Embedding Shepard's Interpolation into CNN Models for Unguided Depth Completion

Shambel Fente Mengistu[1 [0009−0002−7738−1297]], Mara Pistellato[1 [0000−0001−6273−290X]], and Filippo Bergamasco[1 [0000−0001−6668−1556]]

DAIS, Università Ca'Foscari Venezia, 155, via Torino, Venezia Italy
{shambel.mengistu, mara.pistellato, filippo.bergamasco}@unive.it

Abstract. When acquiring sparse data samples, an interpolation method is often needed to fill in the missing information. An example application, known as "depth completion", consists in estimating dense depth maps from sparse observations (e.g. LiDAR acquisitions). To do this, algorithmic methods fill the depth image by performing a sequence of basic image processing operations, while recent approaches propose data-driven solutions, mostly based on Convolutional Neural Networks (CNNs), to predict the missing information. In this work, we combine learning-based and classical algorithmic approaches to ideally exploit the performance of the former with the ability to generalize of the latter. First, we define a novel architecture block called IDWBlock. This component allows to embed Shepard's interpolation (or Inverse Distance Weighting, IDW) into a CNN model, with the advantage of requiring a small number of parameters regardless of the kernel size. Second, we propose two network architectures involving a combination of the IDWBlock and learning-based depth completion techniques. In the experimental section, we tested the models' performances on the KITTI depth completion benchmark and NYU-depth-v2 dataset, showing how they present strong robustness to input sparsity under different densities and patterns.

Keywords: Shepard's Interpolation · Inverse Distance Weighting · Depth Completion · CNN · Sparse convolution

1 Introduction

A dense and accurate depth map is beneficial to many computer vision tasks such as 3D object detection [3, 33, 23], and reconstruction [24, 21, 22], optical flow estimation [26, 41], and semantic segmentation [37, 39]. The popular LiDAR depth sensors produce reliable observations, and are widely employed in real-world applications such as autonomous driving [14] or in industrial setups [34]. However, the resulting depth maps are too sparse, with about 5% of the acquired pixels having a valid depth value [33]. For several applications such data are not sufficient, and methods aiming at densifying sparse data samples are needed. In this context, depth completion is usually regarded as the task of recovering an accurate dense depth map from a sparse input. The literature counts several approaches to perform depth completion, ranging from classical algorithmic

methods [2, 11, 29] to learning-based techniques [1, 10, 32]. Non-learning-based approaches are based on predefined rules, do not require training data and rely only on image processing operations. However some of them, such as [11], outperform learning-based methods. Regarding learning-based approaches, state-of-the-art methods are based on deep convolutional neural networks (CNNs). When the network input is sparse and the values are irregularly distributed, applying conventional convolutions gives inaccurate results since not all the input values are actually observable [8]. There are several approaches designed to solve the input sparsity problem with CNNs. This includes the naive approach which assigns a default value to all missing pixels [12], to more effective approaches that apply sparse convolutions to weight the elements of the kernel according to a validity mask [10, 32]. The former method does not lead to optimal results, as the learned filters must be invariant to all possible validity patterns. The method proposed in [32] overcomes the problem by introducing a novel sparse convolution, while Zixuan et al. [10] proposed an extension with a multi-scale encoder-decoder CNN.

In this paper, we revisit the idea of sparsity invariant convolution and propose a family of hybrid CNN architectures that mix learning-based elements and a classic interpolation technique to perform unguided depth completion (i.e. based only on depth data). Specifically, we adopt Inverse Distance Weighted (IDW) interpolation, originally presented in [29], which can be easily reformulated as a convolution operation, and embed it in a CNN. To do so, we define the novel IDWBlock, which is able to adjust IDW parameters during training according to local data sparsity and distribution of observable input samples. Such block is combined with trainable sparse convolution layers in two alternative architectures which effectively mix the two approaches in a single or multi-scale fashion. This enables the proposed hybrid model to generate a dense and accurate depth map with clear boundaries. We tested the two proposed IDW-embedding architectures on the KITTI depth completion benchmark [32] and NYU-depth-v2 dataset [30] and show that they offer a more accurate reconstruction with respect to the simple sparse convolution approach.

