Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera

Citation: Ma, Fangchang, Guilherme Venturelli Cavalheiro and Sertac Karaman. "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera." Paper presented at the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20-24 May 2019, IEEE © 2019 The Author(s)

As Published: 10.1109/ICRA.2019.8793637

Publisher: IEEE

Persistent URL: https://hdl.handle.net/1721.1/126545

Version: Original manuscript: author's manuscript prior to formal peer review

Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera

Fangchang Ma, Guilherme Venturelli Cavalheiro, Sertac Karaman[*]
Massachusetts Institute of Technology
{fcma,guivenca,sertac}@mit.edu

Abstract: Depth completion, the technique of estimating a dense depth image from sparse depth measurements, has a variety of applications in robotics and autonomous driving. However, depth completion faces 3 main challenges: the irregularly spaced pattern in the sparse depth input, the difficulty in handling multiple sensor modalities (when color images are available), as well as the lack of dense, pixel-level ground truth depth labels. In this work, we address all these challenges. Specifically, we develop a deep regression model to learn a direct mapping from sparse depth (and color images) to dense depth. We also propose a self-supervised training framework that requires only sequences of color and sparse depth images, without the need for dense depth labels. Our experiments demonstrate that our network, when trained with semi-dense annotations, attains state-of-the-art accuracy and is the winning approach on the KITTI depth completion benchmark[2] at the time of submission. Furthermore, the self-supervised framework outperforms a number of existing solutions trained with semi-dense annotations.

Keywords: RGB-D Perception, Visual Learning, Sensor Fusion

1 Introduction

Depth sensing is fundamental in a variety of robotic tasks, including obstacle avoidance, 3D mapping [1, 2], and localization [3]. LiDAR, given its high accuracy and long sensing range, has been integrated into a large number of robots and autonomous vehicles. However, existing 3D LiDARs have a limited number of horizontal scan lines, and thus provide only sparse measurements, especially for distant objects (*e.g.*, the 64-line Velodyne scan in Figure 1 (a)). Furthermore, increasing the density of 3D LiDARs measurements is cost prohibitive[3]. Consequently, estimating dense depth from sparse measurements (*i.e.*, *depth completion*) is valuable for both academic research and large-scale industrial deployment.

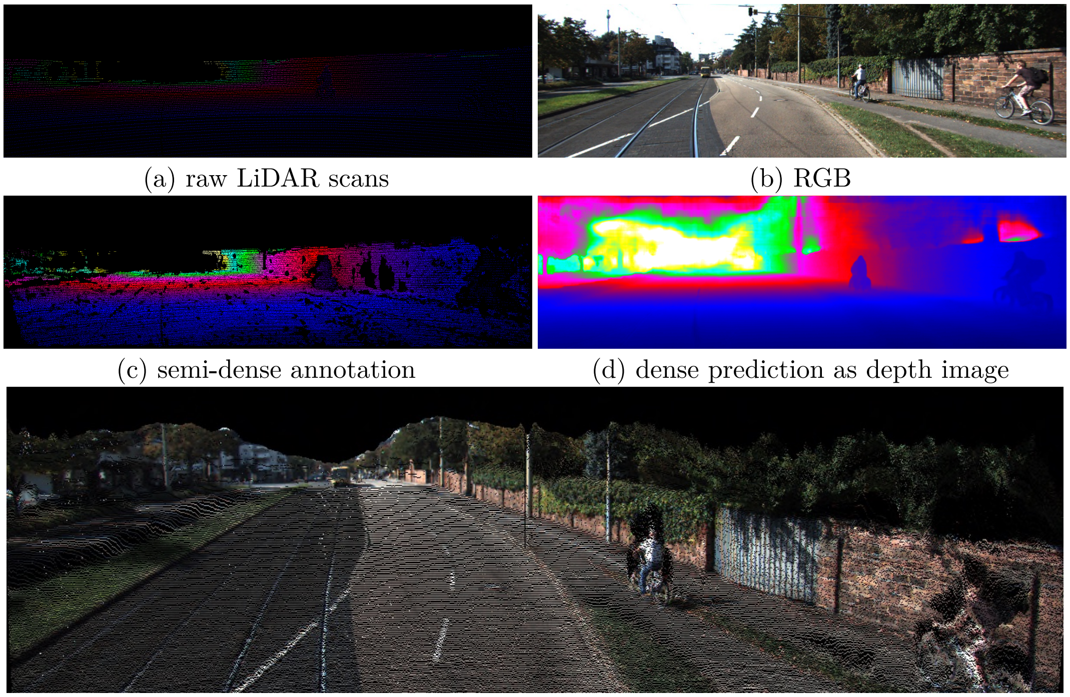
Depth completion from LiDAR measurements is challenging for several reasons. Firstly, the LiDAR measurements are highly sparse and also irregularly spaced in the image space. Secondly, it is a non-trivial task to improve prediction accuracy using the corresponding color image, if available, since depth and color are different sensor modalities. Thirdly, dense ground truth depth is generally not available, and obtaining pixel-level annotations can be both labor-intensive and non-scalable.

