.

We note that the photometric term can be replaced with more sophisticated measures of reprojection error [8] and additional regularizers such as pose consistency [23] or adaptive regularization weighting schemes [24, 22] – which would likely boost performance even more. However, we choose a simple loss to demonstrate the efficacy of our novel architecture. We note that $g_{\tau t}$ can be obtained by the means of a visual inertial odometry (VIO) system or a pose network if the VIO is not available. In the case where pose is obtained from network, the pose network can be trained jointly with our depth completion network (KBNet). Relative pose is learned as a byproduct of minimizing Eqn. 6 in main text. Also, since $g_{\tau t}$ is only need for reprojection during training; hence, the VIO system and the pose network are not necessary for inference. Because our network is fast and light-weight (16ms run time per image, 6.9M parameters and 2.6GB memory as benchmarked on 1216×352 images from KITTI [19]), it can be deployed with a VIO system to learn online.

Epochs	Learning Rate
KITTI	
0 to 2	5×10^{-5}
2 to 8	1×10^{-4}
8 to 20	1.5×10^{-4}
20 to 30	1×10^{-4}
30 to 45	5×10^{-5}
45 to 50	2×10^{-5}
VOID	
0 to 10	1×10^{-4}
10 to 15	5×10^{-5}
NUYv2	
0 to 10	1×10^{-4}
10 to 15	5×10^{-5}

Table 1: *Learning schedule for KITTI, VOID, and NYUv2.*

Dataset	Min Pool	Max Pool
KITTI [19]	5, 7, 9, 11, 13	15, 17
VOID [23]	15, 17	23, 27, 29
NYUv2 [18]	15, 17	23, 27

Table 2: *Min pool and max pool kernel sizes for our sparse-to-dense module* Kernel sizes for VOID [23] and NYUv2 [18] are larger because the point cloud generated from VIO [6] is much sparser than that of LIDAR in KITTI [19].

1.2. Implementation and Training Details

We optimized our networks using Adam [10] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We trained for a total of 50 epochs on KITTI [19], 15 epochs on VOID [23], and 15 epochs on NYUv2 [18]. We use a batch size of 8 with 768×320 crops for KITTI, 640×480 for VOID and 576×416 for NYUv2. For KITTI, we choose $w_{ph} = 1$, $w_{co} = 0.15$, $w_{st} = 0.95$, $w_{sz} = 0.6$, and $w_{sm} = 0.04$; for VOID and NYUv2, we set $w_{sz} = 2$ and $w_{sm} = 2$. Kernel sizes for our sparse-to-dense (S2D) module are shown in Table 2 for each dataset. We detail our learning rate schedule for each dataset in Table 1.