2 Related Work

Data Interpolation The problem of scattered data interpolation consists in fitting a continuous function of two or more independent variables that interpolates values that are measured at some scattered points. The sparse observations can be located in a grid or can be distributed with a non-uniform pattern, making the task even more challenging. A considerable number of methods have been proposed to perform this task, from early approaches [7] to more recent solutions [20]. The inverse distance weighted (IDW) interpolation, also known as Shepard's method [29] consists in computing the values of missing points as a weighted average of the observed points, with weights being a power of the inverse of their distance. The authors of [19] applied inverse distance weighted interpolation for topographic surface modeling, while in [6] it is used for particu-

late matter (PM) estimation and mapping. Another popular method is based on radial basis function (RBF) [31, 42, 36]. In these approaches the interpolant is a weighted sum of radial basis functions (e.g. Gaussian, polynomial), that depend only on the distance between the input and a fixed point. Since the technique involves the solution of a linear system that depend on the number of points, it is unpractical in real-world applications due to computational complexity.

Depth Completion Depth completion task is a specific instance of data interpolation, where observations are scattered depth data and the goal is to recover dense depth maps. Depth completion approaches can be classified into different categories, depending on different criteria. The first categorization for depth completion methods is algorithmic or learning-based. Learning-based approaches are typically based on deep neural networks, whereas algorithmic solutions rely on a sequence of image processing techniques. Ku et al. [11] proposed to use of a sequence of well-known image processing algorithms to transform the sparse input into dense depth maps. The proposed work first utilizes morphological operations, such as dilation and closure, to make the input depth map denser, and then fills holes to obtain the final output. Based on the input data, depth completion algorithms can be divided into *guided* and *non-guided*: the former method works with an aligned RGB image used as a guide in addition to the sparse input, while the latter only works on the sparse input. Fangchang M. et al. [16] used color images as guidance in their proposed model that learns a direct mapping from sparse depth to dense depth. Alex W. et al. [35] introduced a method to infer dense depth from camera motion and sparse depth using a visual inertia odometry system, while other works [27, 38] applied a transformer-based architecture to produce a dense depth map from the given RGB image and sparse input. Fabian M. et al. [18] used a segmentation map instead of RGB image as guidance in their vgg05-like architecture. Other papers proposed by Uhrig et al. [32], Huang et al. [10], and Chodosh N. et al. [5] used the sparse depth only for depth completion, and thus are classified as non-guided. To handle sparse inputs and sparse intermediate feature maps, Uhring et al [32] proposed a non-guided sparsity-invariant convolution to replace the conventional convolution in CNNs. The sparsity-invariant CNN involves sparse convolution layers which weight the kernel elements according to the pixel validity. Additionally, a second stream carries information about the pixel validity to subsequent layers. Huang et al. [10] proposed three novel sparsity-invariant operations, based on which, a sparsity-invariant multi-scale encoder-decoder network (HMS-Net) is proposed to handle sparse data at different scales. Additional RGB features could also be incorporated to further improve the performance.

3 Combining IDW and Sparsity-invariant CNNs

We start by discussing how Shepard's interpolation can be expressed in terms of convolutions. Then, we observe how a recent approach based on Convolutional Neural Network can be seen as a special case of such an interpolation algorithm, but with trainable kernel weights. Therefore, we describe how to combine the two

so that the learnable part can be trained with a reduced number of parameters regardless of the network receptive field size.

3.1 Inverse Distance Weighted Interpolation

Inverse Distance Weighted (IDW) interpolation, also known as Shepard's method, is an old yet effective spatial interpolation approach for scattered data [29]. It creates estimates for locations without data based on values at nearby locations. The advantage of IDW interpolation includes its simplicity, ease of use, and fast execution time [13].

Suppose to have a scattered set of 2D point samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with associated values $v_1, \ldots, v_N$. Such values can represent any scalar field of interest, from terrain elevation of some topographic data to temperature values measured by an array of thermometers in an area. The IDW principle is to interpolate the value at any point $\hat{\mathbf{x}}$ as a weighted average of the values at the neighboring points. Weights are computed according to the distance between $\hat{\mathbf{x}}$ to each sampling point $\mathbf{x}_i, i = 1, \ldots, N$ as:

$$w_p(\hat{\mathbf{x}}, \mathbf{x}_i) = \frac{1}{\left(\sqrt{\hat{\mathbf{x}}^T \mathbf{x}_i}\right)^p} \tag{1}$$

where $p \geq 0$ is a free parameter that governs the relative importance to the point closer to $\hat{\mathbf{x}}$ with respect to the ones farther away.