In this work, we address all these challenges with two contributions: *(1)* We develop a network architecture that is able to learn a direct mapping from the sparse depth (and color images, if available) to dense depth. This architecture achieves state-of-the-art accuracy on the KITTI Depth Completion Benchmark [4] and is currently the leading method. *(2)* We

[*] The authors are affiliated with the Department of Aeronautics and Astronautics (AeroAstro), and the Laboratory for Information & Decision Systems (LIDS), both at MIT.

[2] http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion

[3] Currently, the 16- and 64-line Velodyne LiDARs cost around $4k and $75k, respectively



(a) raw LiDAR scans

(b) RGB

(c) semi-dense annotation

(d) dense prediction as depth image

(e) dense prediction as point cloud

Figure 1: We develop a deep regressional network for *depth completion*: given *(a)* sparse LiDAR scans, and possibly *(b)* a color image, estimate *(d)* a dense depth image. Semi-dense depth labels, illustrated in *(d)* and *(e)*, are generally hard to acquire, so we develop a highly-scalable, self-supervised framework for training such networks. Best viewed in color.

propose a self-supervised framework for training depth completion networks. Our framework assumes a simple sensor setup with a sparse 3D LiDAR and a monocular color camera. The self-supervised framework trains a network without the need for dense labels, and outperforms some existing methods that are trained with semi-dense annotations. Our software[4] and demonstration video[5] will be made publicly available.

2 Related Work

Depth completion. *Depth completion* is an umbrella term that covers a collection of related problems with a variety of different input modalities (*e.g.*, relatively dense depth input [5, 6, 7] vs. sparse depth measurements [8, 9]; with color images for guidance [6, 10] vs. without [4]). The problems and solutions are usually sensor-dependent, and as a result they face vastly different levels of algorithmic challenges.

For instance, depth completion for structured light sensor (*e.g.*, Microsoft Kinect) [11] is sometimes also referred to as *depth inpainting* [12], or *depth enhancement* [5, 6, 7] when noise is taken into account. The task is to fill in small missing holes in the relatively dense depth images. This problem is relatively easy, since most pixels (typically over 80%) are observed. Consequently, even simple filtering-based methods [5] can provide good results. As a side note, the inpainting problem also finds close connection to *depth denoising* [13] and *depth super-resolution* [14, 15, 16, 17, 18, 19].

However, the completion problem becomes much more challenging when the input depth image has much lower density, because the inverse problem is ill-posed. For instance, Ma et al. [8, 9] addressed depth reconstruction from only hundreds of depth measurements, by assuming a strong *a priori* of piecewise linearity in depth signals. Another example is autonomous

[4] https://github.com/fangchangma/self-supervised-depth-completion

[5] https://youtu.be/bGXfvF261pc

driving with 3D LiDARs, where the projected depth measurements on the camera image space account for roughly 4% pixels [4]. This problem has attracted a significant amount of recent interest. Specifically, Ma and Karaman [10] proposed an end-to-end deep regression model for depth completion. Ku et al. [20] developed a simple and fast interpolation-based algorithm that runs on CPUs. Uhrig et al. [4] proposed *sparse convolution*, a variant of regular convolution operations with input normalizations, to address data sparsity in neural networks. Eldesokey et al. [21] improved the normalized convolution for confidence propagation. Chodosh et al. [22] incorporated the traditional dictionary learning with deep learning into a single framework for depth completion. Compared with all these prior work, our method achieves significantly higher accuracy.

Depth prediction. Depth completion is closely related to depth prediction from a monocular color image. Research in depth prediction dates further back to early work by Saxena et al. [23]. Since then, depth prediction has evolved from simple handcrafted feature representations [23] to the deep learning based approaches [24, 25, 26, 27] (see the reference therein). Most learning-based work relied on pixel-level ground truth depth training. However, ground truth depth is generally not available and cannot be manually annotated. To address such difficulties, recent focus has shifted towards seeking other supervision signals for training. For instance, Zhou et al. [28] developed an unsupervised learning framework for simultaneous estimation of depth and ego-motion from a monocular camera, using photometric loss as a supervision. However, the depth estimation is only up-to-scale. Mahjourian et al. [29] improved the accuracy by using 3D geometric constraints, and Yin and Shi [30] extended the framework for optical flow estimation. Li et al. [31] recovered the absolute scale by using stereo image pairs. In contrast, in this work we propose the first self-supervised framework that is designed specifically for depth completion. We utilize the `RGBd` sensor data and the well-studied, traditional model-based methods for pose estimation, in order to provide absolute-scale depth supervision.