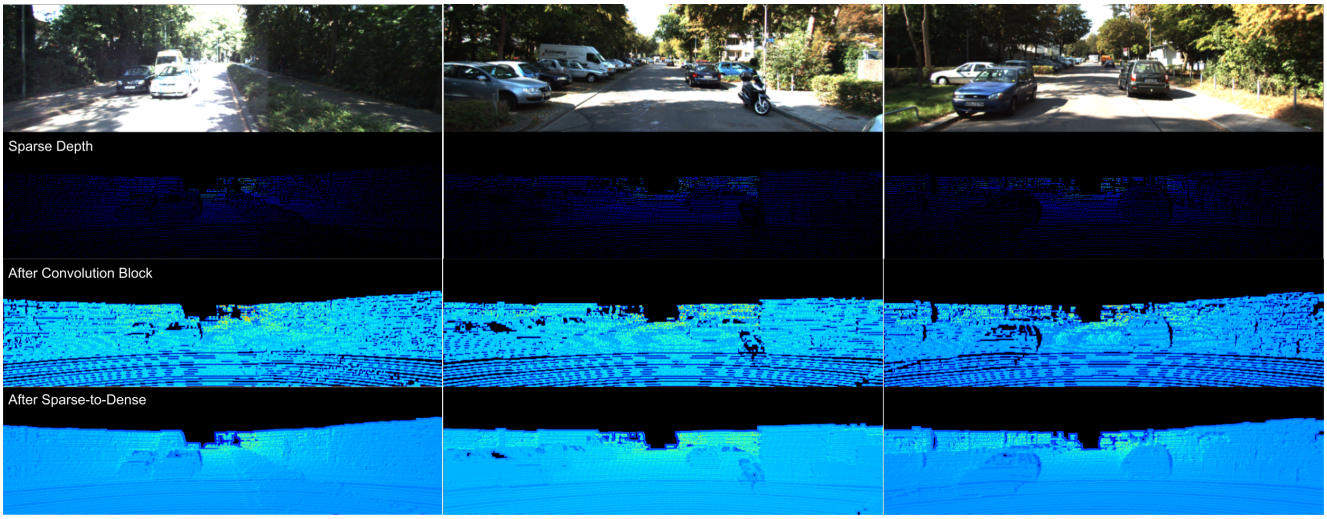

Figure 2: *Visualization of depth features.* Row 3: "After Convolution Block" denotes the depth features produced by a typical first convolutional block used by [13, 23, 26] *without* any form of densification. Row 4: "After Sparse-to-Dense" denotes the depth features learned by the proposed sparse-to-dense (S2D) module. Those learned without our module are still sparse; whereas S2D produces a dense or quasi-dense representation before it reaches the depth completion network. This alleviates the network from having to densify or propagate the sparse signal, making the overall architecture more efficient.

For data augmentations on KITTI, we performed random horizontal shifts to the image and depth map and randomly removed between 60% to 70% of the sparse points. For VOID and NYUv2, we randomly removed 30% to 60% of the sparse points. Augmentations are enabled 100% of the time throughout training and each augmentation has a 50% probability of being applied.

2. Features Learned by Sparse-to-Dense

In Sec 2.1 of the main text, we proposed a sparse-to-dense module (S2D) to learn a dense or quasi-dense representation of the sparse depth inputs. S2D utilizes a series of min and max pooling layers of various kernel sizes to densify the sparse depth inputs (for a list of kernel sizes used for each dataset, please see Table 2). To balance the trade-off between density and detail (large vs. small kernel sizes), and near and far structures (min vs. max pooling), we concatenate the pooled results and learn three 1×1 convolutions. The output of which is fused with the input sparse depth using a 3×3 convolution to "fill in the gaps".

Fig. 2 shows visualizations of features learned by S2D and a comparison to the features learned by typical convolutional e.g. ResNet or VGG blocks used by [13, 23, 26]. Row 2 of Fig. 2 shows that despite passing through several convolutional layers ($\approx 10Kto20K$ parameters), the representation obtained by a typical convolution block is still sparse; so the later layers will still have many zero-activations and must continue to densify the features. In contrast, using our proposed S2D (≈ 900 parameters), the depth representation learned is dense or quasi-dense before reaching the depth completion network (row 3). This enables non-zero activations in the later layers, which allows the network to use its early convolutions for learning scene geometry rather than densification.

We note that our sparse-to-dense module may bare some resemblance to Spatial Pyramid Pooling (SPP) employed in classification [9] or stereo matching [1]. However, we note that [9] used SPP with max pooling to ensure that feature map sizes are consistent for different input sizes. [1] used average pooling to increase receptive field. Both use cases are intended for dense input. We discussed the drawbacks of max pooling [9] in Sec. 2.1 of the main text and showed in Table 3 of main text that SPP underperforms compare to our S2DM. Also we note that using average pooling [1] will destroy the signal because the kernel will convolve and average over mostly zeros. The work that is most similar to our S2D module is the SPP for depth completion proposed by [21]. However, [21] only uses max pooling which decimates the detail of nearby structures.

3. Sensitivity Studies

In this section, we provide additional studies to quantify the sensitivity of our model to incorrect calibration and various sparse depth density levels.