To get the interpolated value $\hat{v}$ at a point $\hat{\mathbf{x}}$, IDW simply computes the weighted average:

$$\hat{v} = \begin{cases} \dfrac{\sum_{i=1}^{N} w_p(\hat{\mathbf{x}}, \mathbf{x}_i) v_i}{\sum_{i=1}^{N} w_p(\hat{\mathbf{x}}, \mathbf{x}_i)}, & \text{if } \hat{\mathbf{x}}^T \mathbf{x} \neq 0 \ \ \forall i \\ v_i, & \text{if } \hat{\mathbf{x}}^T \mathbf{x} = 0 \text{ for some } i. \end{cases} \tag{2}$$

Note that if $\hat{\mathbf{x}}$ coincides with any of the given points, the interpolated value is given directly by v_i since $w(\hat{\mathbf{x}}, \mathbf{x}_i)$ would be undefined in that case. This also agrees with the mathematical definition of "interpolation" which provides a continuous function passing exactly at the given samples. It is easy to observe that the higher the value of p, the more $\hat{v}$ will converge to the value of the nearest neighbour of $\hat{\mathbf{x}}$, as its relative weight will dominate the others. On the other hand, small values of p tend to produce a smoother interpolation since $\hat{v}$ is averaged among several neighbouring values. In the extreme case, when $p = 0$, all the interpolated values will be equal to the average of the given values $v_1 \ldots v_N$. The original formulation described so far can be used for any scattered point set but involve the computation of distances between the interpolated points and the given data points. Since we are dealing with sparse depth images, point coordinates are restricted to the image lattice. In such a case, we can precompute the

weights among pixel pairs at certain distances and perform the same operation in terms of convolutions.

Let S be the size of a sparse depth image I[1]. Let M be a binary mask of the same size of I containing 1 for each valid pixel in I and 0 for the missing values. We can compute the $S \times S$ correlation kernel:

$$\mathcal{K}_{S,p} = \begin{pmatrix} k_{-\frac{S}{2},-\frac{S}{2}} & \dots & k_{\frac{S}{2},-\frac{S}{2}} \\ \vdots & \ddots & \vdots \\ k_{-\frac{S}{2},\frac{S}{2}} & \dots & k_{\frac{S}{2},\frac{S}{2}} \end{pmatrix}, \; k_{i,j} = \begin{cases} 0, & \text{if } i = 0 \wedge j = 0 \\ w_p\Big(\big(i \; j\big)^T, \mathbf{0} \Big), & \text{otherwise} \end{cases} \tag{3}$$

weighting the contribution of the neighbouring pixels with respect to the center of the kernel. For example, $\mathcal{K}_{3,1} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \\ 1 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 1 & \frac{1}{\sqrt{2}} \end{pmatrix}$.

It is easy to see that the interpolation in (2) can be expressed in terms of convolution as follows:

$$\hat{I} = M \cdot I + (1 - M) \cdot \frac{I \star \mathcal{K}}{M \star \mathcal{K}} \tag{4}$$

Since $\mathcal{M}$ is a binary mask, $\mathcal{M} \star \mathcal{K}$ computes the sum of the weights for each given sample used to normalize the weighted average defined in (2). To enforce the resulting value to be equal to v_i at each sample $\mathbf{x}_i$, the interpolated value is overwritten by the original value in I by the linear combination with the binary mask. For this reason, it has to be noted that $\mathcal{K}(0,0)$ can be chosen arbitrarily without affecting the resulting $\hat{I}$ since each produced value is overwritten every time the convolution kernel is centered on a given sample.

Considering IDW in terms of convolutions allows us to modify its formulation providing an additional parameter controlling the resulting interpolation. Indeed, the described operation is well-defined even if the size of the kernel is smaller than S. In that case, instead of computing the value in $\hat{\mathbf{x}}$ as the weighted average of all the given samples, we restrict the average to the neighbouring samples closer to $\hat{\mathbf{x}}$ by half the kernel size. This allows the control of the interpolation "receptive field" to limit the contribution of samples farther away even with low power values. We define the Convolutional-IDW interpolator as follows:

$$\text{CIDW}_{s,p}(I, M) = M \cdot I + (1 - M) \cdot \frac{I \star \mathcal{K}_{s,p}}{(M + \epsilon) \star \mathcal{K}_{s,p}} \tag{5}$$

where $\epsilon > 0$ is a small constant to avoid division by zero. Since $s < S$, CIDW will not produce a valid value if no sample falls within the area encompassed by the convolution kernel. Therefore, the interpolation produced by the function is undefined in all the pixels where the output mask $M' = \text{sign}(M \star \mathcal{K}_{s,p})$ is zero.

[1] We can assume without loss of generality that I is square and that $S = 2a+1, a \in \mathbb{N}$. If that is not the case, I can be padded with zeros to meet such condition.

