Test-Time Adaptation for Depth Completion

Hyoungseob Park
Yale Vision Lab
hyoungseob.park@yale.edu

Anjali Gupta
Yale Vision Lab
anjali.gupta@yale.edu

Alex Wong
Yale Vision Lab
alex.wong@yale.edu

Abstract

It is common to observe performance degradation when transferring models trained on some (source) datasets to target testing data due to a domain gap between them. Existing methods for bridging this gap, such as domain adaptation (DA), may require the source data on which the model was trained (often not available), while others, i.e., source-free DA, require many passes through the testing data. We propose an online test-time adaptation method for depth completion, the task of inferring a dense depth map from a single image and associated sparse depth map, that closes the performance gap in a single pass. We first present a study on how the domain shift in each data modality affects model performance. Based on our observations that the sparse depth modality exhibits a much smaller covariate shift than the image, we design an embedding module trained in the source domain that preserves a mapping from features encoding only sparse depth to those encoding image and sparse depth. During test time, sparse depth features are projected using this map as a proxy for source domain features and are used as guidance to train a set of auxiliary parameters (i.e., adaptation layer) to align image and sparse depth features from the target test domain to that of the source domain. We evaluate our method on indoor and outdoor scenarios and show that it improves over baselines by an average of 21.1%. Code available at github.com/seobbro/TTA-depth-completion.

1. Introduction

Reconstructing the 3-dimensional (3D) structure of an environment can support a number of spatial tasks, from robotic navigation and manipulation to augmented and virtual reality. Most systems addressing these tasks are built for sensor platforms equipped with range (i.e., lidar or radar) or optics (i.e., camera or sensors). While range sensors can measure the 3D coordinates of the surrounding space, they often yield point clouds that are sparse. Likewise, these coordinates can also be estimated from images by means of Structure-from-Motion (SfM) or Visual Inertial Odometry (VIO). For the goal of dense mapping, depth completion is the task of recovering the dense depth of a 3D scene as observed from a sparse point cloud, which is often post-processed into a sparse depth map by projecting the points onto the image plane, and guided by a synchronized calibrated image.

Training a depth completion model can be done in a supervised (using ground truth) or unsupervised (using SfM) manner. The former dominates in performance, but requires expensive annotations that are often unavailable; the latter uses unannotated images, but they must satisfy SfM assumptions between frames, i.e., motion, covisibility, etc. Like most learning-based methods, models trained under both paradigms typically experience a performance drop when tested on a new dataset due to a covariate shift, i.e., domain gap. As we can only assume that a single pair of image and sparse depth map is available in the target domain for the depth completion, models belonging to either learning paradigms cannot easily be trained or adapted to the new domain even when given the testing data. We focus on test-time adaptation (TTA) for depth completion, where one is given access to the test data in a stream, i.e., one batch at a time, without being able to revisit previously-seen examples. The goal is to learn causally and to quickly adapt a set of pre-existing weights trained on a source domain to a target test domain, so one can reduce the performance gap.

We begin with some motivating observations on the effects of the domain gap: (i) Errors in target domain tend to be higher when feed both the image and sparse depth as input rather than *sparse depth only*, as shown in Fig. 1. This implies that the depth modality exhibits a smaller covariate shift between the source and target domains than the image modality, to the extent that forgoing the image altogether often yields superior results than using either both sparse depth and image or the image alone. (ii) Yet, when operating in the source domain, we observe the opposite effect – forgoing the image is detrimental to performance. Naturally, this begs the question: How should one leverage data modalities that are less sensitive to the domain shift (e.g., sparse depth) to support alignment between source and target domains for modalities that are more sensitive (e.g., RGB image)?

To answer this question, we investigate a test-time adaptation approach that learns an embedding for guiding the

model parameter update by exploiting the data modality (sparse depth) that is less sensitive to the domain shift. The embedding module maps the latent features encoding sparse depth to the latent features encoding both image and sparse depth. The mapping is trained in the source domain and frozen when deployed to the target domain for adaptation. During test time, sparse depth is first fed through the encoder and mapped, through the embedding module, to yield a proxy for image and sparse depth embeddings from the source domain – we refer to the embedded sparse depth features as *proxy embeddings*. Note: As the mapping is learned in the source domain, the proxy embeddings will also follow the distribution of source image and sparse depth embeddings. Next, both image and sparse depth from the target test domain are fed as input to the encoder. By maximizing the similarity between test-time input embeddings and the proxy embeddings, we align the target distribution to that of the source to reduce the domain gap. In other words, our method exploits a proxy modality for guiding test-time adaptation and we call the approach, *ProxyTTA*. When used in conjunction with typical loss functions to penalize discrepancies between predictions and input sparse depth, and abrupt depth transitions, i.e., Total Variation, the embeddings serve as regularization to guide the model parameter update and prevent excessive drift from those trained on the source data.

Following test-time adaptation conventions, we assume limited computational resources, and that inputs arrive in a stream of small batches and must be processed within a time budget without access to the past data. To ensure fast model updates under these constraints, we deploy auxiliary parameters, or an adaptation layer, to be updated while freezing the rest of the network – thus achieving low-cost adaptation. We demonstrate our method in both indoor (VIO) and outdoor (lidar) settings across six datasets, where we not only target typical adaptation scenarios where the shift exists between real and synthetic data domains with similar scenes, i.e. from KITTI [45] to Virtual KITTI [12], but also between different scene layouts, i.e., from VOID [51] to NYUv2 [30] and SceneNet [29]. Our proxy embeddings consistently improve over baselines by an average of 21.09% across all methods and datasets. To the best of our knowledge, we are the first to introduce test-time adaptation for depth completion.

2. Related work

Test Time Adaptation (TTA) aims to adapt a given model, pretrained on source data, to test data without access to the source training data. Related fields along this vein include unsupervised domain adaptation [13, 32], which utilizes source domain data (in practice, this may not be available) for adaptation, and source-free domain adaptation [22], which does not assume access to source data, but allows access to test data on multiple passes. In contrast, we focus on test-time adaptation where we do not have access to source data and must adapt to test data in a single pass.

Previous studies have proposed strategies to select the source model's component to be preserved, such as the class prototypes extracted from the source data [5, 26, 37], the subset of source model parameters [19, 48], and the discriminative feature from the self-supervised learning (SSL) [5]. For instance, [48] proposes TENT, a simple but effective batch-norm layer adaptation with entropy minimization for fully test-time adaptation. TTT [44] performs classification-layer adaptation by updating the last linear layer of the source model; [26] extends this with TTT++ and utilizes joint task-specific and model-specific information based on self-supervised learning. [19] presents T3A, an optimization-free classifier adjustment module. [5] uses shift-agnostic weight regularization (SWR) to prevent an effect from the erroneous signal in test time, jointly with the nearest source prototype classifier and a self-supervised proxy task. [2] proposes a contrastive learning with an online pseudo-label refinement while [37] proposes pseudo-label refinement and momentum update for 3D point cloud segmentation. [49] proposes continual test-time adaptation based on stochastic restoration and weight-averaged pseudo-labels. [41] uses efficient residual modules to realign the pretrained weights. The above methods largely focus on single-image-based tasks, i.e., classification [2, 5, 26] and semantic segmentation [37], and rely on entropy constraints from [48].