3.1. To Incorrect Calibration

We showed in Table 5 of the main text that our method generalizes well when given the *correct* calibration at test time. To consider the scenario of a miscalibrated camera, we studied the sensitivity of our model to *incorrect* calibra-

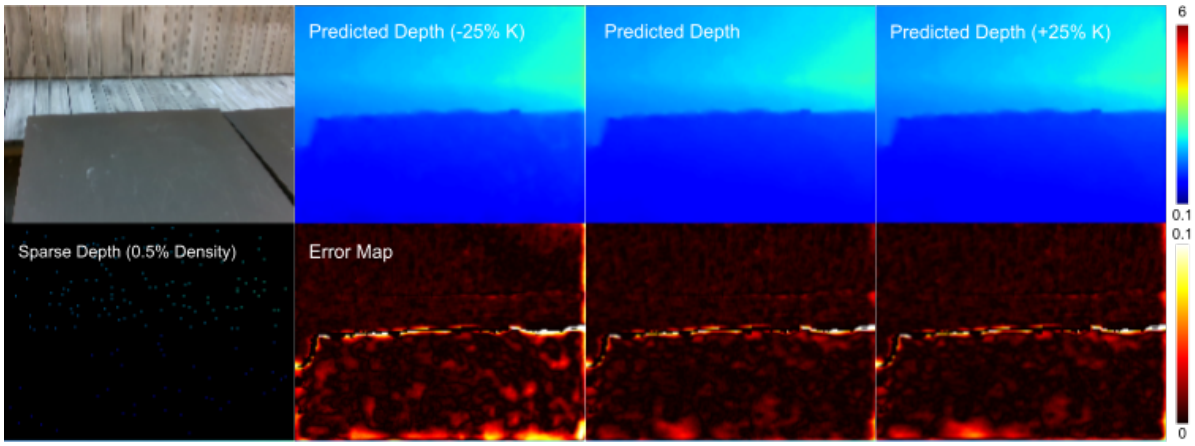

Figure 3: *Visualization of predicted depth for incorrect calibration.* -25% K denotes 25% decrease to intrinsics parameters and +25% K denotes 25% increase. Overall error in -25% is increased (slight brigher shade of red). Larger errors caused by incorrect intrinsics is generally located at the edge of the depth map. +25% have little effect on our predictions. This is because decreasing focal length causes surfaces to be distorted, which in turn affect depth predictions. On the other hand, increasing focal length packs points closer together, which is less detrimental in comparison.

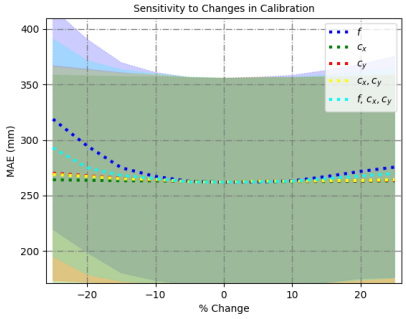

Figure 4: *Sensitivity to changes in calibration on KITTI.* Focal length and principal point are altered to test sensitivity to changes in intrinsics parameters. Our method is robust to change up to $\approx$ 10% change. After which, performance degrades. We note that changes in principal point (c_x, c_y) have little effect; whereas decreasing focal length (f) causes large drop in performance.

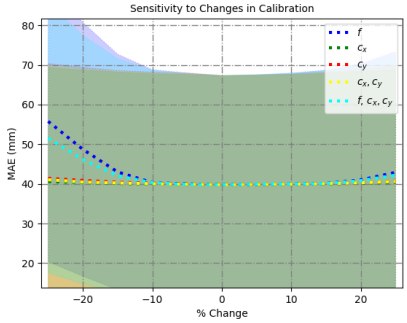

Figure 5: *Sensitivity to changes in calibration on VOID.* Focal length and principal point are altered to test sensitivity to changes in intrinsics parameters. Our method is robust to change up to $\approx$ 10% change. After which, performance degrades. We note that changes in principal point (c_x, c_y) have little effect; whereas decreasing focal length (f) causes large drop in performance.

tion on the KITTI dataset (outdoor scenarios) in Fig. 5 in the main text (also here in Fig. 4). Now, we further extend the sensitivity study to the indoor setting by conducting a similar sensitivity study on the VOID dataset (Fig. 5). To this end, we consider changes to the focal length (f) and principal point (c_x, c_y) parameters to create erroneous intrinsic calibration matrices for input to a pretrained model on VOID.