3 Network Architecture

We formulate the depth completion problem as a deep regression learning problem. For ease of notation, we use `d` for sparse depth input (pixels without measured depth are set to zero), `RGB` for color images (or grayscale images), and `pred` for depth prediction.

The proposed network follows an encoder-decoder paradigm [32], as displayed in Figure 2. The encoder consists of a sequence of convolutions with increasing filter banks to downsample the feature spatial resolutions. The decoder, on the other hand, has a reversed structure with transposed convolutions to upsample the spatial resolutions.

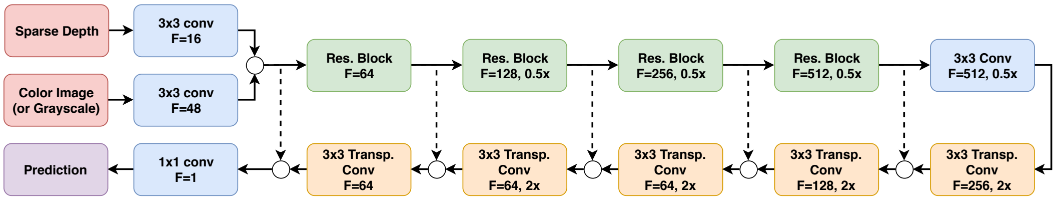


Figure 2: Our deep regression network for depth completion, with both sparse depth and RGB as input. Skip connections are denoted by dashed lines and circles represent concatenation of channels.

The input sparse depth and the color image, when available, are separately processed by their initial convolutions. The convolved outputs are concatenated into a single tensor, which acts as input to the residual blocks of ResNet-34 [33]. Output from each of the encoding layers is passed to, via skip connections, the corresponding decoding layers. A final 1x1 convolution filter produces a single prediction image with the same resolution as network input. All convolutions are followed by batch normalization [34] and ReLU, with the exception at the last layer. At inference time, predictions below a user-defined threshold τ are clipped to τ. We empirically set $\tau = 0.9m$, the minimal valid sensing distance for LiDARs.

In the absence of color images, we simply remove the RGB branch and adopt a slightly different set of hyper parameters: the number of filters is reduced to half (*e.g.*, the first residual block has 32 channels, instead of 64).

4 Self-supervised Training Framework

Existing work on depth completion relies on densely annotated ground truth for training. However, dense ground truth generally does not exist, and even the acquisition of semi-dense labels can be technically challenging. For instance, Uhrig et al. [4] created an annotated depth dataset by aggregating consecutive data frames using GPS, stereo vision, and additional manual inspection. However, this method is not easily scalable. Furthermore, it produces only semi-dense annotations ($\sim 30\%$ pixels) within the bottom half of the image.

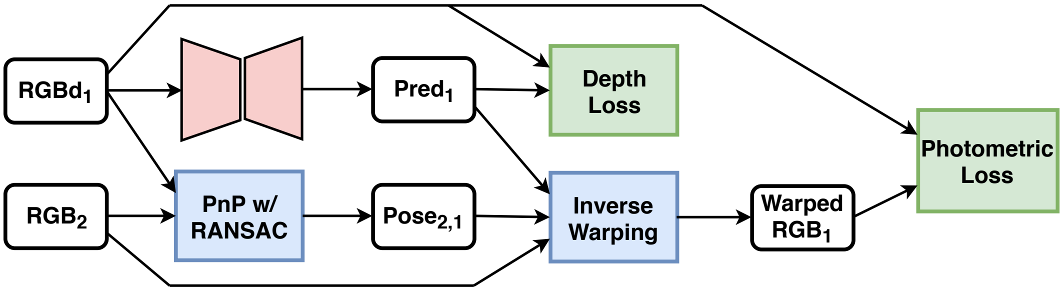


Figure 3: An illustration of the self-supervised training framework, which requires only a sequence of color images and sparse depth images. White rectangles are variables, red is the depth network to be trained, blue are deterministic computational blocks (without learnable parameters), and green are loss functions.

In this section, we propose a model-based self-supervised training framework for depth completion. This framework requires only a synchronized sequence of color/intensity images from a monocular camera and sparse depth images from LiDAR. Consequently, the self-supervised framework does not rely on any additional sensors, manual labeling work, or other learning-based algorithms as building blocks. Furthermore, this framework does not depend on any particular choice of neural network architectures. The self-supervised framework is illustrated in Figure 3. During training, the current data frame $\texttt{RGBd}_1$ and a nearby data frame $\texttt{RGB}_2$ are both used to provide supervision signals. However, at inference time, only the current frame $\texttt{RGBd}_1$ is needed as input to produce a depth prediction $\texttt{pred}_1$.

