Test-Time Adaptation for Depth Completion

Supplementary Material

Summary of contents

- In Section A, we present the GPU time of each adaptation method to show the effectiveness of our method.
- In Section B, we present the preliminary observations with image and range inputs of varying sparsity.
- In Section C, we describe the datasets used.
- In Section D, we present the hyperparameter settings for result reproduction and we elucidate evaluation details.
- In Section E, we provide a study on the learned proxy embedding with a visualization.
- In Section F, we present an ablation study of the loss components in our method.
- In Section G, we present the results on KITTI $\rightarrow$ VKITTI adaptation.
- In Section H, we present the results on a different source dataset (Waymo $\rightarrow$ VKITTI).
- In Section I, we show a qualitavive result of the preliminary observation.

A. Adaptation speed

We compare the GPU time of our adaptation method with the baselines (BN Adapt, CoTTA) on VKITTI in Table 1.

Compared to CoTTA, our adaptation method does not require multiple inferences to get the pseudo-prediction (derived from averaging teacher model predictions with different RGB augmentations) used to adapt the student model. Yet, our method requires an additional computation for the proxy embedding. Thus, the proxy layer's size relative to the model size causes the adaptation time difference. For example, CoTTA reduced the total time by 38.9% over ProxyTTA-fast on MSGCHN, which is a light-weight depth completion model. In this case, the proxy layer is relatively larger than in other models, where multiple inferences require less computation than the proxy layer. As a result, the total time is increased in MSGCHN. However, for large models (NLSPN, CostDCNet), ProxyTTA reduced total time by 56.6% over CoTTA; our proxy layer size is relatively smaller than the large models, while still improving performance by 26.52%. Compared to BN Adapt, our method requires additional parameters for the adaptation layer and the proxy layer. Hence, our method is 38.18% slower in adaptation time, 19.36% slower in evaluation time, and 33.16% in total. Yet, our method improves errors by 15.67% over BN Adapt.

Model	Method	Adaptation time	Evaluation time	Total time
MSGCHN	CoTTA	88.9 (-38.9%)	8.66 (-1.0%)	81.2 (-41.3%)
	ProxyTTA-fast	136.6	8.8	145.4
NLSPN	CoTTA	717.5 (+67.4%)	75.3 (-10.9%)	792.8 (+60.0%)
	BN Adapt	185.0 (-20.8%)	82.8 (-0.8%)	267.8 (-15.6%)
	ProxyTTA-fast	168.2 (-28.0%)	83.4 (-0.1%)	251.6 (-20.66%)
	ProxyTTA	233.6	83.5	317.1
CostDCNet	CoTTA	329.1 (+78.2%)	33.6 (-51.0%)	369.1 (+43.2%)
	BN Adapt	82.1 (-55.5%)	42.5 (-37.9%)	125.6 (-50.8%)
	ProxyTTA-fast	141.9 (-23.2%)	68.7 (+0.3%)	210.6 (-16.8%)
	ProxyTTA	184.7	68.5	253.2

Table 1. *GPU time for various methods and models, tested on Virtual KITTI.* Time is in milliseconds (ms). 'Adaptation time' denotes the time required to adapt (or train) each method for a single test data point. 'Evaluation time' denotes the time taken to test each method for a test data instance.'Total time' is the sum of the Adaptation and Evaluation times.

B. Further observations on image/range inputs

We present additional preliminary observations of the image and range sensor inputs with varying sparsity. Since previous works [2, 10] state that the depth completion model propagates the sparse depth to the dense depth guided by image features, one can raise a question on our preliminary results in the main paper without the lidar input, such as there's no sparse point to propagate to the near pixels. We clarify that the results are intended to highlight the domain distrepancy. Therefore, we show additional results with 1%, 5%, and 10% of sparse points in the range input on indoor datasets, as shown in Table 2. As we increase the range points, the performance is improved yet still worse than the sparse-depth-only results in Tab. 8.