The overall trend for indoor setting, (VOID, Fig. 5) is similar to that of outdoor setting (KITTI, Fig. 4). For both indoors and outdoors, our model is robust to changes in principal point parameters (c_x, c_y) – increasing or decreasing them by up to 25% has little effect on performance. This is because these parameters shifts the optical center so they do not affect the overall structure of the scene. We note that for large values outside of reasonable perturbation range will cause the performance to decrease.

Unlike its behavior with changes in the principal point, the model degrades when focal length (f) is decreased. For both indoors and outdoors, we are robust up to 10% decrease in focal length, after which error will increase. We note that the performance drop is asymmetric, our model is robust to increases in focal length up to 20%. The reason for this phenomeon is as follows: Geometrically, decreases in focal length will cause points to backproject to a wider field of view, which distorts surfaces by pushing points that belong to the same surface far from each other. On the other hand, increases in focal length will cause points to

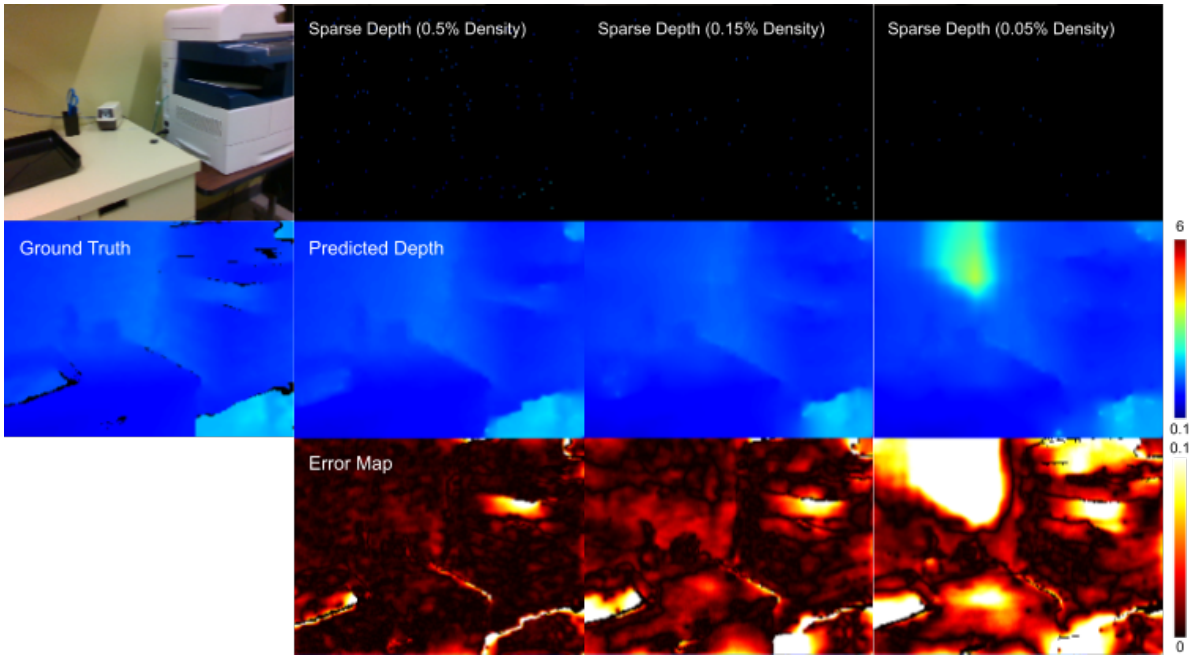

Figure 6: *Visualization of predicted depth for various density levels on VOID.* Columns 1, 2: Our method works well for density levels of 0.5% and 0.15%. Column 3: The quality of predicted depth begins to degrade in far homogeneous regions where there are no sparse points e.g. wall when density level drops to 0.05%.

pack tighter together. This does not disrupt the scene structure for small values, but for large values, points will get squashed together; this is demonstrated by the small uptick in error when increasing focal length by 20 to 25%.