Unlike prior work on classification [26, 44] and segmentation [37], depth completion is a regression problem; hence, existing methods using entropy-based objectives [48], which operate on logits, are not applicable in this task. Instead, we propose to minimize sparse depth reconstruction and local smoothness objectives – similar to that of some existing unsupervised methods [50, 51] – and to maximize cosine similarity between the proxy embeddings and the test time image and sparse depth embeddings.

Depth Completion aims to output dense depth from a single image and synchronized point cloud, i.e., from lidar or tracked by VIO, projected onto a sparse depth map, by multimodal fusion [8, 51, 56–58].

Unsupervised depth completion approaches rely on Structure-from-Motion (SfM) for training and require access to an auxiliary training dataset containing stereo image pairs [38] or monocular videos [25, 28, 51–54] with synchronized sparse depth maps. Typically, they minimize a linear combination of photometric reprojection consistency, sparse depth reconstruction error, and local smoothness [28, 51–53]. These methods can support online training, but are limited by the need for stereo or monocular videos with sufficient parallax and co-visibility. In contrast, our approach does not rely on SfM and can be used in more general scenarios.

Supervised depth completion trains the model by minimizing a loss with respect to ground truth. [3, 18] focus on network operations and designs to effectively deal with

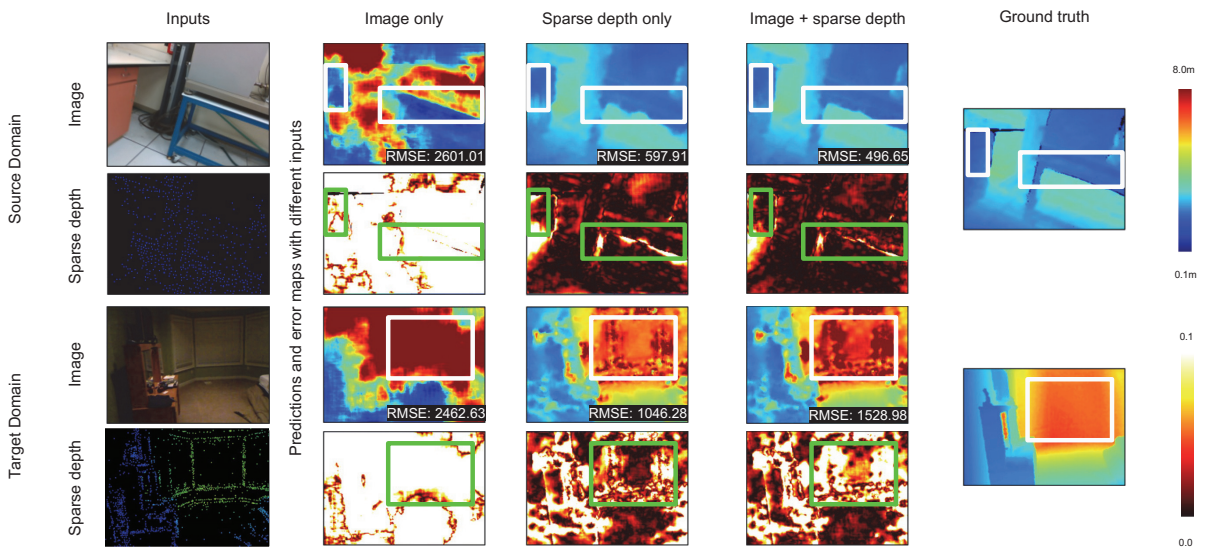


Figure 1. *Model sensitivity to input modalities.* While utilizing both sparse depth and image as input, the best performance is achieved in the source domain (VOID). Yet, forgoing the image in the test domain (NYUv2) often yields lower error than using both as input.

sparse inputs. [20, 28, 59] propose early and late fusion of image and depth encoder features while [17] uses separate networks for each. [23] proposes a multi-scale cascaded hourglass network to enhance the depth encoder with image features. [4] proposes convolutional spatial propagation network; [31] extends it to non-local spatial propagation to refine an initial depth map based on confidence and learnable affinity; [24] further extends it to dynamic spatial propagation. [9, 10, 34, 35] learn uncertainty of the depth estimates. [47] utilizes confidence maps to combine depth predictions while [33, 55, 61] use the surface normals to guide depth prediction. [21] incorporates cost volume for depth prediction. [40] used radar. [36] devises transformer architecture with cross-modal attention, and [60] proposes a hybrid convolution-transformer architecture for depth completion.

While both unsupervised and supervised methods have demonstrated strong performance on benchmarks, they often fail to generalize to test datasets with large domain discrepancies. Moreover, obtaining ground truth is unrealistic for real-time applications, and accumulating sufficient parallax incurs large latencies – presenting significant challenges for online adaptation. Unlike past works, we do not assume access to ground truth nor data outside of the input.

Unsupervised Domain Adaptation (UDA) addresses the discrepancy between labeled source data and unlabeled target data [6, 13, 32, 43]. The only existing UDA depth completion method [27] models the domain gap as the noise in sparse points and the appearance in images. Unlike most UDA approaches that require source data during adaptation, we are only given the inputs necessary for inference in a stream without the ability to revisit past data, and must update the model online under a limited computational budget.

Our Contributions. We present (i) a study on how the domain shift in each data modality (e.g., image and sparse depth) affects model performance when transferring it from source to target test domain. This study motivates (ii) our approach to learn an embedding of sparse depth features (which are less sensitive to the domain shift) that serves as proxy to source features for guiding test-time adaptation. (iii) To the best of our knowledge, we are the first to propose test-time adaptation for the depth completion task, and (iv) will release code, models, and dataset benchmarking setup to make development accessible for the research community.

3. Method Formulation

For ease of use, we assume access to a (source) pretrained depth completion model f_θ that infers a dense depth map $\hat{d}$ from a calibrated RGB image $I \in \mathbb{R}^{H\times W\times 3}$ and its associated sparse point cloud projected onto the image plane as a sparse depth map $z \in \mathbb{R}_+^{H\times W}$, i.e., $f_\theta(I, z) \rightarrow \hat{d} \in \mathbb{R}_+^{H\times W}$. For simplicity, we assume that the model was trained to minimize a supervised loss between prediction and ground truth $d \in \mathbb{R}_+^{H\times W}$ on a source dataset $\mathcal{D}_s = \{I_s^{(n)}, z_s^{(n)}, d^{(n)}\}_{n=1}^{N_s}$, where N_s indicates the number of data samples. Following conventions in TTA, we assume access to the source domain dataset prior to deployment.

During test-time adaptation, we follow the protocol of [26, 44], where we only have access to the target domain data $\mathcal{D}_t = \{I_t^{(n)}, z_t^{(n)}\}_{n=1}^{N_t}$ and utilize an online procedure to adapt to unseen N_t target data samples. Note that we make no assumptions about supervision during test-time; hence, while we present results on supervised methods for controlled experiments, we see our method being applicable towards unsupervised methods as well.