C. Datasets

KITTI [6] is composed of calibrated RGB images with synchronized point clouds from Velodyne lidar, inertial, and GPS information, and from more than 61 driving scenes. There are ≈80K raw image frames and associated sparse depth maps, both with ≈5% density, available for depth completion [13]. Semi-dense depth is available for the lower 30% of the image space, and 11 neighboring raw lidar scans comprise the ground-truth depth. We did not use a test or validation set, and the training set contains ≈86K single images.

VOID [14] contains synchronized 640×480 RGB images and sparse depth maps from indoor scenes of laboratories and classrooms and from outdoor scenes of gardens. Sparse depth maps (of ≈0.5% density and containing ≈1,500 sparse dense points) are obtained by the VIO system XIVO [4], and dense ground-truth depth maps are obtained by active stereo. VOID uses rolling shutter to capture challenging 6 DoF motion for 56 sequences - as opposed to KITTI's typically

Method	MSG-CHN		NLSPN		CostDCNet		MSG-CHN		NLSPN		CostDCNet	
Dataset	VOID→NYUv2						VOID→ScanNet					
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Image + sparse depth (1%)	1643.34	2177.71	602.17	858.19	809.36	1144.91	1597.41	2240.43	490.13	738.77	665.57	982.32
Image + sparse depth (5%)	996.54	1599.14	379.45	638.55	427.69	736.23	809.38	1455.69	240.55	441.70	337.39	620.53
Image + sparse depth (10%)	785.65	1376.93	327.41	591.99	339.31	622.75	581.93	1165.63	191.75	379.10	264.66	516.74
Sparse depth only	**734.13**	**1046.28**	**237.47**	**402.47**	**147.76**	**354.57**	**211.86**	**444.62**	**162.29**	**276.29**	**88.25**	**205.46**

Table 2. *Model sensitivity to input modalities with varying sparsity.*

planar motion. We use a training set of ≈46K images to prepare the model.

NYUv2 [11] contains 372K synchronized 640×480 RGB images and depth maps (via Microsoft Kinect) from 464 indoor scenes of household, office, and commercial types. To generate sparse depth maps in the style of SLAM/VIO, we used the Harris corner detector [8] to sample ≈1,500 points from the depth maps. We use a set of 654 test set images for adaptation.

ScanNet [3] contains 2.5 million images and dense depth maps for 1,513 indoor scenes. To generate sparse depth maps in the style of SLAM/VIO, we used the Harris corner detector [8] to sample ≈1,500 points from the depth maps. We use a set of ≈21K test images for adaptation.

Virtual KITTI (VKITTI) [5] contains ≈17K 1242×375 images from 35 synthetic videos created by applying 7 variations in weather, lighting, or camera angle to each of 5 cloned KITTI [13] videos. There exists a large domain gap between RGB images from VKITTI and KITTI, even though the virtual worlds created in Unity by [5] are similar to KITTI scenes. Thus, we only use the dense depth maps of VKITTI to avoid the domain gap in photometric varations. The sparse depth maps are obtained by simulating KITTI's lidar-generated sparse depth measurements such that the marginal distribution of VKITTI's sparse points mimics that of KITTI's. We use a set of ≈2,300 test images for the adaptation.

nuScenes [1] consists of 1600×900 calibrated RGB images and synchronized sparse point clouds, 27.4K images from 1000 outdoor driving scenes for training, and 5.8K images from 150 scenes for testing. We set up the ground truth for the test images by merging projected sparse depth from forward-backward frames. The setup code will be released to clarify further details and reproducibility.

SceneNet [9] contains 5 million 320×240 RGB images and depth maps from indoor trajectories of randomly arranged rooms. We use a single split (out of 17 available) containing 1000 subsequences of 300 images each, generated by recording the same scene over a trajectory. Because there are no sparse depth maps provided, we sampled from the depth map via Harris corner detector [8] to mimic the