We note that these values are well out of the typical range of calibration error and should not be of concern. For example when using off-the-shelf calibration packages that implements [27] to calibrate our camera, we obtained a standard error of $\approx 0.6\%$, which yields $\pm \approx 1.1\%$ margin of error for a 95% confidence interval. Nonetheless, there exists the risk of using the wrong calibration; however, we believe this trade-off is well worth the performance boost provided by the proposed architecture.

Fig. 3 shows a visualization of depth predicted by our model when using erroneous calibration. -25% K denotes a 25% decrease to focal length and principal point and +25% K denotes a 25% increase to both. As we can see, the larger errors are typically located along the border of the predicted depth map; there is also a slight increase in error (brighter shade of red) for the entire scene. Increasing intrinsics by 25% affects the output less significantly, but nonetheless we observe an increase in errors.

3.2. To Various Density Levels

In Table 3, we consider three different levels of density for the sparse depth inputs, 0.50%, 0.15%, 0.05% of the image space, that are provided by the VOID dataset [23]. To this end, we train a *single* model on VOID using sparse depth maps of 0.50% density and evaluate it on 0.50%, 0.15%, 0.05% density test sets. We also compare our method against [21, 23] under these density levels.

Method	MAE	RMSE	iMAE	iRMSE
		0.50% Density		
VOICED [23]	85.05	169.79	48.92	104.02
ScaffNet [21]	59.53	119.14	35.72	68.36
Ours	**39.80**	**95.86**	**21.16**	**49.72**
		0.15% Density		
VOICED [23]	124.11	217.43	66.95	121.23
ScaffNet [21]	108.44	195.82	57.52	103.33
Ours	**77.70**	**172.49**	**38.87**	**85.59**
		0.05% Density		
VOICED [23]	179.66	281.09	95.27	151.66
ScaffNet [21]	150.65	**255.08**	80.79	133.33
Ours	**131.54**	263.54	**66.84**	**128.29**

Table 3: *Sensitivity study on various sparse depth density levels on VOID.* We train a single model on VOID using sparse depth maps of 0.50% density and evaluate it on 0.50%, 0.15%, 0.05% density test sets. As expected, performance degrade as the input become more sparse. Overall, we perform better than [21, 23]; however, at 0.05%, [21] performs better on the RMSE metric.

As expected, as density decreases, our performance also

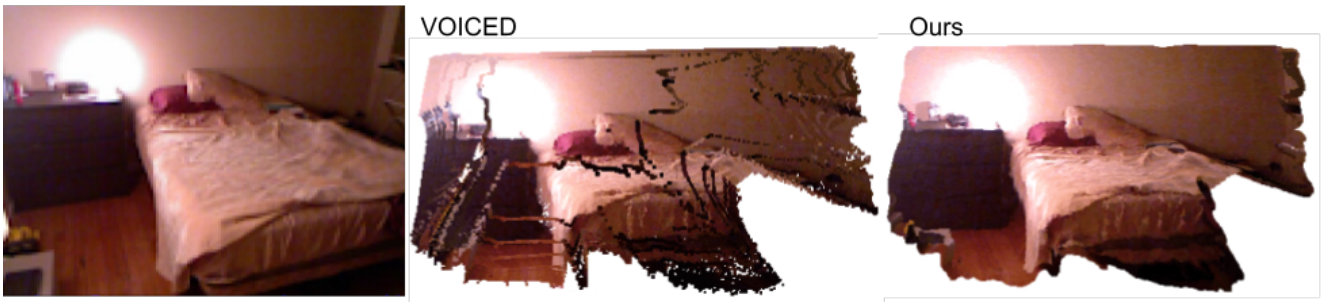

Figure 7: *Qualitative results on generalization*. We trained our model on VOID [23] (captured by Intel RealSense) and tested the model on NYUv2 [18] (captured by Microsoft Kinect). We also trained the baseline [23] on VOID and tested it on NYUv2. Here we show the point clouds in 3D and colored. Because of the mismatch in cameras, [23] predicted a distorted scene; whereas, while ours is not perfect, it is reasonable.

degrades. However, we still outperform both [21, 23] under all three levels. We note that at the sparsest setting of 0.05%, [21] does beat us on the RMSE metric. The reason for this is that we selected the kernel sizes for our model based on the sparsity level of 0.5%; therefore, when testing it on $10\times$ sparser point cloud, our depth representation will be more sparse as well, which limits the potential of our calibrated backprojection layers. In contrast, [21] proposed a network to first estimate the dense coarse topology. This phenomenon is also observed in KITTI, shown in Table 3 of the main text, where we removed our sparse-to-dense module and we observed a significant drop in performance.

