



Project to Adapt: Domain Adaptation for Depth Completion from Noisy and Sparse Sensor Data

Adrian Lopez-Rodriguez¹ · Benjamin Busam^{2,3} · Krystian Mikolajczyk¹

Received: 30 April 2021 / Accepted: 21 November 2022 / Published online: 16 December 2022
© The Author(s) 2022

Abstract

Depth completion aims to predict a dense depth map from a sparse depth input. The acquisition of dense ground-truth annotations for depth completion settings can be difficult and, at the same time, a significant domain gap between real LiDAR measurements and synthetic data has prevented from successful training of models in virtual settings. We propose a domain adaptation approach for sparse-to-dense depth completion that is trained from synthetic data, without annotations in the real domain or additional sensors. Our approach simulates the real sensor noise in an RGB + LiDAR set-up, and consists of three modules: simulating the real LiDAR input in the synthetic domain via projections, filtering the real noisy LiDAR for supervision and adapting the synthetic RGB image using a CycleGAN approach. We extensively evaluate these modules in the KITTI depth completion benchmark.

Keywords Depth completion · Domain adaptation · LiDAR · Sensor fusion

1 Introduction

Motivation Active sensors such as LiDAR determine the distance of objects within a specified range via a sparse sampling of the environment whose density decreases quadratically with the distance. RGB cameras densely capture their field of view, however, monocular depth estimation from RGB is an ill-posed problem that can be solved only up to a geometric scale. The combination of RGB and depth modalities form a rich source for mutual improvements where each sensor can benefit from the advantage of the other.

Many pipelines have been proposed for a fusion of these two inputs [46,47,54,66,74,76]. Ground-truth annotations for this task, however, require elaborate techniques, man-

ual adjustments and are subject to hardware noise or costly and time-consuming labeling. The most prominent publicly available data for this task [65] creates a ground-truth by aligning consecutive raw LiDAR scans that are cleaned from measurement errors, occlusions, and motion artifacts in a post-processing step involving classical stereo reconstruction. Even after the use of this additional data and tedious processing, the signal is not noise-free as discussed in [65]. To avoid the need for such annotations, some methods perform self-supervision [47,69–71,76], where a photometric loss is employed with stereo or video data. The dependence on additional data such as stereo or temporal sequences brings other problems such as line-of-sight issues and motion artifacts from incoherently moving objects. Modern 3D engines can render highly realistic virtual environments [13,20,58] with perfect ground-truth. However, a significant domain gap between real and virtual scenes prevents from successful training on synthetic data only.

Contributions and Outline In contrast to the self-supervised methods [47,69–71,76], we propose to use a domain adaptation approach to address the depth completion problem without real data ground-truth as shown in Fig. 1. We train our method from the synthetic data generated with the driving simulator CARLA [13] and evaluate it on the real KITTI depth completion benchmark [65]. The real LiDAR data is noisy with the main source of noise being the see-through

Communicated by Cheng-Lin Liu.

✉ Adrian Lopez-Rodriguez
al4415@imperial.ac.uk

Benjamin Busam
b.busam@tum.de

Krystian Mikolajczyk
k.mikolajczyk@imperial.ac.uk

¹ Imperial College London, London, UK

² Technische Universität München, Munich, Germany

³ Huawei Noah's Ark, London, UK

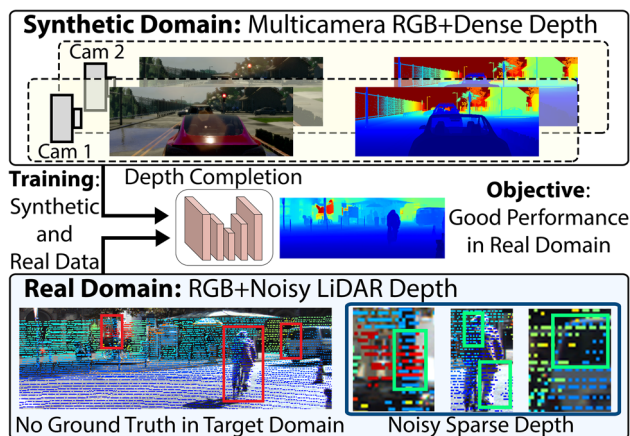


Fig. 1 We investigate the depth completion problem without ground-truth in the real domain, which contains paired noisy and sparse depth measurements and RGB images. We highlight some noise present in the real data: see-through on the tree trunk and bicycle, self-occlusion on the bicycle and missing points on the van. We leverage synthetic data with multicamera dense depth and RGB images

artifacts that occur after projecting the LiDAR’s point cloud to the RGB cameras. We aim to generate synthetic LiDAR data with a noise distribution similar to real LiDAR data. Hence, we propose an approach to simulate the see-through artifacts by generating data in CARLA using a multicamera set-up (shown in Fig. 2), employing random masks from the real LiDAR to sparsify the virtual LiDAR map, and projecting from the virtual LiDAR camera to the RGB reference frame. We further improve the model by filtering the noisy input in the real domain to obtain a set of reliable points that are used as supervision. Finally, to reduce the domain gap between the RGB images, we use a CycleGAN [82] to transfer the image style from the real domain to the synthetic one.

We compare our approach to other state-of-the-art depth completion methods and provide a detailed analysis of the proposed components. The proposed domain adaptation for RGB-guided sparse-to-dense depth completion is a novel approach for the task of depth completion, which leads to significant improvements over using non-adapted synthetic data as demonstrated by the results. To this end, our main contributions are:

1. A novel domain adaptation method for depth completion that includes geometric and data-driven sensor mimicking, noise filtering and RGB adaptation.
2. We demonstrate that adapting the synthetic sparse depth is crucial for improving the performance, whereas RGB adaptation is secondary.
3. An improvement in depth completion performance by all of the proposed modules in the KITTI depth completion benchmark.

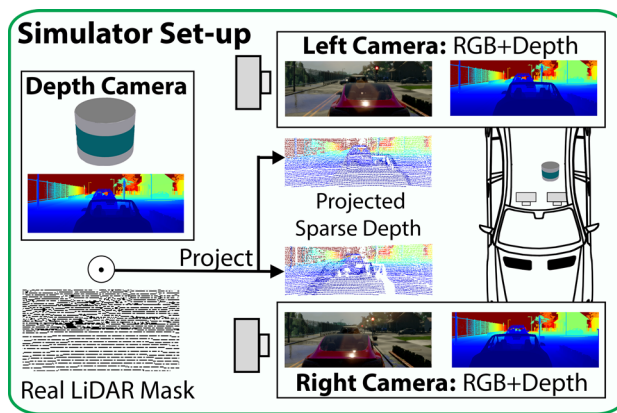


Fig. 2 Overview of the multicamera set-up in CARLA used to simulate the real projection LiDAR artifacts. The Depth Camera acts as a virtual LiDAR and collects a dense depth, which is sparsified using real LiDAR binary masks and projected to either the Left Camera or Right Camera reference frame. Both the Left Camera and the Right Camera collect RGB information, used as part of the input data, and a dense depth map, used for supervision

We extend [43] by adding experiments that explore: the effect of varying the hyperparameters and sparsity levels, the distribution of errors depending on the semantic class and distances to the camera, and the use of an image translation module less complex than CycleGAN. We also include a computational complexity analysis, extra qualitative results comparing our method to the state-of-the-art, and we highlight some issues with the available KITTI ground-truth that affect the predictions of models trained with such ground-truth. Furthermore, we strengthen the motivation for the proposed synthetic projections by showing the input depth noise distribution for both the real data and our adapted synthetic data.

2 Related Work

We first review related works on depth estimation using only either RGB or LiDAR as input, and then discuss depth completion methods from both RGB and LiDAR.

2.1 Unimodal Approaches

RGB Images RGB based depth estimation has a long history [37,60,63] reaching from temporal Structure from Motion (SfM) [18,30] and SLAM [17,27,49] to recent approaches that estimate depth from a static image [22,23,25,36,40]. Networks are either trained with full supervision [14,36] or use additional cameras to exploit photometric consistency during training [22,53]. Some monocular depth estimators leverage a pre-computation stage with an SfM pipeline to provide supervision for both camera pose and

depth [34,75] or incorporate hints from stereo algorithms [68]. These approaches are in general tailored for a specific use case and suffer from domain shift errors, which has been addressed with stereo proxies [25] or various publicly available pretraining sources [40,56].

Sparse Depth While recent advantages in depth super-resolution [45,57,67] show good performance, they are not directly applicable to LiDAR data which is sparsely and irregularly distributed within the image. Classical image processing techniques adapted to handle LiDAR sparse signals are used in [35], which contain hand-crafted elements that depend on the input sampling pattern. An encoder-decoder architecture is applied for this task in [12,15,32,65], which have better performance and ability to generalize to other sampling patterns than non-learning approaches [46]. These methods using only sparse depth as input usually outperform those leveraging only RGB images. However, the combination of RGB and sparse depth modalities can increase the depth completion performance compared to either using only RGB [5,66,77] or sparse depth [47,66] information.

2.2 Depth Completion from RGB and LiDAR

Although there are some classical methods that leverage both RGB and sparse depth modalities [4,19], most recent solutions to depth completion from RGB and LiDAR use deep neural networks due to their increased performance compared to non-learning methods [54]. These deep learning methods can be divided into supervised and self-supervised approaches.

Supervision and Ground Truth Usually, an encoder-decoder network is used to encode the different input signals into a common latent space, where feature fusion is possible, and a decoder reconstructs an output depth map [38,46,54,66,74], while fusing 2D and 3D representations [8] can improve depth boundaries. Spatial Propagation Networks [41], along with its convolutional [9,11] and non-local variants [50], refine depth predictions learning affinities from neighbouring pixels. Other methods [54,74] leverage different input modalities such as surface normals to increase the amount of diversity in the input data. Some methods have also analyzed the effect of the input depth pattern [73], including methods proposing adaptive sampling approaches [6,24]. Similarly to our approach, some works consider the case where the input depth may have noise when designing the architecture of the depth completion model. To reduce the effect of the input noise, FusionNet [66] fuses local and global information with a confidence map, whereas PENet [29] instead fuses the output of a color-dominant branch and a depth-dominant branch also using a confidence map. In [29,66], the confidence maps are used internally to fuse multiple-branch information, whereas other works focus on producing meaningful

high-quality uncertainty predictions [16,55]. Most of these methods test their performance in either NYU-v2 dataset [61] by synthetically sparsifying the ground-truth, hence assuming the input sparse data is noise-free, or in the publicly available dataset KITTI [21,65]. KITTI includes real driving scenes where a stereo RGB camera system is fixed on the roof of a car along with a LiDAR scanner that acquires data while the car is driving. A post-processing stage fuses several LiDAR scans and filters outliers with the help of stereo vision to provide labeled ground-truth. While this process is intricate and time-consuming, a further error is accumulated from calibration and alignment [65].