Sparse Depth Supervision The sparse depth input $\texttt{d}_1$ itself can be used as a supervision signal. Specifically, we penalize the differences between network input and output on the set of pixels with known sparse depth, and thus encouraging an identity mapping on this set. This loss leads to higher accuracy, improved stability and faster convergence for training. The depth loss is defined as

$$\mathcal{L}_{\text{depth}}(\texttt{pred}, \texttt{d}) = \left\| \mathbb{1}_{\{\texttt{d}>0\}} \cdot (\texttt{pred} - \texttt{d}) \right\|_2^2 . \tag{1}$$

Note that a denser ground truth (*e.g.*, the 30% dense annotation from the KITTI depth completion benchmark [4]), if available, can also be used in place of the sparse input $\texttt{d}_1$.

Model-based Pose Estimation As an intermediate step towards the photometric loss, the relative pose between the current frame and the nearby frame needs to be computed. Prior work assumes either known transformations (*e.g.*, stereo [31]) or the use of another learned neural network for pose estimation (*e.g.*, [28]). In contrast, in this framework, we adopt a model-based approach for pose estimation, utilizing both $\texttt{RGB}$ and $\texttt{d}$.

Specifically, we solve the Perspective-n-Point (PnP) problem [35] to estimate the relative transformation $T_{1\to 2}$ between the current frame 1 and the nearby frame 2, using matched feature correspondences extracted from $\texttt{RGBd}_1$ and $\texttt{RGB}_2$ respectively. Random sample consensus (RANSAC) [36] is also adopted in conjunction with PnP to improve robustness to outliers in feature matching. Compared to `RGB`-based estimation [28] which is up-to-scale, our estimation is scale-accurate and failure-aware (flag returned if no estimation is found).

Photometric Loss as Depth Supervision Given the relative transformation $T_{1\to 2}$ and the current depth prediction $\texttt{pred}_1$, the nearby color image $\texttt{RGB}_2$ can be inversely warped to the current frame. Specifically, given the camera intrinsic matrix K, any pixel p_1 in the current frame 1 has the corresponding projection in frame 2 as $p_2 = KT_{1\to 2}\texttt{pred}_1(p_1)K^{-1}p_1$. Consequently, we can create a synthetic color image using bilinear interpolation around the 4 immediate neighbors of p_2. In other words, for all pixels p_1:

$$\texttt{warped}_1(p_1) = \text{bilinear}(\texttt{RGB}_2(KT_{1\to 2}\texttt{pred}_1(p_1)K^{-1}p_1)). \tag{2}$$

`warped` is similar to the current $\texttt{RGB}_1$ when the environment is static and there's limited occlusion due to change of view point. Note that this photometric loss is made differentiable by the bilinear interpolation. Minimizing the photometric error reduces the depth prediction error, only when the depth prediction is close enough to the ground truth (*i.e.*, when the projected point p_2 differs from the true correspondence by no more than 1 pixel). Therefore, a multi-scale strategy is applied to ensure $\left\|p_2^{(s)} - p_1^{(s)}\right\|_1 < 1$ on at least one scale s. In additional, to avoid conflicts with the depth loss, the photometric loss is evaluated only on pixels without direct depth supervision. The final photometric loss is

$$\mathcal{L}_{\text{photometric}}(\texttt{warped}_1, \texttt{RGB}_2) = \sum_{s\in S} \frac{1}{s} \left\| \mathbb{1}^{(s)}_{\{\texttt{d==0}\}} \cdot (\texttt{warped}_1^{(s)} - \texttt{RGB}_2^{(s)}) \right\|_1 , \tag{3}$$

where S is the set of all scaling factors, and $(\cdot)^{(s)}$ represents image resizing (with average pooling) by a factor of s. Losses at lower resolutions are weighted down by s.

Smoothness Loss The photometric loss only measures the sum of all individual errors (*i.e.*, color differences computed on each pixel independently) without any neighboring constraints. Consequently, minimizing the photometric loss alone usually results in an undesirable local optimum, where the depth pixels have incorrect values (despite having a low photometric error) and high discontinuity. To alleviate this issue, we add a third term to the loss functions in order to encourage smoothness of the depth predictions. Inspired by [9, 8, 28], we penalize $\left\|\nabla^2 \texttt{pred}_1\right\|_1$, the $\mathcal{L}_1$ loss of the second-order derivatives of the depth predictions, to encourage piecewise-linear depth signal.

In summary, the final loss function for the entire self-supervised framework consists of 3 terms:

$$\mathcal{L}_{\text{self}} = \mathcal{L}_{\text{depth}}(\texttt{pred}_1, \texttt{d}_1) + \beta_1\ \mathcal{L}_{\text{photometric}}(\texttt{warped}_1, \texttt{RGB}_1) + \beta_2\ \left\|\nabla^2 \texttt{pred}_1\right\|_1 \tag{4}$$

where β_1, β_2 are relative weightings. Empirically we set $\beta_1 = 0.1$ and $\beta_2 = 0.1$.