Our method, ProxyTTA, is split into three stages (Fig. 3):

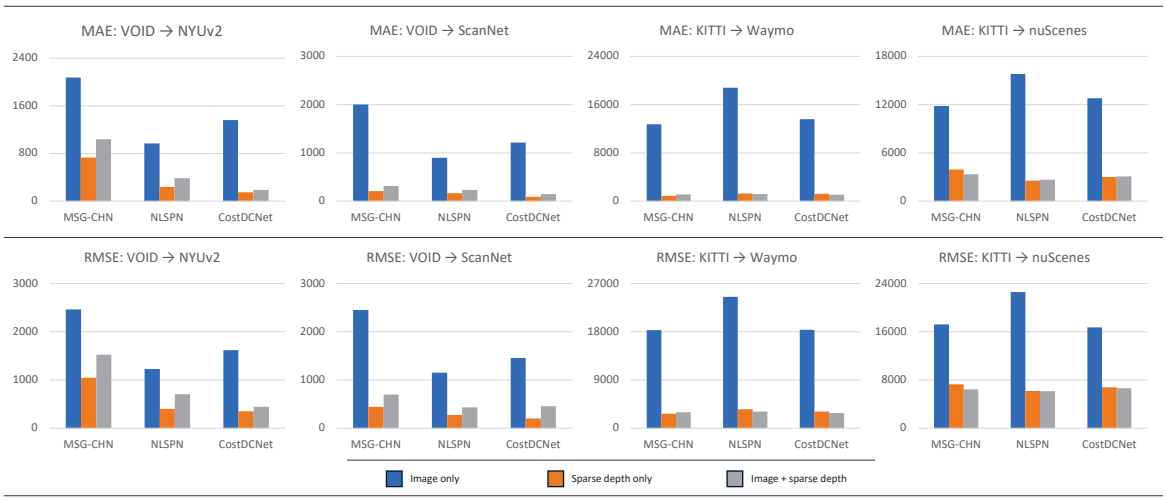


Figure 2. *Model sensitivity to input modalities.* Depth completion networks have a high reliance on sparse depth modality. Performing inference in a novel domain without the RGB image, i.e., using just sparse depth as input, can improve over using both data modalities.

(a) During an intialization stage, we augment the network encoder with an adaptation layer and train it using source domain data. (b) In the preparation stage, we learn a mapping from sparse depth features to image and sparse depth (proxy) embeddings. (c) During test time, we do not need the source dataset; we freeze the mapping and use its proxy embeddings for updating the adaptation layer parameters in test domain.

3.1. Sensitivity Study on Data Modalities

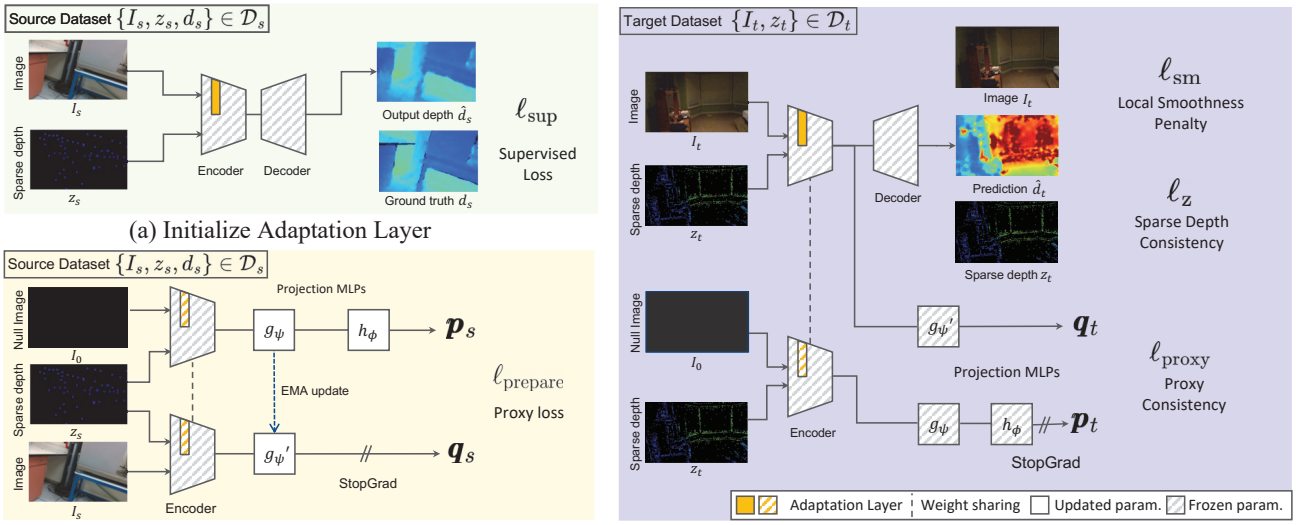
To motivate our approach, we begin with a sensitivity study of depth completion networks to input modalities, e.g. image, sparse depth, and the effect of domain shift on them. To this end, we alter the inputs by zeroing out either I or z to yield (I, z), (I_0, z), and (I, z_0), where I_0 and z_0 indicate the zero matrices with identical size to I and z, respectively. We evaluate the pretrained models using (I, z), (I_0, z), and (I, z_0) to highlight their dependence on each input modality and to gauge their sensitivity when one modality gives no useful information at all. Fig. 1 and Fig. 2 show qualitative (error maps) and quantitative results (bar graphs), respectively, of pretrained depth completion models when fed the different inputs on the source dataset $\mathcal{D}_s$ and the target dataset $\mathcal{D}_t$.

In the source domain, inference using both image and sparse depth as inputs, i.e., $\hat{d}_s(I, z)$, shows the best performance. Surprisingly, the inference using sparse depth alone (i.e., with null-image) $\hat{d}_s(I_0, z)$ is comparable to $\hat{d}_s(I, z)$. This shows the first intuition behind our approach: (i) Even though depth inputs are sparse, they are sufficient to support the reconstruction of the scene. Additionally, inference with image alone (i.e., null-depth) $\hat{d}_s(I, z_0)$ is worse than $\hat{d}_s(I_0, z)$ and $\hat{d}_s(I, z)$, which suggests that a depth completion network relies heavily on sparse depth modality for inference, and the image for guiding recovery of finer details.

In the target test domain, expectedly, performance degrades for inference using both image and sparse depth due to a covariate shift. Remarkably, we observe that predictions from sparse depth alone $\hat{d}_t(I_0, z)$ remain consistent in performance to those using both inputs $\hat{d}_t(I, z)$. Moreover, we observe that in most cases $\hat{d}_t(I_0, z)$, in fact, outperforms $\hat{d}_t(I, z)$ across several methods and datasets, i.e., inference without image information in the test domain is better than with it. Conversely, the performance gap between inference with both inputs, $\hat{d}_t(I, z)$, and just the image, $\hat{d}_t(I, z_0)$, becomes more evident under the domain shift. This observation illustrates another intuition: (ii) The domain shift largely affects the image modality, and less so depth.

The two intuitions above motivate our approach. As object shapes tend to persist across domains, and the measured sparse points being a coarse representation of them, we aim to leverage sparse depth modality to bridge the domain gap. To this end, we exploit the observation that depth completion networks are able to recover (coarse) 3D scenes from sparse points alone and that the image serves to propagate and refine depth for regions lacking points. This is done by learning to map features encoding sparse depth inputs to features encoding both modalities in the source domain and, during test-time, recover the source domain features compatible with target domain sparse depth to guide model adaptation. Specifically, as observed, the covariate shift is largely photometric, so we propose to adapt the RGB image encoder branch by introducing a *adaptation layer*: a single convolutional layer designed to align target domain RGB embeddings to those of the source domain. As the rest of the network is frozen, adapting just the adaptation layer allows for low-cost model updates.