Self-Supervised Approaches Obtaining ground-truth data for LiDAR depth completion is difficult due to the need for accurate depth that is denser than the input LiDAR. For that reason, self-supervised approaches, which instead leverage a view either from a second camera or a video sequence for supervision, can be used to reduce the need for real-domain ground-truth. Self-supervision for depth completion with a photometric loss and predicted relative poses between RGB video frames was proposed in [47], and other works followed suit by adding a self-supervision loss term in their method [69–71,76]. A probabilistic formulation was proposed in [76] with a conditional prior within a maximum a-posteriori (MAP) estimation, which also leverages stereo information. [70] introduced a scaffolding block as a first step, which is a non-learning method that forms a spatially dense but coarse depth approximation from the sparse points. The coarse approximation was then refined using a trained network. This non-learning scaffolding approach assumes piece-wise planar geometry and is prone to errors in objects with complex geometry. Thus, further work [71] combined this scaffolding idea with a learning approach leveraging synthetic data to learn topology from sparse depth. Similarly to [71], the recent KBNNet [69] densifies first the sparse depth before inputting any RGB information. However, KBNNet also uses the intrinsics parameters during training and inference to lift and process the depth information in the 3D space, which improves the performance. Our method can be combined with these recent advances in self-supervised approaches for improved performance. Furthermore, compared to our domain adaptation approach, these self-supervised methods present less accurate predictions around the edges of a given object and have a lower ability to correct possible artifacts in the input LiDAR, as we will show in the qualitative examples given in Sect. 4.

Synthetic Data Unlike real-domain ground-truth collection, synthetic depth ground-truth can be easily generated in large quantities. Thus, in our work, we aim to leverage synthetic data to train a depth completion approach with good performance in the real domain. With that aim, we adopt a domain adaptation approach, where we adapt the synthetic-domain

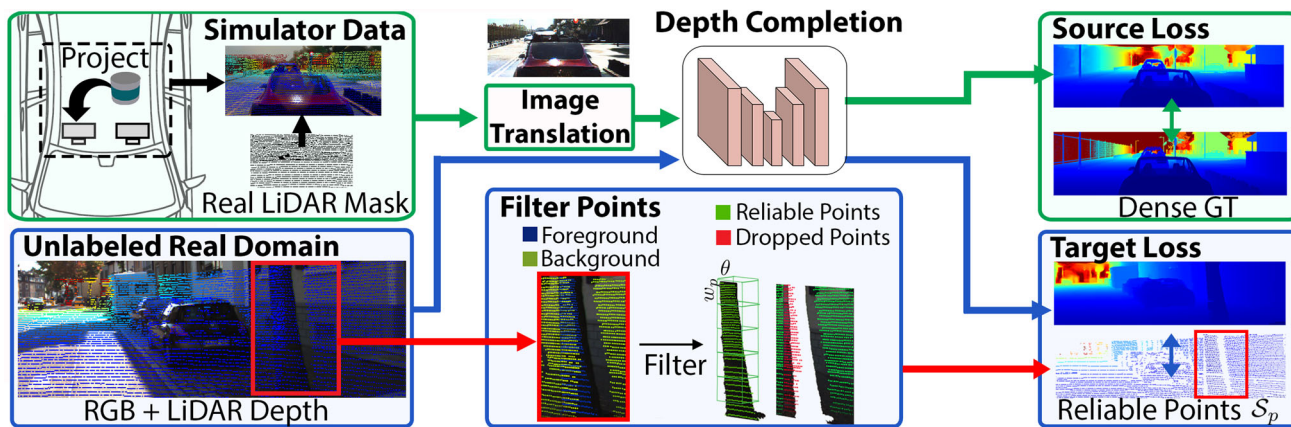


Fig. 3 Overview of our method. We use a simulator with a multicamera set-up and real LiDAR binary masks to transform the synthetic dense depth map into a noisy and sparse depth map. We train in two steps: green blocks are used in the 1st and 2nd step of training while blue

blocks in the 2nd step only. In the filtering block, green points are the reliable points S_p and red points are dropped. The *Image Translation* network is pretrained using a CycleGAN approach [82] (Color figure online)

input to be similar to a real LiDAR and RGB set-up. For monocular depth estimation, past works already used domain adaptation approaches by leveraging style-transfer methods [2,42,81]. Sparse-to-dense methods, however, have used synthetic data without any adaptation apart from sampling depth values from the dense depth map [3,54,71,76] so far. Regarding synthetic data, multiple simulators for driving scenarios have also been researched: SYNTHIA [58] provides synthetic urban images together with semantic annotations, while Virtual KITTI [20] gives synthetic renderings that closely match the videos of the KITTI dataset [21] including semantic and depth ground-truth. A LiDAR simulator using ray-casting and a learning process to drop points was proposed in [48], which was tested in detection and segmentation tasks, but is not publicly available. The CARLA simulator [13] allows for photo-realistic simulations of driving scenarios, which we utilize to generate realistic RGB images. While LiDAR scans can be simulated with CARLA via ray-casting, the car shapes in the returned LiDAR depth are approximated with cuboids, thus losing detail. For that reason, we leverage the CARLA z-buffer to obtain fine-granular depth and then sparsify the signal to simulate LiDAR scans.

3 Method

Our method, shown in Fig. 3, consists of two main components that include an adaptation of the synthetic data to make it similar to the real data, as well as a retrieval of reliable supervision from the real but noisy LiDAR signal.

3.1 Data Generation via Projections

Supervised depth completion methods strongly rely on the sparse depth input, achieving good performance without RGB information [47,66]. To train a completion model from synthetic data that works well in the real domain we need to generate a synthetic sparse input that reflects the real domain distribution. Instead of simulating a LiDAR via ray-casting, which can produce more realistic results but is more computationally expensive and harder to implement [78], we leverage the z-buffer of our synthetic rendering engine to provide a dense depth ground-truth at first. We thus aim to transform this synthetic dense data into a sparse depth resembling a real LiDAR sparse input.

Previous approaches used synthetic sparse data to evaluate a model in indoor scenes or synthetic outdoor scenes [31,32] or to pretrain a depth completion model [54,71,76]. To sparsify the data a Bernoulli distribution per pixel is used in some works [31,32,46] which, given a probability p_B and a dense depth image x_D , samples each of the pixels by either keeping the value of the pixel with probability p_B or setting its value to 0 with probability $(1 - p_B)$, thus generating the sparse depth x_D^B . We argue that a model trained with x_D^B does not perform well in the real domain, and our results in Sect. 4 support this observation. There are two reasons for the drop in performance in the real LiDAR data. Firstly, the distribution of the points x_D^B does not follow the LiDAR sparse distribution. Secondly, there is no noise in the sampled points, as we directly sample from the ground-truth. Regarding the noise in the input data, Fig. 4 shows the distribution of errors in the input sparse data for the training examples in the KITTI dataset. Most of the LiDAR sparse depth contains either no noise ($\approx 65\%$) or a low amount of error, e.g., $\approx 28\%$ of the

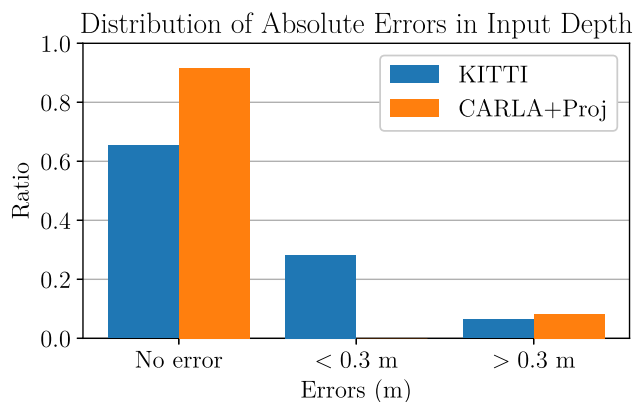


Fig. 4 Distribution of errors in the input depth for the KITTI training set and for our collected CARLA dataset with our projection module

data contain errors of less than 30 cm. However, the problematic points are the small percentage of input depth points with a large amount of error, (e.g., only 7% of the points have an error larger than 30 cm), as those points may have errors in the range of tens of meters. If we train a network with synthetic data containing no noise, the model will not learn to correct the input points and will also propagate the large error from the incorrect depth values to nearby pixels in the output prediction (as shown in the examples in Fig. 7). However, instead of adding random noise to the synthetic input depth, we posit that the large error in some depth values is mainly a consequence of the projection from the LiDAR to the RGB camera, which we aim to simulate to reduce the domain gap between the synthetic and the real-domain input depth.

Mimicking LiDAR Sampling Distribution To simulate a pattern similar to a real LiDAR, we propose to sample at random the real LiDAR inputs from the real domain similarly to [3]. We use the real sparse LiDAR depth to generate a binary mask M_L , which is 1 for those pixels with valid depth in the sparse LiDAR input and 0 in the rest. We then apply the masks to the dense synthetic depth data by $x_D^M = M_L \odot x_D$. This approach adapts the synthetic data directly to the sparsity level in the real domain without the need to tune it depending on the LiDAR used.

Generating Projection Artifacts As mentioned, previous works use noise-free sparse data to pretrain [54,71] or evaluate a model [32] with synthetic data. However, simulating the noise of real sparse data can reduce the domain gap and improve the adaptation result. Real LiDAR depth contains noise from several sources including the asynchronous acquisition due to the rotation of lasers, dropping of points due to low surface reflectance and projection errors. Simulating a LiDAR sampling process by modelling all of these noise sources can be costly and technically difficult as a physics-based rendering engine with additional material properties

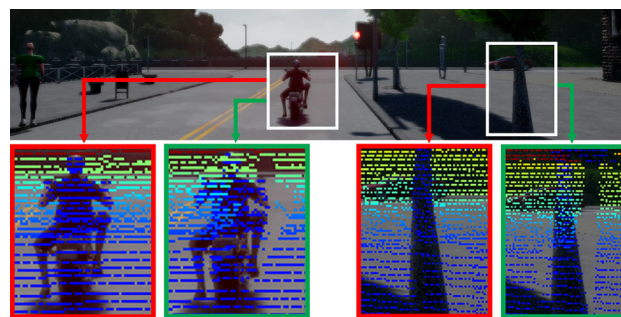


Fig. 5 Example of the generated projection artifacts in the simulator. The zoomed-in areas marked with red rectangles correspond to x_D^M and the zoomed-in areas marked with green rectangles correspond to x_D^P , where we can see simulated projection artifacts, e.g., see-through points on the left side of the motorcyclist (Color figure online)

is necessary to simulate the photon reflections individually. We propose a more pragmatic solution that allows us to use the z-buffer of a simulator by assuming that the dominating noise is a consequence of the point cloud projection from the LiDAR to the RGB camera reference frame. For that set-up, the error is a result of projecting a sparse point cloud arising from another viewpoint, as we do not have a way to filter the overlapping points by depth. This creates see-through patterns that do not respect occlusions as shown in Fig. 5, which is also observed in the real domain [10]. Therefore, a simple point drawing from a depth map at the RGB reference cannot recreate this effect and such a method does not perform well in the real domain.