5 Implementation

For the sake of benchmarking against state-of-the-art methods, we use the KITTI depth completion dataset [4] for both training and testing. The dataset is created by aggregating LiDAR scans from 11 consecutive frames into one, producing a semi-dense ground truth with roughly 30% annotated pixels. The dataset consists of 85,898 training data, 1,000 selected validation data, and 1,000 test data without ground truth.

For the PnP pose estimation, we dialate the sparse depth images $\texttt{d}_1$ with a 4×4 kernel, since the extracted features points might not have spot-on depth measurements. In each epoch, we iterate through the entire training dataset for the current frame 1, and choose a neighbor frame 2 randomly from the 6 nearest frames in time (excluding the current frame itself). In presence of PnP pose estimation failure, $T_{1\to 2}$ is set to be an identity matrix and the neighbor $\texttt{RGB}_2$ image is overwritten by the current $\texttt{RGB}_1$. Consequently, the photometric loss is made to be 0, and does not affect the training.

The training framework is implemented in PyTorch [37]. Zero-mean Gaussian random initialization is used for the network weights. We use a batch size of 8 for the `RGBd`-network, and 16 for the simpler d-network. Adam with a starting learning rate of 10^{-5} is used for network optimization. The learning rate is reduced to half every 5 epochs. We use 8 Tesla V100 GPUs with 16G of RAM for training, and 12 epochs takes roughly 12 hours for the RGBd-network and 4 hours for the d-network.

6 Results

In this section, we present experimental results to demonstrate the performance of our approach. We first compare our network architecture, trained in a purely supervised fashion, against state-of-the-art published methods. Secondly, we conduct an ablation study on the proposed network architecture to gain insight into which components contribute to the prediction accuracy. Lastly, we showcase training results using our self-supervised framework, and present an empirical study on how the algorithm performs under different level of sparsity in the input depth signals.

6.1 Comparison with State-of-the-art Methods

In this section, we train our best network in a purely supervised fashion to benchmark against other published results. We use the official error metrics for the KITTI depth completion benchmark [4], including rmse, mae, irmse, and imae. Specifically, rmse and mae stand for the root-mean-square error and the mean absolute error, respectively; irmse and imae stand for the root-mean-square error and the mean absolute error in the inverse depth representation. The results are listed in Table 1 and visualized in Figure 4.

Table 1: Comparison against state-of-the-art algorithms on the test set.

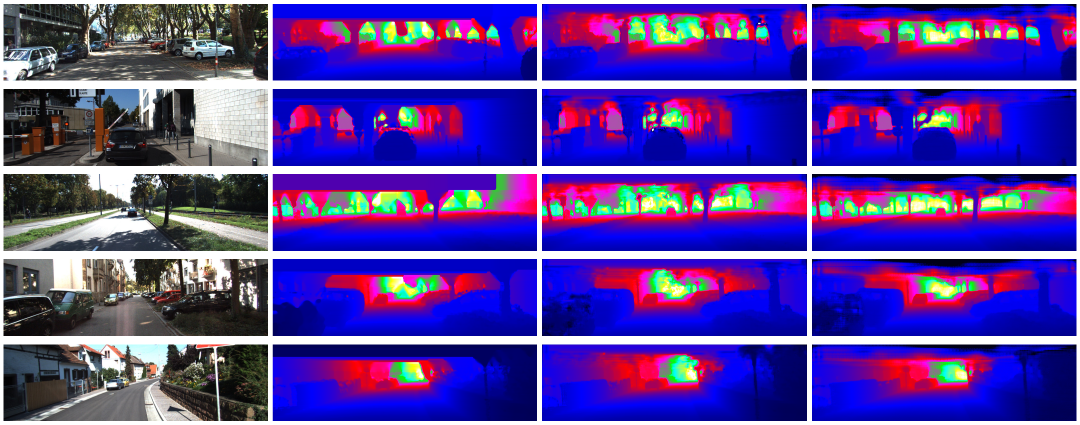
Method	Input	rmse [mm]	mae [mm]	irmse [1/km]	imae [1/km]
NadarayaW [4]	d	1852.60	416.77	6.34	1.84
SparseConvs [4]	d	1601.33	481.27	4.94	1.78
ADNN [22]	d	1325.37	439.48	59.39	3.19
IP-Basic [20]	d	1288.46	302.60	3.78	**1.29**
NConv-CNN [21]	d	1268.22	360.28	4.67	1.52
NN+CNN2 [4]	d	1208.87	317.76	12.80	1.43
Ours-d	d	**954.36**	**288.64**	**3.21**	1.35
SGDU [18]	RGBd	2312.57	605.47	7.38	2.05
Ours-RGBd	RGBd	**814.73**	**249.95**	**2.80**	**1.21**

Our d-network leads prior work with a large margin in almost all metrics. The RGBd-network attains even higher accuracy, leading all submissions to the benchmark. Our predicted depth images also have cleaner and sharper object boundaries (*e.g.*, see trees, cars and road signs), which can be attributed to the fact that our network is quite deep (and thus might be able to learn more complex semantic representations) and has large skip connections (and thus preserves image details). Note that all these supervised methods produce poor predictions at the top of the image, because of 2 reasons: (a) the LiDAR returns no measurements, and thus the input to the network is all zero at the top; (b) the 30% semi-dense annotations do not contain labels in these top regions.