Dataset	Learning Rate	w_{sm}	w_z	w_{proxy}	Inner Iter.
MSG-CHN					
VKITTI	2e-3	1.0	1.0	0.2	1
VKITTI-FOG	5e-3	3.0	1.0	0.1	1
nuScenes	3e-3	9.0	1.0	0.2	1
SceneNet	2e-3	8.0	1.0	0.1	3
NYUv2	2e-4	0.8	1.0	0.4	3
ScanNet	5e-3	8.0	1.0	0.3	3
NLSPN					
VKITTI	2e-3	0.8	1.0	0.4	1
VKITTI-FOG	1e-3	1.0	1.0	0.2	1
nuScenes	1e-3	1.0	1.0	0.1	1
SceneNet	2e-3	0.7	1.0	2.0	3
NYUv2	4e-3	5.0	1.0	1.0	3
ScanNet	1e-4	2.0	1.0	0.3	3
CostDCNet					
VKITTI	4e-3	4.5	1.0	0.1	1
VKITTI-FOG	5e-3	3.0	1.0	0.04	1
nuScenes	5e-3	3.0	1.0	0.1	1
SceneNet	7e-3	2.0	1.0	0.2	3
NYUv2	6e-3	4.0	1.0	0.1	3
ScanNet	3e-3	1.0	1.0	0.2	3

Table 3. *Hyperparameters.* For MSG-CHN, NLSPN, and CostDCNet methods for initialization, preparation, and adaptation.

sparse depth produced by SLAM/VIO. The final 375 corners are obtained by using k-means to subsample the resulting points, representing 0.49% of the total pixels. We use a set of ≈2,300 test images for adaptation.

Waymo Open Dataset [12] contains 1920×1280 RGB images and lidar scans from autonomous vehicles. The training set contains ≈158K images from 798 scenes and the validation set ≈40K images from 202 scenes, collected at 10Hz. Objects are annotated across the full 360° field. We obtain our validation set by sampling from the whole validation dataset every 0.6 seconds. Range sensor inputs are obtained by projecting the top lidar's point cloud scan to the camera frame. We obtained the ground truth by projecting 10 forward and backward frames from front lidar and top lidar to the image frame, which approximately counts for

1 second of capture. To assume that the reprojected scenes are static, we removed the moving objects in the scenes using object annotations. Also, outlier removal is utilized for filtering out errorenous depth points.

D. Implementation details

Hyperparameter. We specifically note the hyperparameters of three methods for initialization, preparation, and adaptation on Table 3.

Epochs and training details Adaptation occurs in a single epoch, with 'the number of iterations per data point' (*inner-iter*) specified in Tab. 3. During initialization and preparation stages, the adaptation and proxy layers are trained for 6 epochs. Batch sizes for all methods are: 48 for preparation stage, 16 for initialization and adaptation stages, with the exception of ScanNet [6], using a batch size of 36. To prevent collapse during preparation stage, we follow the protocol of [7]; we exploit the projection / prediction layers and divide online / target branch, and update target projection layer with exponential moving average of online branch. We used embedding dimension and hidden dimension of 512 for MSGCHN, and 1024 for CostDCNet and NLSPN. The learning rates for initialization and preparation stage will be released with the code release.

Evaluation. We evaluate our adaptation models on bottom-cropped regions in the outdoor dataset, where the sparse depth exists. For outdoor dataset, models are evaluated on the bottom cropped region of the test split, 1242×240 for Virtual KITTI, and 1600×544 for nuScenes. For indoor dataset, we evaluated the models on the entire region. The definition of the error metrics in evaluation are described in Table 5. We evaluate our model on depth range from 0.0 to 80.0 meters for the ourdoor, and 0.2 to 5.0 meters for the indooor.

E. Discussion on learned proxy embeddings

Here, we provide the t-SNE visualization of image & sparse depth and proxy embedding from source and target.

Fig. 1 shows the embeddings visualized by t-SNE, where the target domain proxy embeddings' centroid is closer to that of source's proxy and image & sparse depth embeddings, than to the centroid of target's image & sparse depth embeddings, highlighting effectiveness of proxy embedding for adaptation.

F. Ablation study

Here, we ablate the effect of each loss term denoted with the checkmarks in Table 4. Using *sparse depth consistency loss* ℓ_z (Eqn. 4) alone can improve the pretrained model as it learns the shapes of the test domain. However, because of the sparsity, the supervision signal is weak, leading the model to exhibit artifacts and distortions in the depth map.