To recreate this pattern, we use the CARLA simulator [13], which allows us to capture multicamera synchronized synthetic data. Our CARLA set-up mimics the camera distances in KITTI [21], as our benchmark is the KITTI depth completion dataset [65]. Instead of LiDAR, we use a virtual dense depth camera. The set-up is illustrated in Fig. 2. As the data is synthetic, the intrinsic and extrinsic parameters needed for the projections are known. After obtaining the depth from the virtual LiDAR camera, we sparsify it using the LiDAR masks resulting in x_D^M , which is then projected onto the RGB reference with

$$x_D^P = K_{RGB} P_{RGB}^L K_L^{-1} x_D^M \quad (1)$$

where K_L , K_{RGB} are the LiDAR and RGB camera intrinsics and P_{RGB}^L is the rigid transformation between the LiDAR and RGB reference frame. The resulting x_D^P is the projected sparse input to either the left or right camera.

Figure 4 shows the distribution of errors in the input points after the projection step. Our projection module generates a comparable amount of input sparse depth values with an error larger than 30 cm in the synthetic data compared to the real data. The similar distribution of noise between real and synthetic data resulting after our projection module reduces

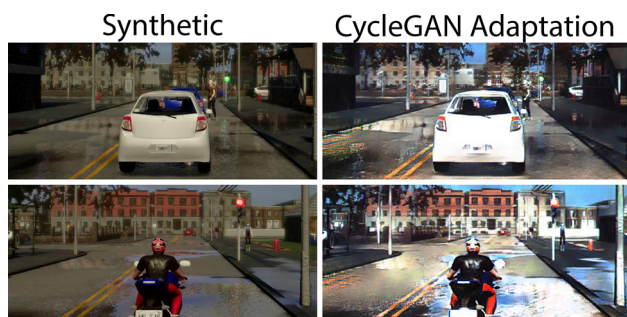


Fig. 6 We reduce the domain gap in the RGB modality using a CycleGAN approach. We show synthetic CARLA images and the resulting adapted images

the input depth domain gap, and allows the network to learn from the synthetic data to correct those artifacts. Figure 4 also shows that our projection module does not simulate the low-scale errors, i.e., those with an error lower than 30 cm, which may be a consequence of other sources of error in the LiDAR data, and needs to be addressed in further work.

3.2 RGB Adaptation

Similarly to domain adaptation for depth estimation methods [2,42,80,81], we address the domain gap in the RGB modality with style translation from synthetic to real images. Due to the complexity of adapting high-resolution images, we first train a model to translate from synthetic to real using a CycleGAN [82] approach. The generator is not further trained and is used to translate the synthetic images to the style of real images, thus reducing the domain gap as shown in Fig. 6.

3.3 Filtering Projection Artifacts for Supervision

In a depth completion setting, the given LiDAR depth can also be used as supervision data, as in [47]. Nevertheless, the approach in [47] did not take into account the noise present in the data. The given real-domain LiDAR input is accurate in most points with an error of only a few centimeters as shown in Fig. 4. However, due to the noise present, some points cannot be used for supervision, such as the see-through points, which have errors in the order of meters. Another method [10] also used the sparse input as guidance for LiDAR-stereo fusion while filtering the noisy points using stereo data. However, we aim to remove the noisy inputs without relying on additional data, such as a stereo pair, as this may not always be available.

Thus, to supervise the real data, we first aim to find a set of reliable sparse points in the real-domain depth, \mathcal{S}_p , that contains those input LiDAR points that are likely to have low error. Using \mathcal{S}_p for supervision can lead to better performance compared to using the original input LiDAR due to its

reduced amount of noise. To obtain the set \mathcal{S}_p , we propose to apply a filter to the real-domain input LiDAR points. This filter is designed based on the assumption used in Sect. 3.1 to generate our data, i.e., the main source of error in the input LiDAR points are the see-through artifacts after projecting from the LiDAR reference frame to the RGB camera. Following this see-through assumption, we can assume that in any given local window there are two modes of depth distribution, approximated by a closer and a further plane. We show an overview of the idea in Fig. 3. The points from the closer plane are more likely to be correct as part of the occluding objects. To retrieve \mathcal{S}_p we apply a minimum pooling with window size w_p yielding a minimum depth value d_m per window. Then, we include in \mathcal{S}_p the points $s \in [d_m, d_m + \theta]$ where θ is a local thickness parameter of an object. The number of noisy points not filtered out depends on the window w_p and object thickness θ , e.g., larger windows remove more points but the remaining points are more reliable. We use the noise rate η , which is the fraction of noisy points as introduced in noisy labels literature [26,39,79], to select w_p and θ in the synthetic validation set, thus not requiring any ground-truth in the real domain. Section 4 shows that using a large object thickness parameter θ or a small window size w_p leads to a higher noise rate due to an increased tolerance of the filter. As our proposed filter is not able to remove all of the see-through artifacts in the real input LiDAR, after the filtering step a certain number of noisy elements will remain in \mathcal{S}_p and will be used for supervision. Due to the supervision performed with our adapted synthetic data, our network is able to correct some of the input noise present in the real LiDAR. Hence, the noisy points in \mathcal{S}_p are more likely to be further away from the dense depth prediction \hat{y} than the noise-free points in \mathcal{S}_p . Thus, the Reverse Huber (BerHu) loss [85], which we use for supervision in the synthetic domain and behaves as a squared error loss for larger errors and linearly for lower errors (see Eq. 4), would give more weight to those noisy points in \mathcal{S}_p and, in turn, reduce the performance of the method. To provide extra robustness against these false positives, we use in the real domain a Mean Absolute Error (MAE) loss, as MAE weights all values equally, showing more robustness to the noisy outliers.

3.4 Summary of Losses

Our proposed loss is

$$\mathcal{L} = \lambda_S \mathcal{L}_S + \lambda_R \mathcal{L}_R \quad (2)$$

where \mathcal{L}_S is the loss used for the synthetic data, \mathcal{L}_R the loss used for the real data and λ_S and λ_R are hyperparameters. We use a two-step training approach similar to past domain adaptation works using pseudo-labels [62,84], aiming first for good performance in the synthetic data before introduc-

ing noise in the labels. First, we set $\lambda_S = 1.0$ and $\lambda_R = 0.0$, to train only from the synthetic data. For \mathcal{L}_S we use a Reverse Huber loss [85], which works well for depth estimation problems [36]. Hence, we define \mathcal{L}_S as

$$\mathcal{L}_S = \frac{1}{b_S} \sum_i \frac{1}{n_i} \sum_k \mathcal{L}_{bh}(\hat{y}_k, y_k) \quad (3)$$

where b_S is the synthetic batch size, n_i the number of ground-truth points in image i , \hat{y} is the predicted dense depth, y is the ground-truth depth and the Reverse Huber loss \mathcal{L}_{bh} [85] is

$$\mathcal{L}_{bh}(\hat{y}_k, y_k) = \begin{cases} \frac{(\hat{y}_k - y_k)^2 + c^2}{2c} & \text{if } |\hat{y}_k - y_k| > c \\ |\hat{y}_k - y_k| & \text{if } |\hat{y}_k - y_k| \leq c \end{cases} \quad (4)$$

where $c = \delta \cdot \max_k |\hat{y}_k - y_k|$ and we set $\delta = 0.05$, which is the default value in the official code of the model we use [66].

In the second step we set $\lambda_S = 1.0$ and $\lambda_R = 1.0$ as we introduce real domain data into the training process using \mathcal{S}_p for supervision. We define \mathcal{L}_R as

$$\mathcal{L}_R = \frac{1}{b_R} \sum_i \frac{1}{\#(\mathcal{S}_{p,i})} \sum_k |\hat{y}_k - y_k| \quad (5)$$

where b_R is the real domain batch size and $\#(\mathcal{S}_{p,i})$ is the cardinality of the set of reliable points \mathcal{S}_p for an image i .

4 Experiments

We use PyTorch 1.3.1 [51] and an NVIDIA 1080 Ti GPU, as well as the official implementation of FusionNet [66] as our sparse-to-dense architecture. The batch size is set to 4 and we use Adam [33] with a learning rate of 0.001. For the synthetic data, we train by randomly projecting to the left or right camera with the same probability. In the first step of training, we use only synthetic data (i.e., we set the loss weight and batch size for the synthetic data to $\lambda_S = 1.0$ and $b_S = 4$, whereas for the real data they are set to $\lambda_R = 0.0$ and $b_R = 0$) until performance plateaus in the synthetic validation set. In the second step, we mix real and synthetic data setting $\lambda_S = 1.0$, $\lambda_R = 1.0$, $b_S = 2$, $b_R = 2$, the filter's window size to $w_p = 16$ pixels, the filter's object thickness to $\theta = 0.5$ m, and train for 40,000 iterations. The link to the implementation is <https://github.com/alopezgit/project-adapt>.

To test our approach, data from a real LiDAR+RGB set-up is needed as we address the artifacts arising from projecting the LiDAR to the RGB camera. There are no standard real LiDAR+RGB indoor depth completion datasets available. In NYUv2 [61] the dense ground-truth is synthetically

sparsefied using Bernoulli sampling, while VOID [70] provides sparse depth from visual inertial odometry that contains no projection artifacts. Thus, the KITTI depth completion benchmark [65] is our real domain dataset, as it provides paired real noisy LiDAR depth with RGB images, along with denser depth ground-truth for testing. We use the official KITTI code to evaluate our method in the selected validation set and in the online test set, each containing 1,000 images. Following [66], we train using images of 1216×256 by cropping their top part. We evaluate on the full-resolution images of 1216×356 . The metrics used are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), reported in mm, and inverse RMSE (iRMSE) and inverse MAE (iMAE), in 1/km.