6.2 Ablation Studies

To examine the impact of network components on performance, we conduct a systematic ablation study and list the results column-wise in Table 2.

The most effective components in improving final accuracy includes using RGBd for input and $\mathcal{L}_2$ loss for training. This is in contrary to the findings that $\mathcal{L}_1$ is more effective [10, 38], implying that the optimal loss functions might be dataset- and architecture-dependent. Adding skip connections, training from scratch (without ImageNet-pretraining), and not using max pooling also result in substantial improvement. Increasing network depth (from 18 to 34) and encoders-decoders pairs (from 3 to 5), as well as a proper split of filters allocated to the RGB and the d branches (16/48 split), also create small positive impact on the results.



(a) RGB (b) NConv-CNN [21] (c) NN+CNN2 [4] (d) Ours-`d`

Figure 4: Comparision against other methods (best viewed in color). Our predictions have not only lower errors, but also cleaner and sharper boundaries.

Table 2: Ablation study of the network architecture for depth input. Empty cells indicate the same value as the first row of each section. See Section 6.2 for detailed discussion.

image	fusion split	loss	ResNet depth	with skip	reduced filters	pre-trained	№ pairs	down-sample	dropout & weight decay	`rmse` [mm]
None	-	L_2	34	Yes	2x ($F_1=32$)	No	5	No	No	**991.35**
		L_1								1170.58
			18							1003.78
				No						1060.64
					1x ($F_1=64$)					992.663
					1x ($F_1=64$)	Yes				1058.218
					4x ($F_1=16$)					1015.204
							4			996.024
							3			1005.935
								Yes		1045.062
									Yes	1002.431
Gray	16/48	L_2	34	Yes	1x ($F_1=64$)	No	5	No	Yes	**856.754**
RGB										859.528
	32/32									868.969
			18							875.477
				No						1070.789
	8/24				2x ($F_1=32$)					887.472
							4			857.154
							3			857.448
								Yes		859.528

However, additional regularization, including dropout combined with a weight decay, leads to degraded performance.

It is worth noting that alternative encoding of the input depth image (such as the nearest neighbor interpolation or the bilinear interpolation of the sparse depth measurements) does not improve the prediction accuracy. This implies that the proposed network is able to deal with highly sparse input image.

6.3 Evaluation of the Self-supervised Framework

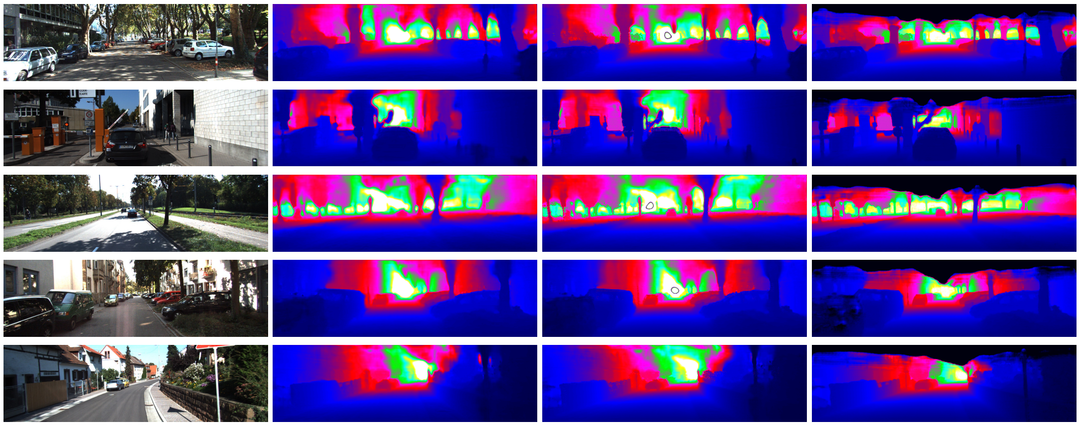
In this section, we evaluate the self-supervised training framework described in Section 4 on the KITTI validation dataset. We compare 3 different training methods: using only photometric loss without sparse depth supervision, the complete self-supervised framework (*i.e.*, photometric loss with sparse depth supervision), and the pure supervised method using the semi-dense annotations. The quantitative results are listed in Table 3. The self-supervised result produces `rmse` = 1384, which already outperforms some of the prior methods that were trained with semi-dense annotations, such as SparseConvs [4].