Intuition for integrating adaptation layer. Guided by our observations, the adaptation layer should be (i) placed in the image encoder branch prior to the fusion of image and depth features, and (ii) located within later layers to modulate higher level representations (i.e., object shapes, as opposed to low-level edges). (iii) connected as a skip connection to decoder to more directly affect the output.



(a) Initialize Adaptation Layer

(b) Preparation – Learning proxy embedding

(c) Adaptation

Figure 3. *Overview*. (a) The pretraining stage integrates an adaptation layer into a pretrained encoder and pretrains the adaptation layer on the source dataset. (b) The preparation stage learns the proxy mapping of features encoding sparse depth to those encoding both inputs. (c) The adaptation stage deploys the model to the target domain and updates the adaptation layer by leveraging proxy embeddings as guidance.

3.2. Preparation Stage - Source Domain

Initialize adaptation layer from source domain. Updating the entire network is largely infeasible in test-time adaptation scenarios. For the sake of speed and efficiency, we implement an adaptation layer m_ϕ, i.e., a convolutional layer, within the encoder of a pretrained network. Note that the entire network will be frozen during all stages of our method with the exception of the adaptation layer and proxy mapping, where both will be initialized during preparation stage in the source domain; the proxy mapping will then be frozen and used to adapt m_ϕ in the target domain. To ease the adaptation process, we initialize the m_ϕ by minimizing a supervised loss over the source dataset (Fig. 3-(a)). We denote the pretrained encoder integrated with m_ϕ as e_ϕ.

Learning proxy mapping from source domain. As observed in Fig. 1, the best results in the source domain are achieved by feeding in both the image and sparse depth modalities for inference. However, the image is susceptible to domain shift which degrades performance when the model is transferred to an unseen test domain. Conversely, sparse depth is more resilient to the domain shift than RGB images, i.e., the shape of a car (or another object) remains similar regardless of (synthetic or real) domain. Our method aims to leverage the sparse depth modality, which is less sensitive to the domain shift, in the downstream adaptation process. To this end, we employ a soft mapping [15] from just the encoded sparse depth features to sparse depth *and image features* to learn the photometric information that is captured from the same scene as the sparse point cloud. This strategy allows us to learn the mapping that projects the sparse depth features to "proxy" embeddings close to those that also encode the image. In other words, it fills in what is missing in the image encoder branch by predicting the residual latent image encoding that is compatible with the input sparse depth, i.e., 3D scene. As this is trained in the source domain, the mapping naturally yields proxy embeddings that encode the source domain image (and sparse depth), which can be later used to guide the adaptation layer m_ϕ to transform test domain RGB features close to those of the source domain.

This mapping by MLPs can be denoted as $g_\psi(\cdot), g_{\psi'}(\cdot)$ and $h_\omega(\cdot)$; to learn them, we get two embeddings $\boldsymbol{p}_s$ and $\boldsymbol{q}_s$,

$$\begin{aligned} \boldsymbol{p}_s &= h_\omega(g_\psi(\text{StopGrad}(e_\phi(I_0, z_s)))), \\ \boldsymbol{q}_s &= \text{StopGrad}(g_{\psi'}(e_\phi(I_s, z_s))) \end{aligned} \tag{1}$$

where e_ϕ denotes the encoder augmented with the adaptation layer trained on source dataset, I_s, z_s, the image and sparse depth from source domain, and I_0 the null-image. The embedding modules g_ψ and h_ω are updated to maximize the similarity between $\boldsymbol{p}_s$ and $\boldsymbol{q}_s$. To learn them, we minimize:

$$\ell_{\text{prepare}} = 1 - \left(\frac{\boldsymbol{p}_s}{\|\boldsymbol{p}_s\|} \cdot \frac{\boldsymbol{q}_s}{\|\boldsymbol{q}_s\|}\right), \tag{2}$$

where $\|\cdot\|$ is $L2$-norm, and $(\boldsymbol{a} \cdot \boldsymbol{b})$ indicates the dot product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. To this end, we first train the MLP heads g_ψ, h_ω by minimizing Eqn. 2. Note that the MLP head $g_{\psi'}$ is updated with EMA update following BYOL [15] to avoid collapse: $g_{\psi'} \leftarrow \tau \cdot g_{\psi'} + (1-\tau) \cdot g_\psi$.

Once the mapping is learned, we can freeze the embedding module and deploy it for test-time adaptation where we update the adaptation layer weights ϕ to maximize the similarity between the embeddings of a test domain image and sparse depth, and its proxy from the source domain. Naturally, due to the domain shift, the embeddings will yield

low similarity scores; hence, maximizing the scores through our proxy embedding implicitly aligns the target RGB distribution to that of the source distribution, i.e., minimizing the cosine similarity between the source and target distributions.

3.3. Deploying Proxy Mapping to Target Domain

Adaptation stage aims to update the adaptation layer parameters by minimizing a test-time loss function over the target test domain data $\{I_t, z_t\} \in \mathcal{D}_t$. To do so, we deploy the learned proxy mapping module (MLP heads $\{g^*_\psi(\cdot), g^*_{\psi'}(\cdot),$ and $h^*_\omega(\cdot)\}$) along with the adaptation layer m_ϕ integrated into the frozen encoder as e_ϕ.

Adaptation loss. For adaptation, our loss is composed of a linear combination of three loss terms:

$$\mathcal{L}_{\text{adapt}} = w_z \ell_z + w_{\text{sm}} \ell_{\text{sm}} + w_{\text{proxy}} \ell_{\text{proxy}}, \tag{3}$$

where ℓ_z, ℓ_{sm} denote sparse depth consistency loss and local smoothness loss, respectively, ℓ_{proxy} is proxy mapping consistency loss, and w indicates a weight of each loss term.

Sparse Depth Consistency. Sparse point clouds capture a coarse structure of the 3D scene.To obtain metric scale predictions consistent with the scene structure, we minimize L1 error between the sparse depth z_t and the prediction $\hat{d}_t$:

$$\ell_z = \frac{1}{|\Omega(z_t)|} \sum_{x \in \Omega(z_t)} |\hat{d}_t(x) - z_t(x)|, \tag{4}$$

where $x \in \Omega(z_t)$ are the pixel locations where sparse points were projected onto the image plane.