Synthetic Data We employ CARLA 0.84 [13] to generate synthetic data using the camera set-up in Fig. 2. We collect images from 154 episodes resulting in 18,022 multicamera images for training and 3,800 for validation. An episode is defined as an expert agent placed at random on the map and driving around while collecting left and right depth+RGB images, as well as the virtual LiDAR depth. We use for the virtual LiDAR camera a regular dense depth camera instead of the provided LiDAR sensor in CARLA because the objects in the LiDAR view are simplified (e.g., CARLA approximates the cars using cuboids). The resolution of the images is 1392×1392 with a Field Of View of 90° . To match the view and image resolution in KITTI, we first take a center crop of resolution 1216×356 , and from this crop we take the upper part of resolution 1216×256 . To adapt the synthetic RGB images, we train the original implementation of CycleGAN [82] for 180,000 iterations.

4.1 Ablation Study

We include an ablation study in Table 1 using the validation set. For the result of the whole pipeline, we average the results of three different runs to account for training variability. All of the proposed modules provide an increase in accuracy.

CARLA Adaptation Table 1 shows that projecting the sparse depth is as important as matching the LiDAR sampling pattern, decreasing the RMSE by 23.1% when used jointly. Table 1 also shows that using Bernoulli sampling and then projecting the sparse depth results in worse performance compared to training only with Bernoulli sampling. Compared to the LiDAR mask sampling case, where the see-through artifacts pattern is easy to distinguish after projecting the LiDAR points to the RGB camera (see Fig. 5), the lack of structure of a Bernoulli sampling approach may difficult the ability of the network to discern what points are see-through artifacts. Even though CycleGAN mostly adapts the brightness, contrast and colors of the images as shown in Fig. 6, using image translation further reduces the RMSE by

Table 1 Ablation study on the KITTI selected validation set

Model	RMSE	MAE	iRMSE	iMAE
<i>1st Step: Only Synthetic Supervision</i>				
Syn. Baseline 1: Bernoulli ($p_B=0.062$)	1735.59	392.81	7.68	1.73
+ Proj.	3617.98	1411.36	23.42	9.06
Syn. Baseline 2: LiDAR Mask	1608.32	386.49	7.13	1.76
+ Proj.	1335.00	342.16	5.41	1.55
+ Proj. + CycleGAN RGB + Only 1 mask	1273.77	316.22	5.26	1.37
+ Proj. + CycleGAN RGB	1247.53	308.08	4.54	1.34
<i>2nd Step: Adding Real Data</i>				
No Filter	1315.74	315.40	4.70	1.40
S_p + BerHu	1328.76	320.23	4.25	1.33
Full Pipeline: S_p	1150.27	281.94	3.84	1.20
Real GT + Synthetic GT	984.85	236.64	2.95	1.10
Real GT Supervision	802.49	214.04	2.24	0.91

BerHu refers to using BerHu for real data supervision. All of *2nd Step* results use *LiDAR Mask + Proj + CycleGAN RGB*. We use Bernoulli with $p_B=0.062$ as the KITTI LiDAR density for the crop used is approximately 6.2%. RMSE and MAE are reported in mm, and iRMSE and iMAE in 1/km. *Real GT Supervision* is the original result from [66] trained with an MSE loss, whereas *Real GT + Synthetic GT* refers to our result combining both CARLA and KITTI ground-truth and using a BerHu loss

Bolded results refer to the best performance among those methods not using the semi-dense KITTI ground-truth

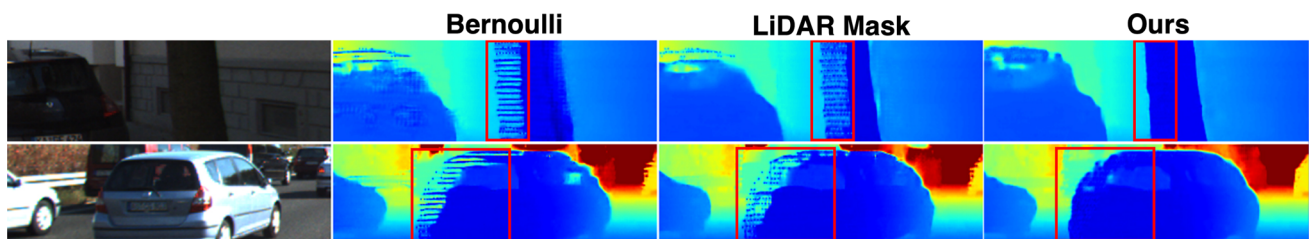


Fig. 7 Qualitative results with different training methodologies. *Bernoulli* refers to training using the Bernoulli sparsified depth x_D^B , *LiDAR Mask* to training using the depth sparsified with real LiDAR

masks x_D^M and *Ours* to our full pipeline that uses both LiDAR masks and projections. Both rows show projection artifacts, which *Ours* deals with correctly

6.6% when used jointly with real LiDAR masks sampling and projections. Figure 7 includes predictions for images with projection artifacts, showing that simulating the see-through artifacts via projections in the synthetic images is crucial to deal with the noisy input in the real domain.

Number of Sampled LiDAR Masks We now aim to test the effect on performance on the number of binary LiDAR masks used, as in real scenarios we may not have access to a large number of real LiDAR masks. To that end, we sampled a single mask at random from KITTI, which is applied to all the synthetic depth images during training. Table 1 shows the performance obtained for our method after the first step of training, i.e., using only synthetic data supervision, when using only one mask (+*Proj.* + *CycleGAN RGB* + *Only 1 mask*), where we see that the performance impact is minimal compared to using all available masks in KITTI (+*Proj.* + *CycleGAN RGB*).

Introducing Real Domain Data Using the reliable points S_p as supervision in the real domain alongside the MAE loss function increases the performance as Table 1 shows. If we use BerHu along with supervision with reliable points, the method deteriorates as the noisy points are likely to dominate the loss. Supervising our network using both MAE and the unfiltered real LiDAR also drops the performance due to the high noise rate η . These results show that using the noisy LiDAR points for supervision as in [47,69–71] can be detrimental to the performance. If we define a point to be noisy if its difference with the ground-truth is more than 0.3 m, the noise rate η for the unfiltered depth is 5.8%, and with our filtering method is reduced to 1.7% while dropping 45.8% of input points. The results suggest that η in S_p is more important than the total amount of points used for supervision. Supervising with real filtered data (*Full Pipeline*) improves both synthetic baselines (*Syn. Baseline*) in Table 1.

Table 2 Results on the KITTI validation set depending on the input type for the whole pipeline

Input data	RMSE	MAE	iRMSE	iMAE
Only Sparse Depth	1175.54	290.51	4.11	1.27
+ RGB	1167.83	289.86	3.87	1.28
+ T2Net transfer [81]	1184.39	306.66	3.99	1.36
+ FDM	1150.42	281.11	3.87	1.21
+ CycleGAN RGB	1150.27	281.94	3.84	1.20

RMSE and MAE are reported in mm, and iRMSE and iMAE in 1/km
 Bolded results refer to the best performance for each metric

Upper Bound Real GT Supervision in Table 1 refers to the results reported in [66], which trains only with KITTI ground-truth and using an MSE loss. *Real GT + Synthetic GT* refers to the results when also using KITTI ground-truth but mixing in the batch both CARLA and KITTI data with equal weighting in the loss, and BerHu for both KITTI and CARLA data. When having real data ground-truth, mixing synthetic and real data affects the attainable upper bound instead of giving a performance increase. However, our method can serve as a good pretraining step in the case where we only have access to a few labeled examples, which will be shown in a later section.

Impact of RGB Modality Contrary to self-supervised methods, which use RGB information to compute a photometric loss, we do not require the RGB image for good performance as shown in Table 2. Including RGB information reduces the error by 0.7% in RMSE, and by using the CycleGAN RGB images the RMSE is reduced by 2.1%. A similar low impact when adding the RGB modality was found in a past self-supervised work [47]. In a fully supervised manner the difference is 16.3% for FusionNet, showing that methods aiming to further reduce the RGB domain gap may increase the performance. Due to computational constraints, we train the CycleGAN model in a separate step. To test an end-to-end approach, we adapt T2Net [81], which, instead of using cycle consistency, relies on a depth estimation network to maintain geometric consistency after translation. However, our depth completion network is not capable of properly constraining the translation, and, hence, we obtained lower-quality translated images and larger error compared to CycleGAN as Table 2 shows.

Less Demanding RGB Transfer The changes performed by CycleGAN on the CARLA images are mostly related to contrast and saturation. Figure 8 shows that the CycleGAN translation also introduces artifacts similar to the chromatic aberration present in KITTI (e.g., road in the middle column images or the back of the van in the right column example), along with other distortions. Additionally, the results in Table 2 show that CycleGAN adaptation does not have a

large impact on our final performance. Thus, we propose to emulate the type of visual changes performed by CycleGAN with a less demanding non-adversarial approach, which does not produce the artifacts introduced by CycleGAN. To that end, we replace the CycleGAN module with the Feature Distribution Matching (FDM) approach proposed in [1], which does not require training and is less computationally expensive. FDM mostly adapts color and contrast by matching the mean and covariance of the synthetic image to a randomly sampled real image. Figure 8 shows examples of the resulting FDM images. In some cases, FDM translates the images to non-realistic colors, but we mostly observe that the translated images are similar to the CycleGAN images but without local artifacts. The performance when training our method with FDM images is given in Table 2, obtaining similar results to the CycleGAN approach.

4.2 Method Evaluation

Comparison to State-of-the-Art In Table 3 we compare our method, *Ours*, with the state-of-the-art methods not using real-domain ground-truth. Our method obtains a competitive performance compared to the GT-free state-of-the-art, which use video self-supervision. This performance is obtained using an architecture with fewer parameters, and without using any self-supervision. Tables 1 and 3 show that we achieve comparable results to most self-supervised methods by training only with synthetic data, i.e., in the first training step, which validates the observation that the main source of error to simulate to reduce the domain gap are the see-through points. Although both DDP [76] and ScaffNet [71] use synthetic ground-truth from Virtual KITTI [20] for training, they result in worse results compared to *Ours* due to not applying any domain adaptation strategy to their synthetic data. The recent work KBNNet [69], which does not use any synthetic data during training, obtains better performance than *Ours*, although KBNNet assumes knowledge of the intrinsic parameters at both training and testing time to design their model.