Fig. 6 shows qualitative evaluations on the three density levels. For 0.50%, error is low overall and the shape of the recovered scene resembles that of the ground truth. When we decrease density to 0.15%, we observe slight blurring in object shapes and increased errors in homogeneous regions. At 0.05%, we begin to observe artifact such as the green "blob" corresponding to the wall with more exaggerated errors in homogeneous regions. This is because locally the textureless surfaces give little to no information on object shape. Without sparse depth to anchor their values, they can be arbitrary. In this case the "empty" region is predicted as far.

4. Generalization to Other Sensor Platforms

In Sec. 3.4 of the main text, we discussed our ability to generalize to other sensor platforms that may use a different test time camera than one used to collect training data. In Table 5 of the main text, we showed quantitatively that we generalize better than the baseline. Here, we demonstrate this qualitatively in Fig. 7.

To this end, we trained our model on VOID [23] (captured by Intel RealSense) and tested the model on NYUv2 [18] (captured by Microsoft Kinect). We similarly trained the baseline [23] on VOID and tested it on NYUv2. Fig. 7 shows the predicted depth, backprojected to the point clouds in 3D and colored. As we can see, [23] predicted a distorted scene; in contrast, ours is not perfect, but reasonable. This demonstrates the benefit of taking calibration as input. It allows the model to generalize well when it is deployed to a sensor platform where the camera that is used is *different* than the one used for training. We also note that neither models have been trained on NYUv2 which features a different scene distribution than that of VOID.

5. KITTI Depth Completion Benchmark

In Sec. 3.3 of the main text, we compare our method against *unsupervised* methods on the KITTI online leaderboard. Here, we show quantitative comparisons against both supervised (Table 4) and unsupervised (Table 5) methods. Results and method names are directly taken from the KITTI online leaderboard. Here we refer to our method as KNBet, as listed on the leaderboard. We note that SS-S2D [13] and DDP [26] compete in both supervised and unsupervised benchmarks. Additionally, we provide high resolution examples of our output in Fig. 8.

Despite being trained without ground-truth annotations, Table 4 shows that our method is competitive even amongst supervised method. We outperform some supervised methods [4, 5] across most metrics. Our iMAE score, which penalizes mean error in close range regions, is ranked 5th overall amongst both supervised and unsupervised methods. We note that supervised methods are generally more computationally expensive with high model complexity e.g. in terms of number of parameters, [14] uses 25.84M, [15] 53.4M, and [25] 28.99M; whereas we only use 6.9M.

Compared to unsupervised methods (Table 5), we rank first amongst all methods with the best scores across all metrics. Our model even beat methods [12, 26, 21] that use additional synthetic data (Virtual KITTI [7]) for training, amongst which is the state of the art [21]. Despite this, we beat [21] by an average of 8% across all metrics while using 11.5% fewer parameters. These results demonstrates the potential of our method to bridge the gap between super-