Local Smoothness. Based on the assumption of local smoothness and connectivity in a 3D scene, we impose the same in the predicted depth map $\hat{d}_t$. Specifically, we apply an L1 penalty to its gradients in both the x- and y-directions (i.e., ∂_X and ∂_Y). We balance the weight of each term with λ_X and λ_Y, to allow discontinuities over object boundaries based on the image gradients, where $\lambda_X(x) = e^{-|\partial_X I_t(x)|}$, $\lambda_Y(x) = e^{-|\partial_Y I_t(x)|}$, and Ω denotes the image domain.

$$\ell_{\text{sm}} = \frac{1}{|\Omega|} \sum_{x \in \Omega} \lambda_X(x) |\partial_X \hat{d}_t(x)| + \lambda_Y(x) |\partial_Y \hat{d}_t(x)|. \tag{5}$$

Proxy Consistency. In order to regularize the adaptation with the learned mapping from the previous stage, we freeze the weight parameters of MLP heads $\{g^*_\psi(\cdot), h^*_\omega(\cdot)\}$, and update the parameters of the adaptation layer m_ϕ. First, we obtain the features $\boldsymbol{p}_t$ and $\boldsymbol{q}_t$ using the null-image I_0 in one and the given target test domain image I_t in the other:

$$\begin{aligned} \boldsymbol{p}_t &= \text{StopGrad}(h^*_\omega(g^*_\psi(e_\phi(I_0, z_t)))), \\ \boldsymbol{q}_t &= g^*_{\psi'}(e_\phi(I_t, z_t)). \end{aligned} \tag{6}$$

We maximize the cosine similarity between the feature $\boldsymbol{q}_t$ and $\boldsymbol{p}_t$ via a proxy loss ℓ_{proxy} to update adaptation layer m_ϕ:

$$\ell_{\text{proxy}} = 1 - \left(\frac{\boldsymbol{p}_t}{\|\boldsymbol{p}_t\|} \cdot \frac{\boldsymbol{q}_t}{\|\boldsymbol{q}_t\|}\right). \tag{7}$$

4. Experiments

We demonstrate the effectiveness of our approach on a mix of both real and synthetic datasets including indoor SLAM/VIO scenarios (VOID [51], NYUv2 [30], SceneNet [29], and ScanNet [7]) and outdoor driving scenarios using lidar sensor (KITTI [45], Virtual KITTI (VKITTI) [12], nuScenes [1], and Waymo Open Dataset [42]). We chose three representative architectures of current depth completion methods to test our method: MSG-CHN [23] (CNN-based), NLSPN [31] (SPN-based) and CostDCNet [21] (cost volume-based). All reported results are averaged over 5 independent trials. We describe implementation details, hyper-parameters used, hardware requirements, evaluation metrics as well as additional experimental results in the Supp. Mat.

Main Result. We use pretrained models (MSG-CHN, NLSPN, and CostDCNet) from the two source datasets, VOID for indoor, and KITTI for outdoor. For indoor, we adapt models pretrained on VOID to NYUv2, SceneNet, and ScanNet; for outdoors, we adapt from KITTI to VKITTI (with fog), nuScenes, and Waymo. BN Adapt denotes updating the batch statistics (i.e., running mean and variance). BN Adapt, ℓ_z, ℓ_{sm} is a variation of TENT [48] which minimizes Eqn. 4, 5 instead of entropy by updating learnable scale factors. CoTTA denotes replacing proxy loss with L1 consistency loss w.r.t. the pretrained prediction [49]. ProxyTTA-fast denotes our method without batch norm update, which improves adaptation runtime by 25.32%.

Our method consistently improves over baselines and variants of BN Adapt (Table 1). Specifically, we improve over BN Adapt, ℓ_z, ℓ_{sm} by 11.60% on average across all methods for indoor, 19.73% on outdoors, and 15.67% overall to achieve state-of-the-art performance. Qualitatively, Fig. 4 and Fig. 5 show that our method performs better in boundary regions and homogeneous regions, thus exhibiting less oversmoothing on curtains in Fig. 4-(a) and car in Fig. 5-(b), and undersmoothing on blackboard in Fig. 4-(d) and road in Fig. 5-(a), respectively, during adaptation. This trend is due to the proxy loss and the adaptation layer, which allows us to adapt with minimum weight adjustments while preserving high-level features (object shapes) learned from the source domain by mapping the target RGB modality to that of the source domain. Notably, ProxyTTA-fast still improves over BN Adapt even though we only adapt our adaptation layer, which demonstrates the effectiveness of our design choice as well as our proposed proxy embeddings. We visualize image and sparse depth features from the source and target domains along with proxy embeddings in the target domain using t-SNE [46] in Fig. 1 of Supp. Mat.; we observe that proxy embeddings are close to source domain features.

Comparison to BN adaptation[1] and CoTTA. To assess the impact of our adaptation layer, we compare to batch norm

[1] MSG-CHN lacks Batch Norm (BN) layer so we cannot use BN adapt.

Method		MAE	RMSE	MAE	RMSE	MAE	RMSE
		KITTI → VKITTI-FOG		KITTI → nuScenes		KITTI → Waymo	
MSG-CHN	Pretrained	2842.88	6557.38	3331.821	6449.094	1107.22	2962.45
	CoTTA	*730.6±11.67*	*3330.23±44.83*	*3157.69*	*6434.14*	*655.77±30.98*	*2213.27±98.80*
	ProxyTTA-fast (Ours)	**728.24±3.73**	**3087.36±15.92**	**2834.08±17.64**	**6096.56±21.08**	**608.91±1.74**	**1921.83±2.54**
NLSPN	Pretrained	1309.99	7423.48	2656.609	6146.590	1175.83	3078.377
	BN Adapt	1140.21±35.89	4592.86±198.21	11291.57±21.32	16670.87±52.56	7283.33±104.58	9670.36±250.22
	BN Adapt, ℓ_z, ℓ_{sm}	775.20±5.65	3465.05±32.73	2928.51±75.89	8209.24±164.31	*494.94±3.08*	*1921.17±338.06*
	CoTTA	767.93±5.47	3799.88±17.29	*2650.45±15.04*	6242.52±33.14	933.41±4.31	2763.88±143.48
	ProxyTTA-fast	*732.61±29.57*	*3002.19± 52.29*	2733.96±34.32	*6099.48±82.32*	875.01±15.8	2400.17±21.44
	ProxyTTA (Ours)	**686.91±22.14**	**2666.70±56.64**	**2589.25±59.03**	**6006.18±90.66**	**477.28±3.32**	**1598.64±18.95**
CostDCNet	Pretrained	1042.98	6301.60	3064.724	6630.649	1093.79	2798.25
	BN Adapt	1476.57±1.38	5428.20±8.15	2306.04±28.86	6391.98±48.97	596.08±5.55	1877.91±45.56
	BN Adapt, ℓ_z, ℓ_{sm}	*729.67±3.14*	3413.76±14.59	*2288.85±14.02*	6338.38±31.31	*469.97±2.47*	**1572.95±10.63**
	CoTTA	756.32±3.59	3686.69±14.75	2676.83±68.92	*6099.49±66.79*	689.94±1.95	2140.23±16.12
	ProxyTTA-fast	756.98±31.07	*3091.78±105.42*	2595.81±12.13	6373.01±7.74	606.10±11.10	1817.79±19.14
	ProxyTTA (Ours)	**512.72±0.74**	**2735.01±3.53**	**2062.28±11.24**	**5509.96±23.41**	**466.44±1.63**	*1580.38±11.48*
		VOID → NYUv2		VOID → SceneNet		VOID → ScanNet	
MSG-CHN	Pretrained	1040.934	1528.983	281.28	645.01	687.988	1201.747
	CoTTA	*876.93±146.95*	*1148.62±173.53*	*223.19±14.77*	*498.46±28.21*	*619.37±4.14*	*1141.04±7.35*
	ProxyTTA-fast (Ours)	**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	Pretrained	388.87	702.80	167.250	438.71	233.33	431.20
	BN Adapt	250.13±5.23	447.18±10.32	143.61±6.34	385.56±9.84	207.00±0.57	401.41±2.84
	BN Adapt, ℓ_z, ℓ_{sm}	*147.55±1.36*	*271.10±2.17*	*120.48±1.94*	*345.91±7.14*	*82.76±0.47*	*181.97±1.21*
	CoTTA	390.50±8.29	704.72±16.74	205.02±1.79	540.01±4.08	234.77±1.52	496.18±2.75
	ProxyTTA-fast	168.43±3.46	309.48±6.92	124.67±1.33	357.56±2.59	104.06±11.03	232.84±20.46
	ProxyTTA (Ours)	**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	Pretrained	189.10	446.71	173.37	443.22	144.31	458.69
	BN Adapt	160.31±2.7	410.55±10.70	176.62±0.72	446.32±8.52	159.65±4.63	399.14±13.92
	BN Adapt, ℓ_z, ℓ_{sm}	136.80±5.35	338.59±22.36	134.22±2.33	385.9±6.68	*68.44±0.46*	*164.59±2.82*
	CoTTA	147.69±5.3	376.87±21.25	136.42±3.41	405.38±11.63	101.98±1.53	322.63±5.04
	ProxyTTA-fast	*131.93±2.58*	*269.02±5.61*	*129.99±3.88*	**353.86±7.91**	128.12±3.41	244.62±7.53
	ProxyTTA (Ours)	**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 1. *Qualitative results.* For indoors, we adapt from VOID to NYUv2, SceneNet, and ScanNet; for outdoors, from KITTI to VKITTI with fog, nuScenes, and Waymo. **Bold** denotes best and *Italics* second-best. ProxyTTA-fast denotes our method without updating BatchNorm.