Adding Self-Supervision When real domain video data is available, our approach can be combined with self-supervised methods [47,70]. With that aim, we add to our pipeline the self-supervision loss from S2D [47]. In S2D, the relative pose between two video frames is assumed unknown, and thus needs to be estimated. To do so, we first obtain 2D matches of SIFT features [44] between the two RGB frames. Those matches from the first frame with a valid LiDAR depth are lifted to 3D to obtain 3D-2D matches between the first and second frame. We then use the 3D-2D matches to solve a Perspective-n-Point (PnP) problem with RANSAC, which estimates a relative pose between the two frames. Next, the estimated relative pose and the pre-

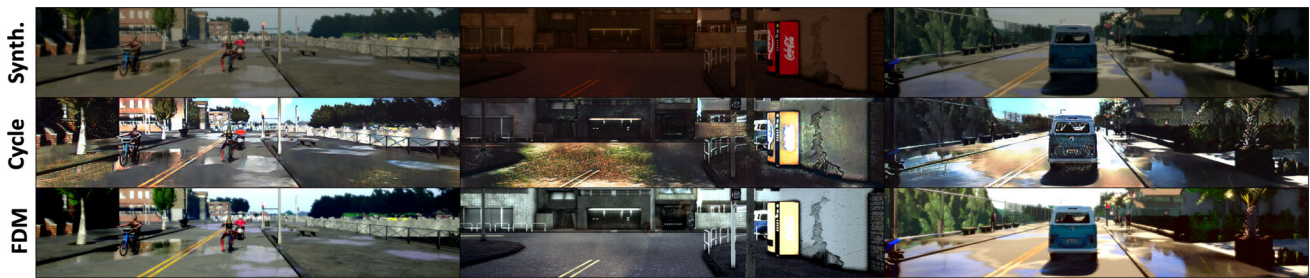


Fig. 8 Qualitative results of image transfer when applying to the original CARLA images (1st row) a CycleGAN approach (2nd row) or an FDM approach (3rd row)

dicted dense depth are used to project the RGB values of the second frame to the pose of the first frame using bilinear interpolation, which is differentiable. Finally, we compute a multi-scale photometric loss $\lambda_{ph}\mathcal{L}_{ph}$ by computing an l_1 -loss between the RGB values of the first frame and the projected second frame using the predicted dense depth and relative pose. Minimizing this loss guides the projected RGB values of the second frame to be similar to those of the first frame, which in turn guides the predicted depth to be closer

to the ground-truth. By using a multi-scale loss, we are able to improve the performance also in those cases where the predicted depth is far from the ground-truth value. We add this multi-scale photometric loss $\lambda_{ph}\mathcal{L}_{ph}$ during the second step of training only for the real KITTI data, where we set $\lambda_{ph} = 10$ to have similar loss values as $\mathcal{L}_S + \mathcal{L}_R$. This approach, which is shown in Table 3 in the line *Ours+SS*, further reduces the error in the test set and achieves lower errors compared to VOICED [70] and ScaffNet [71], although we

Table 3 Comparison of results on the KITTI selected validation set and the official online test set

Model	Param.	Validation set				Online test set			
		RMSE	MAE	iRMSE	iMAE	RMSE	MAE	iRMSE	iMAE
<i>Unsuperv.</i>									
DDP	18.8M	1325.79	355.86	–	–	1285.14	353.16	3.69	1.37
<i>Self-Sup.</i>									
SS-S2D	27.8M	1384.85	358.92	4.32	1.60	1299.85	350.32	4.07	1.57
DDP+St.	18.8M	1310.03	347.17	–	–	1263.19	343.46	3.58	1.32
VOICED	9.7M	1239.06	305.06	3.71	1.21	1169.97	299.41	3.56	1.20
ScaffNet	7.8M	1182.81	286.35	3.55	1.18	1121.93	280.76	3.30	1.15
KBNet	6.9M	1126.85	260.44	3.20	1.03	1069.47	256.76	2.95	1.02
<i>Dom. Ada.</i>									
DA Base	2.6M	1630.31	423.70	6.64	1.98	–	–	–	–
+ D. Out.	2.6M	1636.89	390.59	6.78	1.78	–	–	–	–
+ D. Feat.	2.6M	1617.41	389.88	7.01	1.79	–	–	–	–
Ours	2.6M	1150.27	281.94	3.84	1.20	1095.26	280.42	3.53	1.19
Ours-S2D	16.0M	1211.97	296.19	4.24	1.33	–	–	–	–
Ours+SS	2.6M	1112.83	268.79	3.27	1.12	1062.48	268.37	3.12	1.13
<i>Supervised</i>									
S-S2D	27.8M	878.56	260.90	3.25	1.34	814.73	249.95	2.80	1.21
FusionNet	2.6M	802.49	214.04	2.24	0.91	772.87	215.02	2.19	0.93
DDP	18.8M	–	–	–	–	836.00	205.40	2.12	0.86

DA Base is our Domain Adaptation baseline formed by CycleGAN [82] + LiDAR Masks. RMSE and MAE are reported in mm, and iRMSE and iMAE in 1/km. We compare against the supervised and self-supervised S2D [47] variants, the self-supervised VOICED [70]; the self-supervised ScaffNet [71]; the self-supervised KBNet [69]; the unsupervised, self-supervised and supervised variants of DDP [76]; and the supervised FusionNet [66]. The unsupervised DDP [76] refers to only using Virtual KITTI data without using any KITTI data. *DA Base* is our domain adaptation baseline, with a discriminator at the output (*D. Out*) or in the features (*D. Feat*). *Ours* uses FusionNet architecture, *Ours-S2D* uses S2D architecture and *Ours+SS* refers to our domain adaptation approach combined with the video self-supervision from [47]. Bolded results refer to the best performance among those methods not using the semi-dense KITTI ground-truth

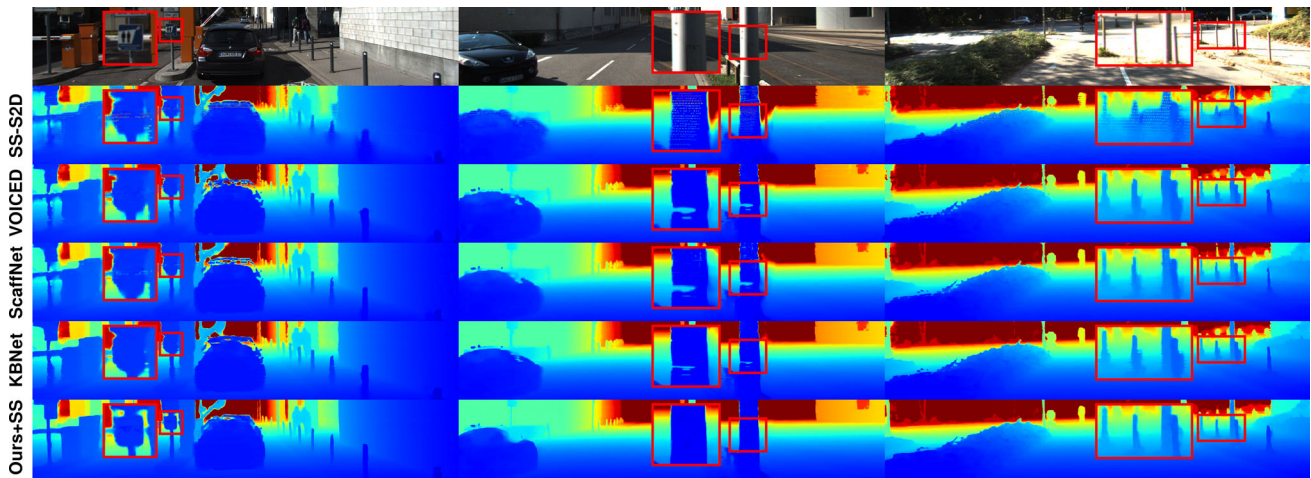


Fig. 9 Qualitative results on the official KITTI test set for SS-S2D, VOICED, ScaffNet, KBNNet and our method with video supervision *Ours+SS*. Red rectangles show zoomed-in areas (Color figure online)

Table 4 Semi-supervised results on the KITTI selected validation set for different pretraining strategies before finetuning on available annotations

Pretraining strategy	S:1 / I:196		S:3 / I:1508		S:5 / I:2690	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Only supervised	2578.72	1175.78	1177.90	302.30	1042.75	295.73
DA baseline	1130.79	310.68	1042.70	255.56	986.09	244.94
Ours	1106.30	262.29	996.28	247.00	949.63	242.61

S and *I* are the number of annotated sequences and images respectively. For *Only supervised*, the weights are randomly initialized. RMSE and MAE are reported in mm, and iRMSE and iMAE in 1/km. Bolded results refer to the best performance for each metric

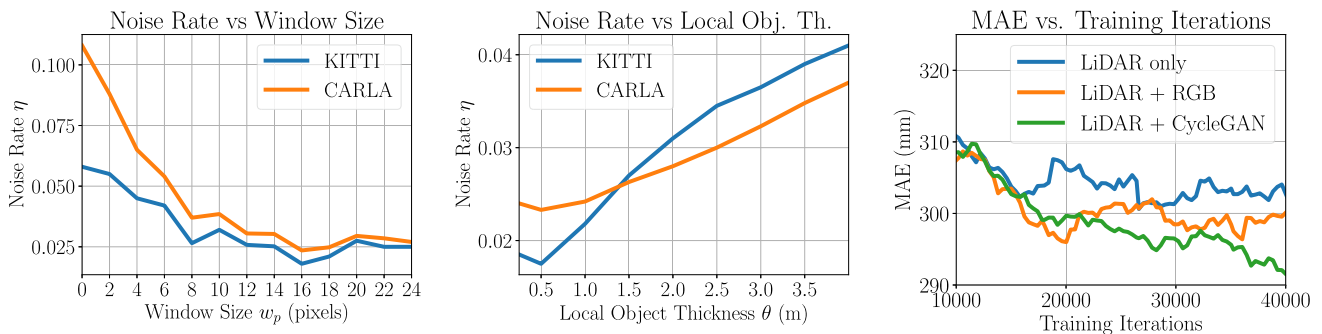


Fig. 10 Hyperparameter analysis. The two left images show the noise rate η versus w_p ($\theta = 0.5$ m) and θ ($w_p = 16$ pixels). The right plot shows MAE versus number of training iterations in the second step,

where we evaluate every 400 iterations, use a moving average with window size 25 and average 3 runs to reduce the variance

obtain comparable performance to the recent KBNNet [69]. However, the combination of our method with the architectural improvements in KBNNet, which assume knowledge of the camera intrinsics for training and testing, may further improve the results. We also show qualitative results of *Ours+SS*, KBNNet, ScaffNet, VOICED and SS-S2D in the official KITTI test set in Fig. 9. We see that the SS-S2D results include sparse see-through artifacts, e.g., the light pole in the middle column images, as SS-S2D uses the noisy depth input as supervision. KBNNet, ScaffNet and VOICED

handle these artifacts better, but the see-through depth points still affect the results in the light pole in the middle column example. Despite the similar quantitative performance of our method compared to the state-of-the-art self-supervised methods, our method shows qualitatively a reduction of these see-through points due to the use of adapted synthetic data that mimics these artifacts. Additionally, our method also shows better object boundaries compared to the other self-supervised approaches, which produce edges less sharp and wider than the original object. In general, self-supervision

presents issues with object boundaries [7,83], which can be alleviated by using accurate synthetic dense depth for supervision as done in our method.