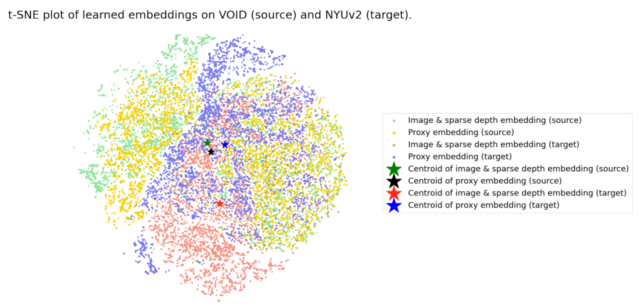

Figure 1. *t-SNE plot of learned embeddings on VOID and NYUv2.*

Including a *local smoothness loss* ℓ_{sm} (Eqn. 5) mitigates this by propagating depth to nearby regions. However, without knowledge of 3D shapes compatible with the sparse points, the wrong predictions are sometimes propagated as in the left bounding box region from Row 1, Column 4 of Fig. 4. The best-performing method employs the proposed proxy embeddings as a regularizer to guide the adaptation layer update. As the proxy mapping produces test-time features that follow the distribution of the source domain, minimizing our *proxy consistency* loss (Eqn. 6) implicitly aligns the test domain features to those of the source domain that are compatible with the 3D scene observed by the test-time sparse point cloud. Not only does this improve overall performance, but it also reduces standard deviation in error, which can be interpreted as an increase in the stability of the adaptation. We show qualitative comparisons against BN Adapt in Fig. 4, where boxes highlight improvements by fixing erroneous propagation by local smoothness (e.g., bleeding effect, which is not mitigated by using image gradients as guidance in Eqn. 5). Quantitatively, we improve over the baseline by an average of 21.09% across all methods and datasets, demonstrating the efficacy of our proxy embedding.

G. KITTI $\rightarrow$ VKITTI results

Here, we present additional results on KITTI $\rightarrow$ VKITTI adaptation. Test-time adaptation results are shown in Table 6. Consistent with the trends observed in the main paper, our method outperforms over both BN Adapt and CoTTA, with a 21.82% improvement compared to BN Adapt and 12.6% improvement over CoTTA.

H. Experiment with different source dataset

In our main paper, the only source dataset for outdoor adaptation scenario was KITTI which is the most popular outdoor depth completion dataset. To validate our method's applicability to models trained on diverse source datasets, we include additional results from adaptation scenarios using a model trained on the Waymo dataset, as shown in Table 7. Our method shows an improvement over CoTTA and BN Adapt by 21.70%.