(BN) adaptation from TENT [48]. In BN adaptation, we only update the batch norm layer's scale and shift factor based on the loss function. On average, BN Adapt with ℓ_z, ℓ_{sm} improves the pretrained model by 32.77%; whereas, updating just our adaptation layer (ProxyTTA-fast) improves it more by 34.07% (Table 1). The improvement of ProxyTTA-fast over BN adapt demonstrates the efficacy of updating adaptation layer, which directly adjusts the high-level features from the RGB branch guided by proxy loss, where BN adapt realigns the learned source features from both RGB and range sensors by updating feature statistics.

Nonetheless, the best results are achieved when we include batch norm update, which improves the pretrained model by 44.53%, but at the cost ≈33.2% of total extra time. The improvement of ProxyTTA over BN adapt implies that the large domain discrepancy may not be addressed by adapting only BN parameters (i.e., scaling and shifting); ProxyTTA explicitly adjusts RGB features by updating the adaptation layer with proxy embeddings as guidance.

We also compared our approach to CoTTA [49], which adapts *the whole model parameters* using the prediction from the teacher model updated by exponential moving average of pretrained weight and the model prediction. We combined additional loss ℓ_z, ℓ_{sm} on top of CoTTA loss, since we observed that the models cannot be adapted with CoTTA alone. Specifically, our method without proxy shows a 25.26% average improvement on the CoTTA method. CoTTA updates the whole parameters including RGB and sparse depth branch, which causes a drift from the learned model parameters. On the other side, our method only updates additional layer at RGB branch, based on the study from the most domain discrepancy comes from RGB modality as studied in Sec. 3.1, and this prevents the model from a drift from learned domain. Also, CoTTA assumes the test-time augmentation can mitigate the domain shift. However, the results shows test-time augmentation on RGB image, causing a small distributional shift, may not solve a large domain discrepancy.

Also, our method with batch normalization layer update shows 26.52% average improvement, while using 25.05% less adaptation time. Note: CoTTA costs not only *additional memory for teacher model* but also *inference time to get the teacher model prediction*, even if CoTTA does not require any preparation process. Overall, our method shows 21.09% average improvement over BN adapt and CoTTA methods.

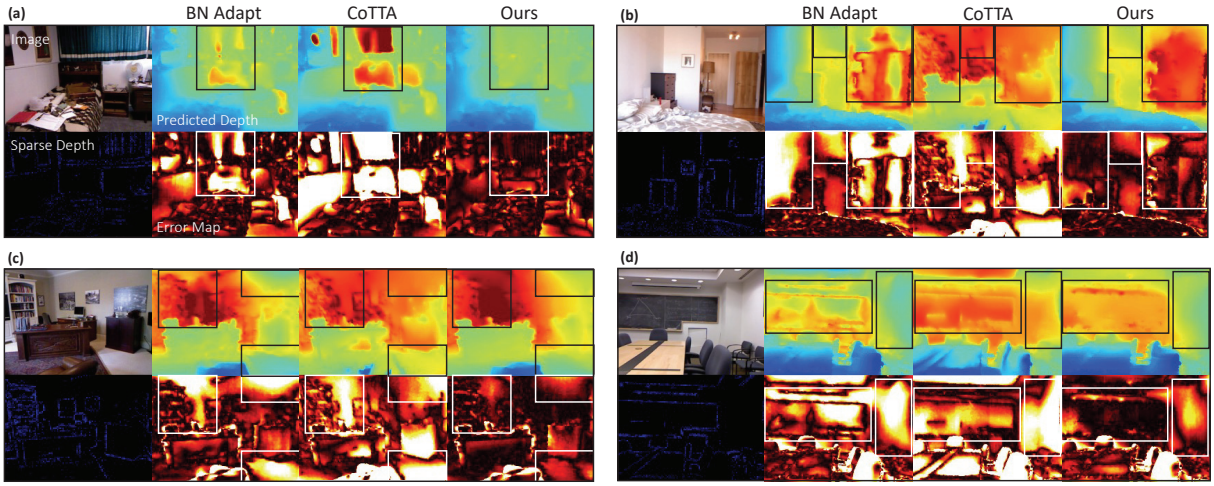


Figure 4. *Qualitative results on NYUv2.* For indoors scenarios, ProxyTTA performs better in boundary regions displaying the discontinuity in depth (e.g., curtains, (a)), as well as homogeneous regions (e.g., blackboard, (d)). Boxes highlight detailed comparisons.

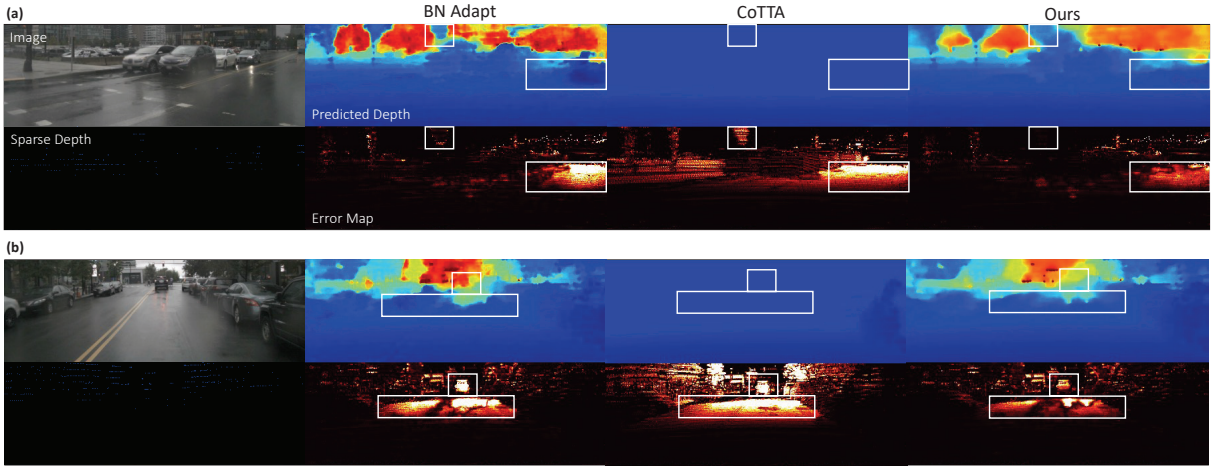


Figure 5. *Qualitative results on NuScenes.* For outdoor adaptation scenarios, ProxyTTA improves over BN Adapt and CoTTA, notably in both depth-discontinuous regions (e.g., car in (b)) and homogeneous regions (e.g., road in (a) and (b)). Boxes highlight detailed comparisons.