Domain Adaptation Baselines Following synthetic-to-real depth estimation methods [2,42,81], we use as a domain adaptation baseline an image translation method, in our case CycleGAN [82], to adapt the synthetic RGB images. To sparsify the synthetic depth, we use the real LiDAR masks [3], shown in Table 1 to perform better than Bernoulli sampling. The performance of this domain adaptation baseline is presented in Table 3 in *DA Base*. We explore the use of adversarial approaches to match synthetic and real distributions on top of the *DA Base*. *DA Base + D. Out.* in Table 3 uses an output discriminator using the architecture in [52], with an adversarial loss weight of 0.001 similarly to [64]. Following [81], we also tested a feature discriminator in the model bottleneck in *DA Base + D. Feat.* with weight 0.01. Table 3 shows that the use of discriminators has a small performance impact and that standard domain adaptation pipelines are not capable of bridging the domain gap.

Semi-Supervised Learning In some settings, a subset of the real data may be annotated. Our full pipeline mimics the noise in the real sparse depth and takes advantage of the unannotated data by using the filtered reliable points S_p for supervision. This provides a good initialization for further finetuning with any available annotations as Table 4 shows. Compared to pretraining using the *DA Baseline*, our method achieves in all cases a better performance after finetuning.

Hyper-Parameter Selection We do not tune the loss weights λ_S and λ_R . The projected points x_D^p in the synthetic validation set are used to choose the filter window size w_p and the filter object thickness θ by employing the noise rate η in the reliable points S_p as the indicator for the filtering process performance. Figure 10 shows the noise percentage depending on w_p and θ , where we see that curves for the noise rate η follow a similar pattern in both the synthetic and real domain. We first select w_p and then θ as the gain in performance is lower for θ . The optimal values found are $w_p = 16$ pixels and $\theta = 0.5$ m. Figure 10 also shows the MAE depending on the number of iterations in the second step. After 40,000 training iterations, we did not see any improvement.

Sensitivity to Filter Parameters Figure 11 shows the RMSE when varying the window size w_p and the object thickness θ after 10,000 iterations of second step training. For values different to those used in our experiments, i.e., $\theta = 0.5$ m and $w_p = 16$ pixels, the performance after 40,000 iterations is decreased compared to the performance after 10,000 iterations and, furthermore, the performance after 40,000 iterations of training is also lower than the performance after only the first fully-synthetic step of training. We hypothesize that the reason for the drop in performance after 40,000 iterations for non-optimal filter parameters is the increased

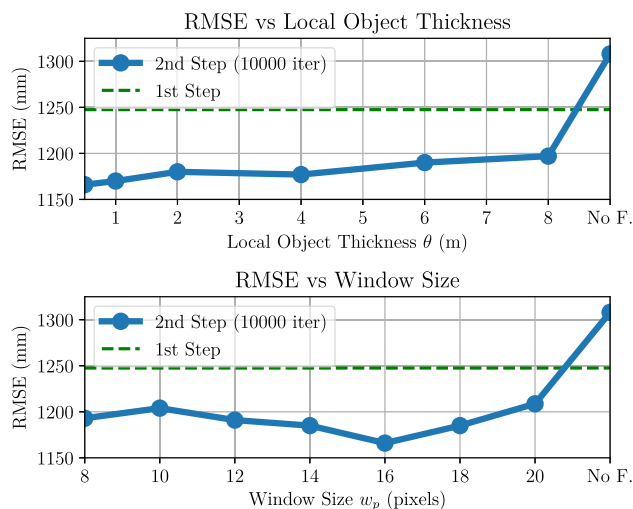


Fig. 11 RMSE in the second step of training when varying the parameters of the filter used to obtain the reliable points S_p . In the *top image* we vary the object thickness θ with window size set to $w_p = 16$ pixels. In the *bottom image* we vary the window size w_p , with object thickness fixed to $\theta = 0.5$ m. *1st Step* is the performance obtained after the first step of training, i.e., only using synthetic data. *No F.* refers to not using any filter in the real sparse depth used for supervision during the second step of training

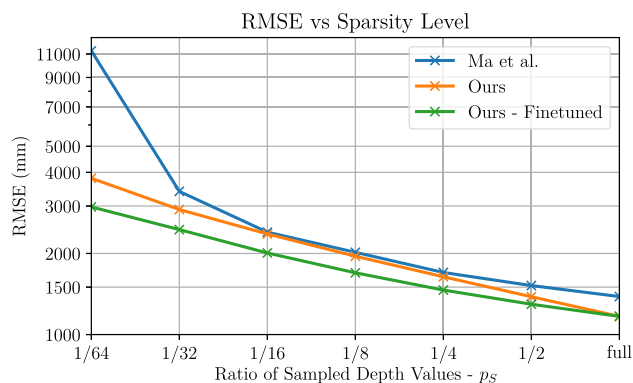


Fig. 12 RMSE obtained when varying the sparsity level by sampling the input sparse depth with probability p_s

noise rate η . Hence, a smaller number of training iterations is a safer choice for a wider range of filter parameters to avoid overfitting to the noisy points remaining after the filtering step.

Model Agnosticism We chose FusionNet [66] as our main architecture, but we also test our approach with the 18-layers architecture from [47] to show our method is robust to changes in architecture. Due to memory constraints we use the 18-layers architecture instead of the 34-layers model from [47], which accounts for the different parameter count in Table 3 between *Ours-S2D* and *SS-S2D*. We set the batch size to 2, increase the number of iterations in the second step to 90,000 (the last 20,000 iterations use a lower learning rate of 10^{-4}), and freeze the batch normalization statistics in the

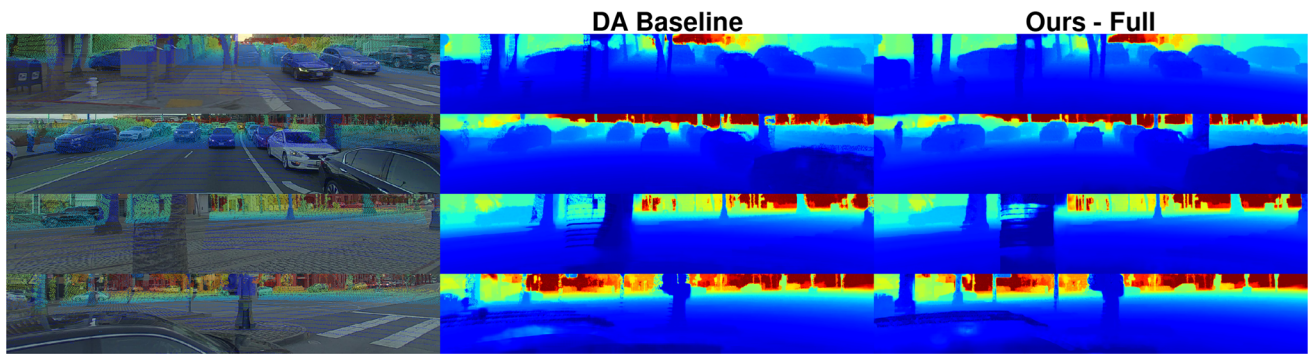


Fig. 13 Qualitative results on PandaSet [59] for our *DA Baseline* and full method (*Ours-Full*) trained in CARLA and KITTI. RGB images also show an overlay of the sparse depth input

second step. The result is given in Table 3 in *Ours w/S2D arch.*, which achieves state-of-the-art RMSE and MAE.

Sparsity of Input Data Similarly to [47], we evaluate the performance of our method when varying the input sparsity level. In this experiment, we keep each of the given input KITTI sparse depths with probability p_S . We evaluate both our final model trained with the original LiDAR maps, and also our final model finetuned for the different sparsity levels for 10,000 iterations with a learning rate of 10^{-4} . Figure 12 shows the results of the experiment when varying p_S . We show a similar trend to the results in [47], however our method is capable of performing better over all ranges of values. Finetuning the model increases the accuracy, but even without any further training the model behaves well for lower values of p_S .

Qualitative Results on PandaSet [59] are shown in Fig. 13 for our method and the *DA Baseline*, both without further tuning in PandaSet. PandaSet contains a different camera set-up and physical distances compared to the CARLA/KITTI training set-up, e.g., the top two rows in Fig. 13 correspond to a back camera not present in KITTI, whereas the two bottom rows are from a frontal-left camera. Our method is still capable of dealing better with the projection artifacts and with the missing data compared to the *DA Baseline*. However, there are regions that our method is not capable of correcting completely, such as the see-through artifacts on the tree trunk in the third row, which is due to a different camera set-up and depth statistics compared to our CARLA dataset. PandaSet does not provide depth completion ground-truth, thus no quantitative results can be computed.