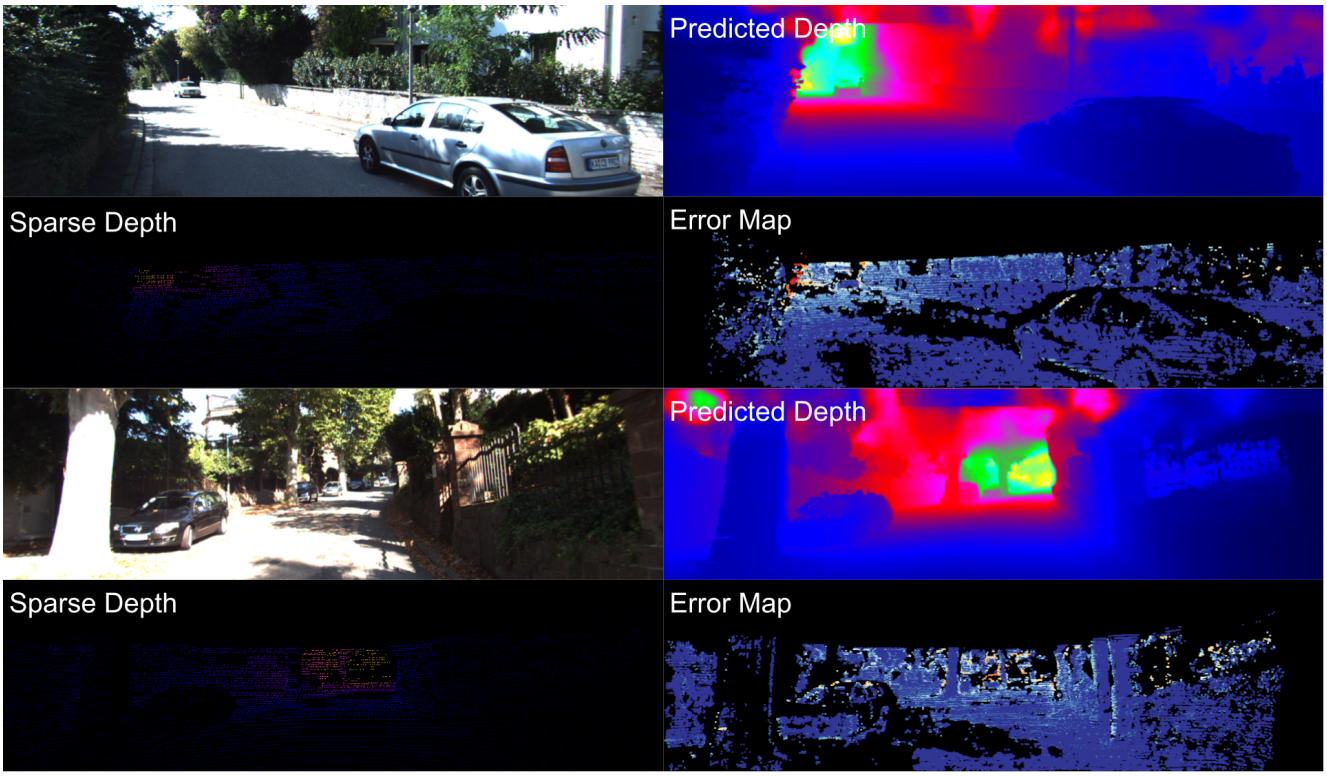

Figure 8: *Qualitative results on KITTI depth completion benchmark.*

Method	MAE	RMSE	iMAE	iRMSE
ADNN [4]	439.48	1325.37	3.19	59.39
Morph-Net [5]	310.49	1045.45	1.57	3.84
KBNet (Ours)	*258.36*	*1068.07*	*1.03*	*3.01*
SS-S2D [13]	249.95	814.73	1.21	2.80
DeepLiDAR [15]	226.50	758.38	1.15	2.56
PwP [25]	235.73	785.57	1.07	2.52
UberATG-FuseNet [2]	221.19	752.88	1.14	2.34
RGB_guide&certainty [20]	215.02	772.87	0.93	2.19
DDP [26]	203.96	832.94	0.85	2.10
CSPN++ [3]	209.28	743.69	0.90	2.07
NLSPN [14]	**199.59**	**741.68**	**0.84**	**1.99**

Table 4: *KITTI supervised depth completion benchmark.* Results are directly taken from online leaderboard. Note: SS-S2D [13] and DDP [26] compete in both supervised and unsupervised benchmarks. Our results are italicized. Despite being an unsupervised method, our method beats some supervised methods [4, 5] and our iMAE score (1.03) is ranked 5th amongst supervised methods.