5. Discussion

We have proposed a method for test-time adaptation for depth completion that leverages the strength of complementary multi-sensor setup in the presence of domain shift. By studying model sensitivity to each input modality as well as the data under domain shift, we designed a way to exploit the modality (sparse depth) that is less sensitive to guide adaptation. We do so through a proxy embedding that learns the photometric information from the source domain that is compatible with the sparse depth depicting a 3D scene. Our proxy embedding works well as a regularizer for scenarios where there exists covariate shifts in photometry (i.e., KITTI to VKITTI) as well as scene layouts (i.e., VOID to NYUv2 and SceneNet). While one may surmise that the application of the embeddings are specific to scene distributions, we show otherwise. VOID (classrooms, laboratories, and gardens), NYUv2 (households and shopping centers), and SceneNet (randomly arranged synthetic rooms) all differ in layouts. The proxy embedding captures latent photometric features of the object shapes populating them; the same proxy embedding can be transferred across domains even when scene differ, but share objects within them.

This leads to possible limitations in the scenarios where the source dataset is sampled from scenes that do not share any objects with the target test dataset; in this case, the proxy embeddings should give little to no gain and one must rely on generic regularizers like local smoothness. Additionally, while we follow the conventions in TTA and assume access to the source dataset prior to deployment, in reality, many models are trained on private datasets, so adapting "off-the-shelf" models remains a challenge. In such cases, one must incorporate our preparation pipeline into their model training and release the adaptation layer and proxy embedding module together with network weights. Nonetheless, this is the first test-time adaptation work in depth completion; in addition to our findings, we release models, dataset, adaptation, and evaluation code, and hope to further motivate interest in TTA for multi-modal tasks like depth completion.

Acknowledgements. This work was supported by NSF 2112562 Athena AI Institute.

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. 6, 13

[2] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022. 2

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

[4] 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. 3, 12

[5] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *ECCV*, pages 440–458. Springer, 2022. 2

[6] Safa Cicek and Stefano Soatto. Unsupervised domain adaptation via regularized conditional alignment. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1416–1425, 2019. 3

[7] 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. 6, 13

[8] Yiming Dou, Fengyu Yang, Yi Liu, Antonio Loquercio, and Andrew Owens. Tactile-augmented radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[9] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Propagating confidences through cnns for sparse data regression. *arXiv preprint arXiv:1805.11913*, 2018. 3

[10] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12014–12023, 2020. 3

[11] Xiaohan Fei, Alex Wong, and Stefano Soatto. Geo-supervised visual depth prediction. *IEEE Robotics and Automation Letters*, 4(2):1661–1668, 2019. 12

[12] 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, 13

[13] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 2, 3

[14] 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. 12

[15] 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. 5, 14

[16] Christopher G. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 13

[17] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13656–13662. IEEE, 2021. 3

[18] Zixuan Huang, Junming Fan, Shenggan Cheng, Shuai Yi, Xiaogang Wang, and Hongsheng Li. Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. *IEEE Transactions on Image Processing*, 29: 3429–3441, 2019. 2

[19] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. 34:2427–2440, 2021. 2

[20] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with cnns: Depth completion and semantic segmentation. In *2018 International Conference on 3D Vision (3DV)*, pages 52–60. IEEE, 2018. 3

[21] Jaewon Kam, Jungeon Kim, Soongjin Kim, Jaesik Park, and Seungyong Lee. Costdcnet: Cost volume based depth completion for a single rgb-d image. In *European Conference on Computer Vision*, pages 257–274. Springer, 2022. 3, 6

[22] Youngeun Kim, Donghyeon Cho, Kyeongtak Han, Priyadarshini Panda, and Sungeun Hong. Domain adaptation without source data. *IEEE Transactions on Artificial Intelligence*, 2(6):508–518, 2021. 2

[23] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 32–40, 2020. 3, 6

[24] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1638–1646, 2022. 3

[25] Tian Yu Liu, Parth Agrawal, Allison Chen, Byung-Woo Hong, and Alex Wong. Monitored distillation for positive congruent depth completion. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 35–53. Springer, 2022. 2

[26] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? 34:21808–21820, 2021. 2, 3

[27] 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. 3

[28] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3288–3295. IEEE, 2019. 2, 3

[29] 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, 6, 13

[30] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 6

[31] 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. 3, 6, 12

[32] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 2, 3

[33] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 3313–3322, 2019. 3

[34] Chao Qu, Ty Nguyen, and Camillo Taylor. Depth completion via deep basis fitting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 71–80, 2020. 3

[35] Chao Qu, Wenxin Liu, and Camillo J Taylor. Bayesian deep basis fitting for depth completion with uncertainty. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16147–16157, 2021. 3

[36] Kyeongha Rho, Jinsung Ha, and Youngjung Kim. Guideformer: Transformers for image guided depth completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6250–6259, 2022. 3

[37] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Sparsh Garg, In So Kweon, and Kuk-Jin Yoon. Mm-tta: multi-modal test-time adaptation for 3d semantic segmentation. In *CVPR*, pages 16928–16937, 2022. 2

[38] 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. 2

[39] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012. 13

[40] Akash Deep Singh, Yunhao Ba, Ankur Sarker, Howard Zhang, Achuta Kadambi, Stefano Soatto, Mani Srivastava, and Alex Wong. Depth estimation from camera image and mmwave radar point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9275–9285, 2023. 3

[41] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11920–11929, 2023. 2

[42] 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. 6, 13

[43] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019. 3

[44] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, pages 9229–9248. PMLR, 2020. 2, 3

[45] 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. 2, 6, 12, 13

[46] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 6

[47] 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. 3

[48] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 2, 6, 7

[49] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 2, 6, 7

[50] Alex Wong and Stefano Soatto. Unsupervised depth completion with calibrated backprojection layers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12747–12756, 2021. 2

[51] 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. 2, 6, 12

[52] 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.

[53] 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

[54] Yangchao Wu, Tian Yu Liu, Hyoungseob Park, Stefano Soatto, Dong Lao, and Alex Wong. Augundo: Scaling up augmentations for unsupervised depth completion. *arXiv preprint arXiv:2310.09739*, 2023. 2

[55] 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/CVF International Conference on Computer Vision*, pages 2811–2820, 2019. 3

[56] Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. Touch and go: Learning from human-collected vision and touch. *Neural Information Processing Systems (NeurIPS) - Datasets and Benchmarks Track*, 2022. 2

[57] Fengyu Yang, Jiacheng Zhang, and Andrew Owens. Generating visual scenes from touch. *International Conference on Computer Vision (ICCV)*, 2023.