Distribution of Errors Figure 14 shows that the distribution of ground-truth depth values is similar in CARLA and KITTI, as the synthetic data matches the camera position, intrinsics and crop used in KITTI, and also simulates similar scenes. Figure 14 also shows that, for both real and synthetic data, the MAE of the predictions increases for larger depth values, having both datasets have a minimum MAE for values

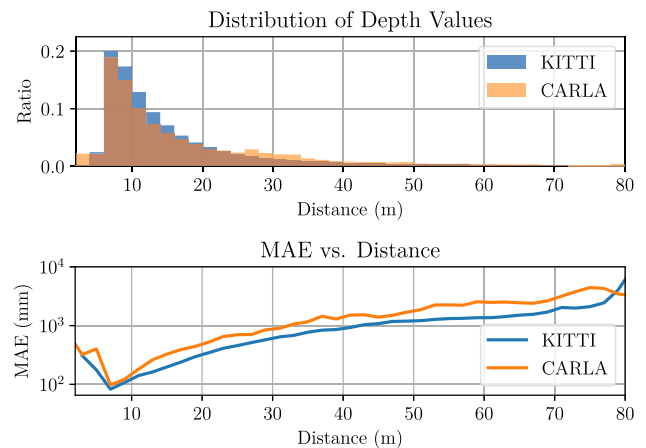


Fig. 14 Top: Distribution of depth ground-truth values in the KITTI validation set and the CARLA training set. Bottom: MAE versus ground-truth depth values in the KITTI validation set and the CARLA training set for our final model

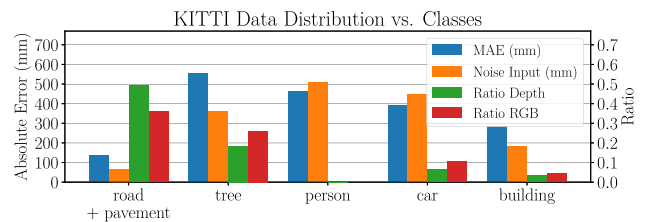


Fig. 15 Error and data distribution depending on the detected class (we use a Mask R-CNN [28] model pretrained on COCO [72]) on the KITTI selected validation set for our used crop of 1216×256 . *MAE* is the average absolute error for the predicted dense depth maps, *Noise Input* is the average absolute error in the input depth values, *Ratio Depth* is the ratio of the depth GT values available divided by the total amount of ground-truth values, *Ratio RGB* is the ratio of pixels representing a class divided by the total amount of pixels

around 6–8 m, where most of the depth values are concentrated. The error in the CARLA training set is consistently higher for most depth ranges, obtaining a total MAE of 500 mm, compared to the 282 mm obtained in the KITTI validation set. This higher error in the CARLA training data is a

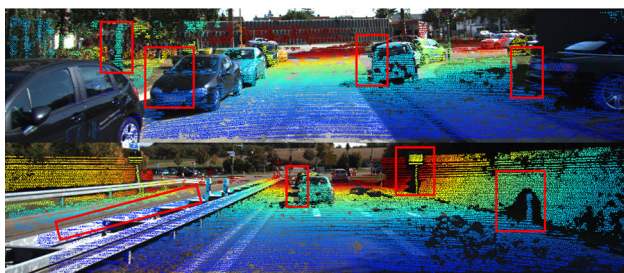


Fig. 16 KITTI ground-truth overlaid on the RGB image. The red rectangles show challenging areas where the ground-truth is missing, which is a consequence of the process to obtain the ground-truth performed in [65]

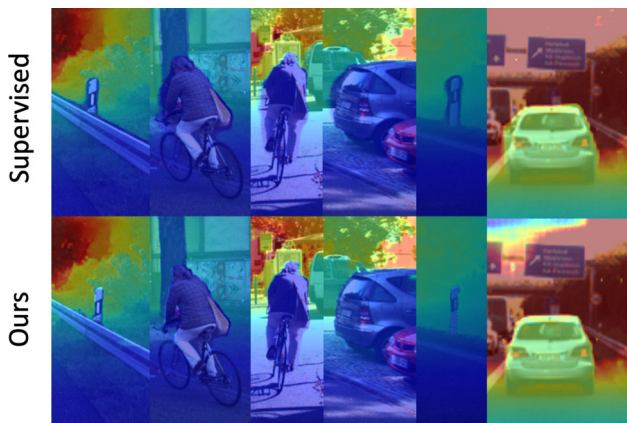


Fig. 17 Due to ground-truth misalignments, methods supervised with the KITTI ground-truth (top) tend to expand the predictions around the edges of the objects compared to our method (bottom), which is supervised with synthetic depth

result of the synthetic ground-truth being completely dense, whereas, due to the process used to generate the semi-dense maps, the KITTI ground-truth has a bias towards easier-to-predict depth values. To show this bias, in Fig. 15 we plot for different semantic classes in KITTI the MAE, the average input noise and the ratio that each class represents in the available ground-truth depth (*Ratio Depth*) and the RGB images (*Ratio RGB*). Figure 15 shows a relationship between the input noise and the prediction MAE, where classes with high input noise, e.g., *tree*, also present high output MAE. In KITTI, the class *road+pavement*, which has both the lowest output MAE and input noise, is given more weight in the depth ground-truth (*Ratio Depth*) compared to the actual ratio of RGB pixels that are from that class (*Ratio RGB*). Furthermore, Fig. 16 shows that the KITTI ground-truth has missing regions around the instances edges, which are areas of usually high prediction error and also contribute to the lower error in the KITTI validation set compared to the CARLA training set.

Ground-Truth Issues The process performed in [65] to obtain the KITTI semi-dense ground-truth is not completely noise-free, especially in dynamic or far-away objects. Sim-

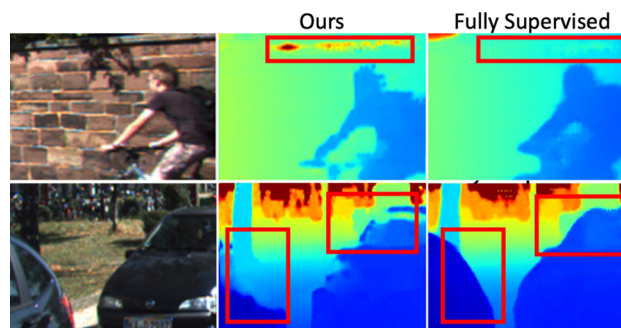


Fig. 18 Failure cases of our method in KITTI, which cannot correct all types of noise. The top row example shows a set of noisy inputs on the wall. The bottom row example shows a dropping of points due to low-reflectance black surfaces

ilarly, the ground-truth is not perfectly aligned around the edges of some objects, which, coupled with the missing depth around some instances shown in Fig. 16, makes the models trained with KITTI ground-truth to predict depth maps that expand the instances borders as shown in Fig. 17. Our model shows predictions better aligned around the boundaries, which highlights the benefits of training using pixel-perfect synthetic depth data.

Computational Complexity The inference speed depends on the architecture used as our method does not add any complexity. We use FusionNet, which has a low number of parameters compared to most of the state-of-art models as shown in Table 3. Furthermore, the computational time for different depth completion models in [29] also shows that FusionNet is fast compared to other models such as DeepLiDAR [54]. When using a 1080 Ti GPU and a batch of 10 images with a resolution of 1216×256 , FusionNet is capable of achieving a speed of 35 images/s. As for training, with our hardware set-up it takes ≈ 5.5 h for the first fully synthetic training step, and ≈ 4.5 h for the second training step, amounting to a total training time of ≈ 10 h. The given training time does not include the CycleGAN step, however as shown in Table 2 using the original RGB images or the lower-complexity FDM [1] method, which does not require any training, results in a similar performance to using the CycleGAN images.

Limitations While we addressed see-through artifacts, other types of noise can be present in the real sparse depth as Fig. 18 shows. The upper row example shows a set of noisy inputs on the wall that is not corrected. The bottom row example shows missing points in the prediction due to the lack of data on the black hood surface. The fully supervised model deals properly with these cases. These examples suggest that approaches focused on other types of noise could further decrease the error. Additionally, our method mimicked the KITTI set-up, e.g., camera positions and image resolution used. The performance of our method needs to be assessed

quantitatively in diverse set-ups. However, currently there is a lack of real LiDAR depth completion datasets that provide a ground-truth.

5 Conclusions

We proposed a domain adaptation method for sparse depth completion using data-driven masking and projections to imitate real noisy and sparse depth in synthetic data. The main source of noise in a joint RGB + LiDAR set-up was assumed to be the see-through artifacts due to projection from the LiDAR to the RGB reference frame. We also found a set of reliable points in the real data that are used for additional supervision, which helped to reduce the domain gap and to improve the performance of our model. A promising direction is to investigate the use of orthogonal domain adaptation techniques capable of leveraging the RGB inputs even more to correct also other types of error in the LiDAR co-modality.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abramov, A., Bayer, C., & Heller, C. (2020). *Keep it simple: Image statistics matching for domain adaptation*. arXiv preprint [arXiv:2005.12551](https://arxiv.org/abs/2005.12551).
- Atapour-Abarghouei, A., & Breckon, T. P. (2018). Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2800–2810).
- Atapour-Abarghouei, A., & Breckon, T. P. (2019). To complete or to estimate, that is the question: A multi-task approach to depth completion and monocular depth estimation. In *Proceedings of the international conference on 3D vision (3DV)* (pp. 183–193). IEEE.
- Barron, J. T., & Poole, B. (2016). The fast bilateral solver. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 617–632). Springer.
- Bartoccioni, F., Zablocki, É., Pérez, P., Cord, M., & Alahari, K. (2021). *Lidartouch: Monocular metric depth estimation with a few-beam lidar*. arXiv preprint [arXiv:2109.03569](https://arxiv.org/abs/2109.03569).
- Bergman, A. W., Lindell, D. B., & Wetzstein, G. (2020). Deep adaptive lidar: End-to-end optimization of sampling and depth completion at low sampling rates. In *2020 IEEE international conference on computational photography (ICCP)* (pp. 1–11). IEEE.
- Chen, X., Zhang, R., Jiang, J., Wang, Y., Li, G., & Li, T. H. (2022). *Self-supervised monocular depth estimation: Solving the edge-fattening problem*. arXiv preprint [arXiv:2210.00411](https://arxiv.org/abs/2210.00411).
- Chen, Y., Yang, B., Liang, M., & Urtasun, R. (2019). Learning joint 2d–3d representations for depth completion. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*.
- Cheng, X., Wang, P., & Yang, R. (2019). Learning depth with convolutional spatial propagation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(10), 2361–2379.
- Cheng, X., Zhong, Y., Dai, Y., Ji, P., Li, H. (2019b). Noise-aware unsupervised deep lidar-stereo fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 6339–6348).
- Cheng, X., Wang, P., Guan, C., & Yang, R. (2020). Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 10615–10622.
- Chodosh, N., Wang, C., & Lucey, S. (2018). Deep convolutional compressed sensing for lidar depth completion. In *Proceedings of the Asian conference on computer vision (ACCV)* (pp. 499–513). Springer.
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). Carla: An open urban driving simulator. In *Proceedings of the conference on robot Learning (CoRL)* (pp. 1–16).
- Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NeurIPS)* (pp. 2366–2374).
- Eldesokey, A., Felsberg, M., & Khan, F. S. (2019). Confidence propagation through cnns for guided sparse depth regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.
- Eldesokey, A., Felsberg, M., Holmquist, K., & Persson, M. (2020). Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 12014–12023).
- Engel, J., Koltun, V., & Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(3), 611–625.
- Faugeras, O. D., & Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 2(03), 485–508.
- Ferstl, D., Reinbacher, C., Ranftl, R., R  ther, M., & Bischof, H. (2013). Image guided depth upsampling using anisotropic total generalized variation. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 993–1000).
- Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.
- Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 3354–3361). IEEE.
- Godard, C., Mac, O., Gabriel, A., & Brostow, J. (2017). UCL_Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (p. 7). <https://doi.org/10.1109/CVPR.2017.699>. <http://visual.cs.ucl.ac.uk/pubs/monoDepth/>.
- Godard, C., Aodha, O. M., Firman, M., & Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *Pro-*