				KITTI → Waymo		KITTI → VKITTI-FOG		KITTI → nuScenes	
Method	ℓ_z	ℓ_{sm}	ℓ_{proxy}	MAE	RMSE	MAE	RMSE	MAE	RMSE
MSG-CHN	✓			951.25±3.14	3512.07±6.40	978.84±3.36	3561.40±15.48	3164.46±11.32	6453.54±17.31
	✓	✓		*613.01±1.99*	*1935.43±9.14*	*732.61±6.02*	*3113.11±21.78*	*2865.15±9.96*	*6144.48±24.14*
	✓	✓	✓	**608.91±1.74**	**1921.83±2.54**	**728.24±3.73**	**3087.36±15.92**	**2834.08±17.64**	**6096.56±21.08**
NLSPN	✓			837.66± 8.73	3668.94± 25.90	715.86±26.36	*3034.21± 57.65*	5076.83±53.85	9710.88± 89.76
	✓	✓		*489.46±5.45*	*1613.66±30.04*	*705.14±16.86*	3059.64±97.85	*2783.61±159.62*	*6313.4±276.09*
	✓	✓	✓	**477.28±3.32**	**1598.64±18.95**	**686.91±22.14**	**2666.70±56.64**	**2589.25±59.03**	**6006.18±90.66**
CostDCNet	✓			816.33±32.01	3431.96±55.34	807.62±69.12	3254.83±179.90	3135.11±81.76	7596.49±159.16
	✓	✓		*469.52±2.54*	*1594.38±6.10*	*516.93±1.62*	*2751.21±17.42*	*2067.42±10.23*	**5487.85±37.21**
	✓	✓	✓	**466.44±1.63**	**1580.38±11.48**	**512.72±0.74**	**2735.01±3.53**	**2062.28±11.24**	*5509.96±23.41*
				VOID → NYUv2		VOID → SceneNet		VOID → ScanNet	
MSG-CHN	✓			*971.64±66.86*	*1291.45±45.67*	242.11±4.24	491.48±10.49	462.95±34.84	659.9±37.93
	✓	✓		1005.49±25.97	1329.76±25.01	*194.60±3.64*	*425.16±10.58*	*330.20±48.46*	*503.73±57.14*
	✓	✓	✓	**699.60±6.00**	**1120.37±9.76**	**192.74±1.72**	**424.49±4.58**	**302.21±4.10**	**480.08±8.03**
NLSPN	✓			145.72 ±6.55	271.78± 9.91	130.49±13.64	*337.14±28.38*	112.38±1.72	234.60±3.46
	✓	✓		*128.17±4.13*	*240.97±3.86*	*118.65±2.24*	337.63±2.58	*77.84±0.28*	*169.81±0.50*
	✓	✓	✓	**124.41±2.27**	**240.73±5.72**	**113.93±1.49**	**333.41±4.32**	**74.77±0.31**	**166.61±0.45**
CostDCNet	✓			152.43±13.07	432.20±54.51	213.4±19.52	597.22±49.78	91.13±1.40	286.17±9.07
	✓	✓		*101.31±1.67*	*217.77±6.00*	*134.51±4.23*	*360.33±9.67*	*69.02±0.51*	*164.90±2.38*
	✓	✓	✓	**95.87±2.16**	**203.83±4.72**	**125.75±1.93**	**357.12±4.13**	**68.17±0.44**	**162.35±1.12**

Table 4. *Ablation study of each loss term.* Note that NLSPN and CostDCNet update the adaptation layer and batch normalization layers, yet MSGCHN only updates the adaptation layer.

Metric	Definition
MAE	$\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert\hat{d}(x)-d_{gt}(x)\rvert$
RMSE	$\left(\frac{1}{\lvert\Omega\rvert}\sum_{x\in\Omega}\lvert\hat{d}(x)-d_{gt}(x)\rvert^2\right)^{1/2}$

Table 5. *Error metrics.* d_{gt} denotes the ground-truth depth.

		KITTI → VKITTI	
Method		MAE	RMSE
MSG-CHN	Pretrained	2433.46	6675.16
	CoTTA	*839.19±12.78*	*3625.38±39.35*
	ProxyTTA-fast (Ours)	**800.88±1.86**	**3268.26±4.12**
NLSPN	Pretrained	1469.19	8060.97
	BN Adapt	1016.87±8.84	*3453.00±3.21*
	BN Adapt, ℓ_z, ℓ_{sm}	855.12±14.56	3516.85±58.63
	CoTTA	*775.09±3.63*	3585.37±13.31
	ProxyTTA-fast	849.43±3.61	3540.44±3.57
	ProxyTTA (Ours)	**639.19±5.68**	**2934.36±33.80**
CostDCNet	Pretrained	845.35	3774.01
	BN Adapt	1248.35±0.25	4267.64±0.62
	BN Adapt, ℓ_z, ℓ_{sm}	1016.87±8.84	3453.00±3.21
	CoTTA	*698.42±9.93*	*3324.59±30.21*
	ProxyTTA-fast	822.49±13.55	3331.24±55.30
	ProxyTTA (Ours)	**639.91±8.92**	**2951.21±30.93**

Table 6. *Additional results for test-time adaptation for depth completion on KITTI → VKITTI.*

A noteworthy observation from the Waymo adaptation results, when compared to the KITTI → VKITTI-fog results from the main paper, is that the adaptation result of KITTI outperforms that of Waymo. This difference is caused by from the domain discrepancies between KITTI and VKITTI-fog datasets versus the domain gap between Waymo and VKITTI-fog. For example, VKITTI's object appearances and resolution (1226×370 for KITTI, and 1242×375 for VKITTI) are more akin to those in the KITTI dataset. Conversely, the Waymo dataset features higher resolution (1920×1280) and different object shapes compared to KITTI and VKITTI. Hence, the adaptation result is influenced by the extent of domain discrepancy between the source and target datasets.