Table 3: Evaluation of the self-supervised framework on the validation set

Training Method	`rmse` [mm]	`mae` [mm]	`irmse` [1/km]	`imae` [1/km]
Photometric Loss Only	1901.16	658.13	5.85	2.62
Self-Supervised	1384.85	358.92	4.32	1.60
Supervised Learning	878.56	260.90	3.25	1.34

However, note that the true quality of depth predictions trained in a self-supervised fashion is probably underestimated by such evaluation metrics, since the "ground truth" itself is biased. Specifically, the evaluation ground truth is characterized by the same limitations as the training annotations: low-density, as well as absence at the top region. As a result, predictions at the top, where the self-supervised framework provides supervision but semi-dense annotations do not, are not reflected in the error metrics, as illustrated in Figure 5.

The self-supervised framework is effective for not only 64-line lidar measurements, but also lower-resolution lidars and more sparse depth input. In Figure 6(b), we show the validation errors of the networks trained with the self-supervised framework with different levels of sparsity in the depth. When the number of input measurements is too small, the validation error is high. This is expected due to failure in PnP pose estimation. However, with sufficiently many measurements (*e.g.*, at least 4 scanlines, or the equivalent number of samples to at least 2 scanlines when input is uniformaly sampled), the validation error starts to decrease as a power function of the input, similar to training with semi-dense annotations.

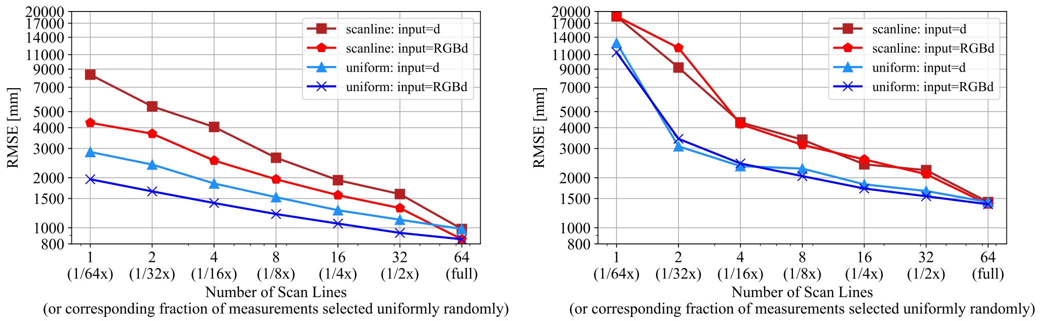


(a)RGB (b)Photometric Only (c)Self-supervised (d)Supervised

Figure 5: Comparision between different training methods (best viewed in color). The photometric loss provides supervision at the top, where the semi-dense annotation does not contain labels.

6.4 On Input Sparsity

In many robotic applications, engineers need to address the following question: *what's the LiDAR resolution (which translates to financial cost) required to achieve certain performance?* In this section, we try to answer this question by evaluating the accuracy of our LiDAR depth completion technique under different input sparsity and spatial patterns. To this end, we provide an empirical analysis on the depth completion accuracy for different depth input with varying levels of sparsity and spatial patterns. In particular, we downsample the raw LiDAR input in two different manners: reducing the number of laser scans (to simulate a LiDAR with fewer scan lines), and uniformly sub-sampling from all LiDAR measurements available. The results are illustrated in Figure 6, for both of these spatial patterns and both input modalities of d and RGBd.



(a) trained with semi-dense annotations

(b) self-supervised

Figure 6: Prediction error against number of input depth samples, for both spatial patterns (uniform random sub-sampling and LiDAR scan lines). (a) When trained with semi-dense ground truth, the depth completion error decreases as a power function cx^p of the number of input depth measurements, for some $c > 0, p < 0$. (b) The self-supervised framework is effective with sufficiently many measurements (at least 4 scanlines, or the equivalent number of samples to 2 scanlines when input is uniformaly sampled).

In Figure 6(a) we show the validation errors when trained with semi-dense annotations. The `rmse` errors form a straight line in the log-log plot, implying that the depth completion error decreases as a power function cx^p of the number of input depth measurements, for some positive c and negative p. This also implies diminishing returns on increasing LiDAR resolutions. Comparing the two spatial patterns, uniform random sub-sampling produces significantly higher accuracy than having a reduced number of scan lines, since the input depth samples are more disperse in the pixel space with uniform random sampling. Furthermore, using `RGBd` substantially reduces prediction error, compared to using only `d`, when trained with semi-dense annotations. The performance gap is especially significant when the number of depth measurements is low. Note that there is a significant drop of RMSE from 32-line to 64-line LiDAR. This accuracy gain may be attributed to the fact that our network architecture is optimized for 64-line LiDAR.