Method	MAE	RMSE	iMAE	iRMSE
SGDU [16]	605.47	2312.57	2.05	7.38
SS-S2D [13]	350.32	1299.85	1.57	4.07
IP-Basic [11]	302.60	1288.46	1.29	3.78
DFuseNet [17]	429.93	1206.66	1.79	3.62
DDP* [26]	343.46	1263.19	1.32	3.58
VOICED [23]	299.41	1169.97	1.20	3.56
AdaFrame [22]	291.62	1125.67	1.16	3.32
SynthProj* [12]	280.42	1095.26	1.19	3.53
ScaffNet* [21]	280.76	1121.93	1.15	3.30
KBNet (Ours)	*258.36*	*1068.07*	*1.03*	*3.01*

Table 5: *KITTI unsupervised depth completion benchmark.* Results are directly taken from online leaderboard. Note: SS-S2D [13] and DDP [26] compete in both supervised and unsupervised benchmarks. Our method outperforms is trained only on KITTI, but still the state of the art [21] (trained on KITTI and Virtual KITTI [7]) by an average of 8% across all metrics. * denotes methods that use additional synthetic data for training.

vised and unsupervised method. Moreover, our network is light-weight and can be deployed on VIO system [6]. While there is still a long road ahead, these results show a lot of promise in enabling unsupervised methods to learn online and to be used for real-time application for low-cost hardware systems.

References

[1] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 3

[2] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10023–10032, 2019. 7

[3] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10615–10622, 2020. 7

[4] Nathaniel Chodosh, Chaoyang Wang, and Simon Lucey. Deep convolutional compressed sensing for lidar depth completion. In *Asian Conference on Computer Vision*, pages 499–513. Springer, 2018. 6, 7

[5] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Learning morphological operators for depth completion. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 450–461. Springer, 2018. 6, 7

[6] Xiaohan Fei, Alex Wong, and Stefano Soatto. Geo-supervised visual depth prediction. *IEEE Robotics and Automation Letters*, 4(2):1661–1668, 2019. 2, 7

[7] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 6, 7

[8] 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

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 3

[10] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, pages 1–15, 2015. 2

[11] Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22. IEEE, 2018. 7

[12] Adrian Lopez-Rodriguez, Benjamin Busam, and Krystian Mikolajczyk. Project to adapt: Domain adaptation for depth completion from noisy and sparse sensor data. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 6, 7

[13] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019. 3, 6, 7

[14] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In-So Kweon. Non-local spatial propagation network for depth completion. In *European Conference on Computer Vision, ECCV 2020*. European Conference on Computer Vision, 2020. 6, 7

[15] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019. 6, 7

[16] Nick Schneider, Lukas Schneider, Peter Pinggera, Uwe Franke, Marc Pollefeys, and Christoph Stiller. Semantically guided depth upsampling. In *German conference on pattern recognition*, pages 37–48. Springer, 2016. 7

[17] Shreyas S Shivakumar, Ty Nguyen, Ian D Miller, Steven W Chen, Vijay Kumar, and Camillo J Taylor. Dfusenet: Deep fusion of rgb and sparse depth information for image guided dense depth completion. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 13–20. IEEE, 2019. 7

[18] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 1, 2, 6

[19] 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. 1, 2

[20] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy lidar completion with rgb guidance and uncertainty. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6. IEEE, 2019. 7

[21] Alex Wong, Safa Cicek, and Stefano Soatto. Learning topology from synthetic data for unsupervised depth completion. *IEEE Robotics and Automation Letters*, 6(2):1495–1502, 2021. 3, 5, 6, 7

[22] Alex Wong, Xiaohan Fei, Byung-Woo Hong, and Stefano Soatto. An adaptive framework for learning unsupervised depth completion. *IEEE Robotics and Automation Letters*, 6(2):3120–3127, 2021. 2, 7

[23] Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from visual inertial odometry. *IEEE Robotics and Automation Letters*, 2020. 1, 2, 3, 5, 6, 7

[24] Alex Wong and Stefano Soatto. Bilateral cyclic constraint and adaptive regularization for unsupervised monocular depth prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5644–5653, 2019. 2

[25] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2811–2820, 2019. 6, 7

[26] Yanchao Yang, Alex Wong, and Stefano Soatto. Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3353–3362, 2019. 3, 6, 7

[27] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000. 5