		Waymo → VKITTI-FOG	
Method		MAE	RMSE
MSG-CHN	Pretrained	1473.14	4676.19
	CoTTA	*1348.02±38.03*	*4016.67±28.16*
	ProxyTTA-fast (Ours)	**1052.78±5.74**	**3891.05±17.34**
NLSPN	Pretrained	2734.27	*37621.10*
	BN Adapt, ℓ_z, ℓ_{sm}	*1205.96±40.14*	3857.88±101.15
	CoTTA	2485.66±18.05	6307.96±48.64
	ProxyTTA (Ours)	**808.16±7.86**	**3536.58±91.15**
CostDCNet	Pretrained	1261.00	4360.37
	BN Adapt, ℓ_z, ℓ_{sm}	*742.99±2.17*	*3403.00±3.62*
	CoTTA	1150.16±5.69	4134.16±9.15
	ProxyTTA (Ours)	**724.77±5.18**	**3349.21±29.00**

Table 7. *Additional results for test-time adaptation for depth completion on Waymo → VKITTI-FOG.*

Method	MSG-CHN		NLSPN		CostDCNet		MSG-CHN		NLSPN		CostDCNet	
Dataset	VOID→NYUv2						VOID→ScanNet					
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Image only	2072.78	2462.63	969.14	1228.44	1359.16	1619.40	2001.90	2451.681	899.41	1151.12	1216.17	1459.46
Sparse depth only	**734.13**	**1046.28**	**237.47**	**402.47**	**147.76**	**354.57**	**211.86**	**444.62**	**162.29**	**276.29**	**88.25**	**205.46**
Image + sparse depth	1040.93	1528.98	387.36	704.66	189.10	446.71	316.646	698.633	232.332	431.199	144.311	458.692
Dataset	KITTI→Waymo						KITTI→nuScenes					
Image only	12766.791	18324.83	18829.96	24495.73	13598.50	18376.15	11823.061	17244.44	15835.04	22613.78	12794.65	16744.15
Sparse depth only	**861.13**	**2706.75**	1290.28	3571.26	1210.93	3102.49	3943.97	7306.33	**2540.58**	6203.66	**2996.28**	6773.06
Image + sparse depth	1103.33	2969.39	**1173.26**	**3092.02**	**1084.18**	**2819.42**	**3331.82**	**6449.09**	2656.61	**6146.59**	3064.72	**6630.65**

Table 8. *Model sensitivity to input modalities.* Depth completion networks have a high reliance on sparse depth modality. Performing inference in a novel domain without the RGB image, i.e., using just sparse depth as input, can improve over using both data modalities.

I. Quantitative preliminary results

To provide a precise observation, we provide the quantitative results of model sensitive study in Tab. 8.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 2

[2] 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*, pages 10615–10622, 2020. 1

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2

[4] Xiaohan Fei, Alex Wong, and Stefano Soatto. Geo-supervised visual depth prediction. *IEEE Robotics and Automation Letters*, 4(2):1661–1668, 2019. 1

[5] 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. 2

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32:1231 – 1237, 2013. 1

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 3

[8] Christopher G. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 2

[9] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016. 2

[10] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 120–136. Springer, 2020. 1

[11] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012. 2

[12] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 2

[13] 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

[14] Alex Wong, Xiaohan Fei, Stephanie Tsuei, and Stefano Soatto. Unsupervised depth completion from visual inertial odometry. *IEEE Robotics and Automation Letters*, 5(2):1899–1906, 2020. 1