In Figure 6(b), we show results when trained with our self-supervised framework. As has been discussed in Section 6.3, the validation error starts to decrease steadily as a power function, similar to training with semi-dense annotations, when there are sufficiently many input measurements. However, with the self-supervised framework, using both RGB and sparse depth yields the same level of accuracy as using sparse depth only, which is different from training with semi-dense annotations. The underlying cause of this difference remains to be further investigated[6].

7 Conclusions

In this paper, we have developed a deep regression model for depth completion of sparse LiDAR measurements. Our model achieves state-of-the-art performance on the KITTI depth completion benchmark, and outperforms existing published work by a significant margin at the time of submission. We also propose a highly scalable, model-based self-supervised training framework for depth completion networks. This framework requires only sequences of RGB and sparse depth images, and outperforms a number of existing solutions trained with semi-dense annotations. Additionally, we present empirical results demonstrating that depth completion errors decrease as a power function with the number of input depth

[6] In the self-supervised framework, the training process is more iterative than training with semi-dense annotations. In particular, it takes many more iterations for the predictions to converge to the correct value. Consequently, the network weights for the RGB input, which has substantially lower correlation with the depth prediction than the sparse depth input, might have dropped to negligible levels during early iterations, resulting in similar performance for using d and RGBd as input. However, this conjecture remains to be verified.

measurements. In the future, we will investigate techniques for improving the self-supervised framework, including better loss functions and taking dynamic objects into account.

Acknowledgments

This work was supported in part by the Office of Naval Research (ONR) grant N00014-17-1-2670 and the NVIDIA Corporation. In particular, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the DGX-1 used for this research. Finally, we thank Jonas Uhrig and Nick Schneider for providing information on how data is generated for the KITTI dataset [4].

References

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[2] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[3] R. W. Wolcott and R. M. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2814–2821. IEEE, 2015.

[4] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *arXiv preprint arXiv:1708.06500*, 2017.

[5] M. Camplani and L. Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. In *Three-dimensional image processing (3DIP) and applications Ii*, volume 8290, page 82900E. International Society for Optics and Photonics, 2012.

[6] J. Shen and S.-C. S. Cheung. Layer depth denoising and completion for structured-light rgb-d cameras. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1187–1194. IEEE, 2013.

[7] S. Lu, X. Ren, and F. Liu. Depth enhancement via low-rank matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3390–3397, 2014.

[8] F. Ma, L. Carlone, U. Ayaz, and S. Karaman. Sparse sensing for resource-constrained depth reconstruction. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 96–103. IEEE, 2016.

[9] F. Ma, L. Carlone, U. Ayaz, and S. Karaman. Sparse depth sensing for resource-constrained robots. *arXiv preprint arXiv:1703.01398*, 2017.

[10] F. Ma and S. Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *arXiv preprint arXiv:1709.07492*, 2017.

[11] Y. Zhang and T. 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.

[12] J. T. Barron and B. Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

[13] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, pages 291–298, 2006.

[14] M. Hornácek, C. Rhemann, M. Gelautz, and C. Rother. Depth super resolution by rigid body self-similarity in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1123–1130, 2013.

[15] J. Xie, C.-C. Chou, R. Feris, and M.-T. Sun. Single depth image super resolution and denoising via coupled dictionary learning with local constraints and shock filtering. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.

[16] J. Lu and D. Forsyth. Sparse depth super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2245–2253, 2015.

[17] J. Xie, R. S. Feris, and M.-T. Sun. Edge-guided single depth image super resolution. *IEEE Transactions on Image Processing*, 25(1):428–438, 2016.

[18] N. Schneider, L. Schneider, P. Pinggera, U. Franke, M. Pollefeys, and C. Stiller. Semantically guided depth upsampling. In *German Conference on Pattern Recognition*, pages 37–48. Springer, 2016.

[19] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016.

[20] J. Ku, A. Harakeh, and S. L. Waslander. In defense of classical image processing: Fast depth completion on the cpu. *arXiv preprint arXiv:1802.00036*, 2018.

[21] A. Eldesokey, M. Felsberg, and F. S. Khan. Propagating confidences through cnns for sparse data regression. *arXiv preprint arXiv:1805.11913*, 2018.

[22] N. Chodosh, C. Wang, and S. Lucey. Deep convolutional compressed sensing for lidar depth completion. *arXiv preprint arXiv:1803.08949*, 2018.

[23] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[24] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[25] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[26] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. *arXiv preprint arXiv:1612.02401*, 2016.

[27] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[28] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. *arXiv preprint arXiv:1704.07813*, 2017.

[29] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, 2018.

[30] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

[31] R. Li, S. Wang, Z. Long, and D. Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. *arXiv preprint arXiv:1709.06841*, 2017.

[32] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[35] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[36] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[38] M. Carvalho, B. Le Saux, P. Trouvé-Peloux, A. Almansa, and F. Champagnat. On regression losses for deep depth estimation.