- ceedings of the *IEEE/CVF international conference on computer vision (ICCV)* (pp. 3828–3838).
24. Gofer, E., Praisler, S., & Gilboa, G. (2021). Adaptive lidar sampling and depth completion using ensemble variance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30, 8900–8912.
 25. Guo, X., Li, H., Yi, S., Ren, J., & Wang, X. (2018). Learning monocular depth by distilling cross-domain stereo networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 484–500). Springer.
 26. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., & Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems (NeurIPS)* (pp. 8527–8537).
 27. Handa, A., Whelan, T., McDonald, J., & Davison, A. J. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 1524–1531). IEEE.
 28. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 2961–2969).
 29. Hu, M., Wang, S., Li, B., Ning, S., Fan, L., & Gong, X. (2021). Towards precise and efficient image guided depth completion. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)*.
 30. Huang, T. S., & Netravali, A. N. (2002). Motion and structure from feature correspondences: A review. In *Advances in image processing and understanding* (pp. 331–347). World Scientific.
 31. Huang, Z., Fan, J., Yi, S., Wang, X., & Li, H. (2018). *Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion*. arXiv preprint [arXiv:1808.08685](https://arxiv.org/abs/1808.08685).
 32. Jaritz, M., De Charette, R., Wirbel, E., Perrotton, X., & Nashashibi, F. (2018). Sparse and dense data with cnns: Depth completion and semantic segmentation. In *Proceedings of the international conference on 3D vision (3DV)* (pp. 52–60). IEEE.
 33. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the international conference on learning representations (ICLR)*.
 34. Klodt, M., & Vedaldi, A. (2018). Supervising the new with the old: learning sfm from sfm. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 698–713). Springer.
 35. Ku, J., Harakeh, A., & Waslander, S. L. (2018). In defense of classical image processing: Fast depth completion on the cpu. In *Proceedings of the conference on computer and robot vision (CRV)* (pp. 16–22). IEEE.
 36. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., & Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *Proceedings of the international conference on 3D vision (3DV)* (pp. 239–248). IEEE.
 37. Lazaros, N., Sirakoulis, G. C., & Gasteratos, A. (2008). Review of stereo vision algorithms: From software to hardware. *International Journal of Optomechatronics*, 2(4), 435–462.
 38. Lee, B. U., Jeon, H. G., Im, S., & Kweon, I. S. (2019). Depth completion with deep geometry and context guidance. In *Proceedings of the international conference on robotics and automation (ICRA)* (pp. 3281–3287). IEEE.
 39. Li, J., Wong, Y., Zhao, Q., & Kankanhalli, M. S. (2019). Learning to learn from noisy labeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 5051–5059).
 40. Li, Z., & Snavely, N. (2018). MegaDepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2041–2050). <https://doi.org/10.1109/CVPR.2018.00218>. <http://arxiv.org/abs/1804.00607>.
 41. Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M. H., & Kautz, J. (2017). Learning affinity via spatial propagation networks. In *Advances in neural information processing systems (NeurIPS)* (pp. 1519–1529).
 42. Lopez-Rodriguez, A., & Mikolajczyk, K. (2020). Desc: Domain adaptation for depth estimation via semantic consistency. In *Proceedings of the British machine vision conference (BMVC)*.
 43. Lopez-Rodriguez, A., Busam, B., & Mikolajczyk, K. (2020). Project to adapt: Domain adaptation for depth completion from noisy and sparse sensor data. In *Proceedings of the Asian conference on computer vision (ACCV)*.
 44. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2), 91–110.
 45. Lutio, Rd., D’Aronco, S., Wegner, J. D., & Schindler, K. (2019). Guided super-resolution as pixel-to-pixel transformation. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 8829–8837).
 46. Ma, F., & Karaman, S. (2018). Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 1–8). IEEE.
 47. Ma, F., Cavalheiro, G. V., & Karaman, S. (2019). Self-supervised sparse-to-dense: self-supervised depth completion from lidar and monocular camera. In *2019 international conference on robotics and automation (ICRA)* (pp. 3288–3295). IEEE.
 48. Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W. C., & Urtasun, R. (2020). Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 11167–11176).
 49. Mur-Artal, R., & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5), 1255–1262.
 50. Park, J., Joo, K., Hu, Z., Liu, C. K., & Kweon, I. S. (2020). Non-local spatial propagation network for depth completion. In *Proceedings of the European conference on computer vision (ECCV), European conference on computer vision*.
 51. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *Advances in neural information processing systems (NeurIPS), Autodiff workshop*.
 52. Pilzer, A., Xu, D., Puscas, M., Ricci, E., & Sebe, N. (2018). Unsupervised adversarial depth estimation using cycled generative networks. In: Proceedings of the international conference on 3D vision (3DV) (pp. 587–595). IEEE.
 53. Poggi, M., Tosi, F., Mattoccia, S. (2018). Learning monocular depth estimation with unsupervised trinocular assumptions. In *Proceedings of the international conference on 3D vision (3DV)*.
 54. Qiu, J., Cui, Z., Zhang, Y., Zhang, X., Liu, S., Zeng, B., & Pollefeys, M. (2019). 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 (CVPR)* (pp. 3313–3322).
 55. Qu, C., Liu, W., & Taylor, C. J. (2021). Bayesian deep basis fitting for depth completion with uncertainty. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 16147–16157).
 56. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(3), 1623–1637.
 57. Riegler, G., Rütger, M., & Bischof, H. (2016). Atgv-net: Accurate depth super-resolution. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 268–284). Springer.

58. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 3234–3243).
59. Scale, A. I. (2020). Pandaset. <https://scale.com/open-datasets/pandaset>.
60. Scharstein, D., & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1–3), 7–42.
61. Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from rgb-d images. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 746–760). Springer.
62. Tang, K., Ramanathan, V., Fei-Fei, L., & Koller, D. (2012). Shifting weights: Adapting object detectors from image to video. In *Advances in neural information processing systems (NeurIPS)* (pp. 638–646).
63. Tippetts, B., Lee, D. J., Lillywhite, K., & Archibald, J. (2016). Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing (JRTIP)*, 11(1), 5–25.
64. Tsai, Y. H., Hung, W. C., Schuster, S., Sohn, K., Yang, M. H., & Chandraker, M. (2018). Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 7472–7481).
65. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., & Geiger, A. (2017). Sparsity invariant cnns. In *Proceedings of the international conference on 3D vision (3DV)* (pp. 11–20). IEEE.
66. Van Gansbeke, W., Neven, D., De Brabandere, B., & Van Gool, L. (2019). Sparse and noisy lidar completion with rgb guidance and uncertainty. In *International conference on machine vision applications (MVA)* (pp. 1–6). IEEE.
67. Voynov, O., Artemov, A., Egiazarian, V., Notchenko, A., Bobrovskikh, G., Burnaev, E., & Zorin, D. (2019). Perceptual deep depth super-resolution. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 5653–5663).
68. Watson, J., Firman, M., Brostow, G. J., & Turmukhambetov, D. (2019). Self-supervised monocular depth hints. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 2162–2171).
69. Wong, A., & Soatto, S. (2021). Unsupervised depth completion with calibrated backprojection layers. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 12747–12756).
70. Wong, A., Fei, X., Tsuei, S., & Soatto, S. (2020). Unsupervised depth completion from visual inertial odometry. *IEEE Robotics and Automation Letters*, 5(2), 1899–1906.
71. Wong, A., Cicek, S., & Soatto, S. (2021). Learning topology from synthetic data for unsupervised depth completion. *IEEE Robotics and Automation Letters*, 6(2), 1495–1502.
72. Wu, Y., Kirillov, A., Massa, F., Lo, W. Y., & Girshick, R. (2019). *Detectron2*. <https://github.com/facebookresearch/detectron2>.
73. Xiong, X., Xiong, H., Xian, K., Zhao, C., Cao, Z., & Li, X. (2020). Sparse-to-dense depth completion revisited: Sampling strategy and graph construction. In *Proceedings of the European conference on computer vision (ECCV)*.
74. Xu, Y., Zhu, X., Shi, J., Zhang, G., Bao, H., & Li, H. (2019). Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 2811–2820).
75. Yang, N., Wang, R., Stuckler, J., & Cremers, D. (2018). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 817–833). Springer.
76. Yang, Y., Wong, A., & Soatto, S. (2019). Dense depth posterior (ddp) from single image and sparse range. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 3353–3362).
77. Yoon, S., & Kim, A. (2020). Balanced depth completion between dense depth inference and sparse range measurements via kiss-gp. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 10468–10475). IEEE.
78. Yue, X., Wu, B., Seshia, S. A., Keutzer, K., & Sangiovanni-Vincentelli, A. L. (2018). A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM international conference on multimedia retrieval (ICMR)* (pp. 458–464). ACM.
79. Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems (NeurIPS)* (pp. 8778–8788).
80. Zhao, S., Fu, H., Gong, M., & Tao, D. (2019). Geometry-aware symmetric domain adaptation for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 9788–9798).
81. Zheng, C., Cham, T.J., & Cai, J. (2018). T2net: synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 767–783). Springer.
82. Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 2223–2232).
83. Zhu, S., Brazil, G., & Liu, X. (2020). The edge of depth: Explicit constraints between segmentation and depth. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 13116–13125).
84. Zou, Y., Yu, Z., Vijaya Kumar, B., & Wang, J. (2018). Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 289–305). Springer.
85. Zwald, L., & Lambert-Lacroix, S. (2012). The berhu penalty and the grouped effect. arXiv preprint [arXiv:1207.6868](https://arxiv.org/abs/1207.6868).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.