[58] Fengyu Yang, Chao Feng, Ziyang Chen, Hyoungseob Park, Daniel Wang, Yiming Dou, Ziyao Zeng, Xien Chen, Rit Gangopadhyay, Andrew Owens, and Alex Wong. Binding touch to everything: Learning unified multimodal tactile representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[59] 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

[60] Zhang Youmin, Guo Xianda, Poggi Matteo, Zhu Zheng, Huang Guan, and Mattoccia Stefano. Completionformer: Depth completion with convolutions and vision transformers. *arXiv preprint arXiv:2304.13030*, 2023. 3

[61] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185, 2018. 3

Supplementary Materials

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 comparison of qualitative results on Waymo dataset's adverse weather sample from pretrained and adapted model to show the effectiveness of our adaptation under adverse weather conditions.
- In Section E, we present the hyperparameter settings for result reproduction and we elucidate evaluation details.
- In Section F, we provide a study on the learned proxy embedding with a visualization.
- In Section G, we present an ablation study of the loss components in our method.
- In Section H, we present the results on KITTI → VKITTI adaptation.
- In Section I, we present the results on a different source dataset (Waymo → VKITTI).
- In Section J, 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 2.

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 2. *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 [4, 31] 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 3. 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 [14] 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 [45]. 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 [51] 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 [11], 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 planar motion. We use a training set of ≈46K

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 3. *Model sensitivity to input modalities with varying sparsity.*

images to prepare the model.

NYUv2 [39] 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 [16] to sample ≈1,500 points from the depth maps. We use a set of 654 test set images for adaptation.

ScanNet [7] 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 [16] to sample ≈1,500 points from the depth maps. We use a set of ≈21K test images for adaptation.

Virtual KITTI (VKITTI) [12] 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 [45] videos. There exists a large domain gap between RGB images from VKITTI and KITTI, even though the virtual worlds created in Unity by [12] 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 [29] 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 [16] to mimic the sparse depth produced by SLAM/VIO. The final 375 corners

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 4. *Hyperparameters.* For MSG-CHN, NLSPN, and CostDC-Net methods for initialization, preparation, and adaptation.

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 [42] 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

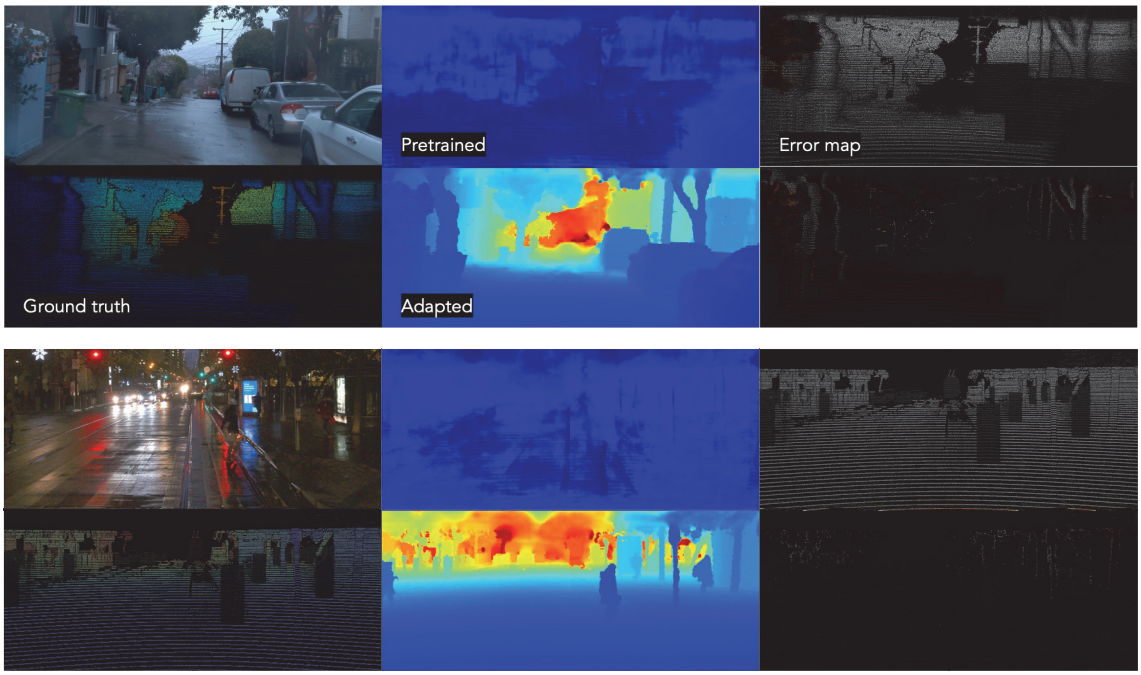


Figure 6. *Qualitative results on Waymo. For outdoor adaptation scenarios, ProxyTTA can adapt under the adverse weather condition, such as raining condition (top row) and low-illumination (bottom row).*

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. Qualitative results on adverse weather conditions

Typically, in real-world scenarios, most systems will encounter non-ideal sensing conditions, which will degrade performance. For example, existing pretrained depth (completion) models will fail under adverse weather conditions, such as nighttime (low-illumination) or rain. To address such failure modes of existing models, we adapt CostDCNet using Proxy-TTA. We demonstrate this capability in Fig. 6, where we improve over the pretrained model significantly as shown in the range and error map visualizations.

E. Implementation details

Hyperparameter. We specifically note the hyperparameters of three methods for initialization, preparation, and adaptation on Table 4.

Epochs and training details Adaptation occurs in a single epoch, with 'the number of iterations per data point' (*inner-iter*) specified in Tab. 4. 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 [15]; 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 6. 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.

F. Discussion on learned proxy embeddings

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

Fig. 7 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.

Method	ℓ_z	ℓ_{sm}	ℓ_{proxy}	KITTI → Waymo MAE	KITTI → Waymo RMSE	KITTI → VKITTI-FOG MAE	KITTI → VKITTI-FOG RMSE	KITTI → nuScenes MAE	KITTI → nuScenes 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 5. *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 6. *Error metrics.* d_{gt} denotes the ground-truth depth.

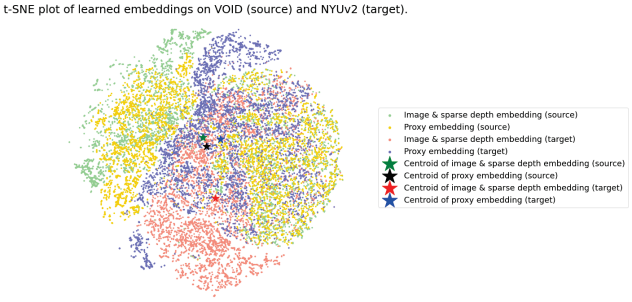


Figure 7. *t-SNE plot of learned embeddings on VOID and NYUv2.*

Method		KITTI → VKITTI MAE	KITTI → VKITTI 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 7. *Additional results for test-time adaptation for depth completion on KITTI → VKITTI.*

G. Ablation study

Here, we ablate the effect of each loss term denoted with the checkmarks in Table 5. 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. 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,

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.

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

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.

H. KITTI → VKITTI results

Here, we present additional results on KITTI → VKITTI adaptation. Test-time adaptation results are shown in Table 7. 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.

I. 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 9. Our method shows an improvement over CoTTA and BN Adapt by 21.70%.

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.

J. Quantitative preliminary results

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