3.2 Sparsity-Invariant CNNs

The sparsity-invariant CNNs proposed in [32] is an effective way to modify conventional convolutions in a CNN to handle sparse input feature maps (i.e. when the input layer $\mathbf{x}$ can only be partially observed at the locations in which the binary mask $\mathbf{m}$ is 1). The sparsity-invariant convolution is formulated as:

$$f_{u,v}(\mathbf{x}, \mathbf{m}) = \frac{\sum_{i,j=-a}^{a} m(u+i, v+j)x(u+i, v+j)w(i,j)}{\sum_{i,j=-a}^{a} m(u+i, v+j) + \epsilon} + b \tag{6}$$

where $\mathbf{w}$ is a learnable kernel of size $(2a+1) \times (2a+1)$, b is a scalar bias, and ϵ is again a small constant to avoid division by zero at locations where none of the input values are observed.
Such formulation can be seen as a generalized version of CIDW in which the weights are fully trainable instead of being a function of samples' location as in (1). However, the normalization component is conceptually different. In CIDW we compute the weighted arithmetic mean of input values with the kernel values. In the sparsity-invariant convolution, instead, the linear combination between input and weights is normalized by the number of observed values encompassed by the kernel, regardless of the weights' values. Also in the case of sparse convolutions, some output values might be invalid. The authors propose to produce the output mask by doing a max-pooling operation with a unitary stride and the same kernel size as the one used for the convolution. This produces the same result as computing the output mask M' as we described before.

3.3 IDWBlock: embedding CIDW in a Sparse CNN

Since the Shepard's interpolation can be formulated in terms of convolutions (as shown in Eq. 5), we studied the idea of embedding such operation into a classical CNN model. In particular, we define a new architectural block, called IDWBlock, with a limited set of parameters that learns the optimal way to combine a set of $CIDW$ outputs performed with different trainable power values $p_1 \dots p_N$.

The architecture of an IDWBlock is sketched in Fig. 1. The upper part performs several CIDW operations with different combinations of kernel sizes and power values. Kernel sizes are hard-coded into the block architecture and therefore cannot be trained. We observed that 5×5, 17×17, and 37×37 are good values for the most popular datasets for depth completion. For each kernel size, CIDW is executed with a set of different power values. Specifically, the 5×5 kernel is applied with 3 power values (p_1, p_2, p_3), the 17×17 with 4 power values and 37×37 with 3. All the $p_1, \dots, p_{11}$ are randomly chosen at the beginning and optimized during the training together with all other network weights. Outputs of each of the 11 CIDW operations are stacked depth-wise to produce a $W \times H \times 11$ tensor $\mathcal{C}$, where W, H are the width and height of the input image.

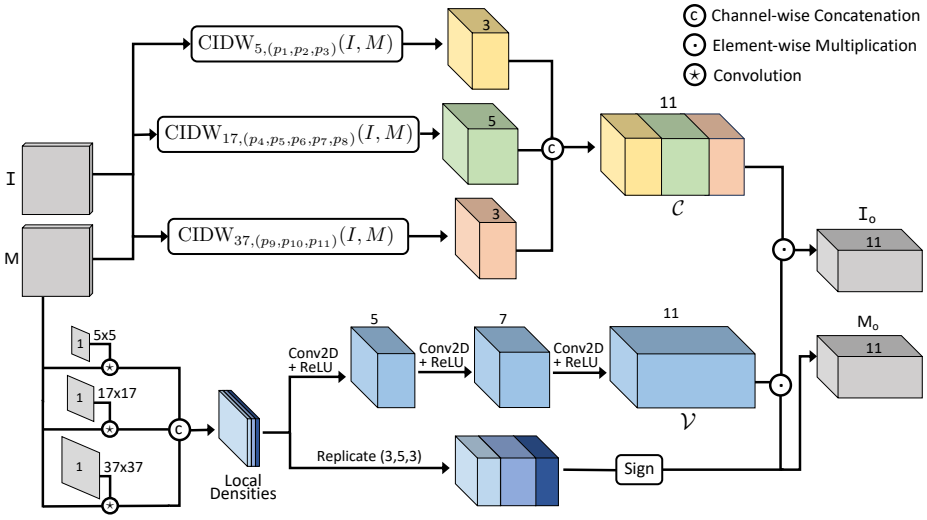

Fig. 1. The Proposed IDWBlock architecture. See text for details.

The lower part of the block learns the relative importance of the produced CIDW outputs $\mathcal{C}$ according to the local density of the given samples. Indeed, we expect that the optimal combination of CIDW kernel size and power value is significantly different if the input samples are very close or far away in a certain region. To approximate the density, we convolve the input binary mask M with unitary (non-trainable) kernels with size 5×5, 17×17, and 37×37. Outputs are stacked together in a $W \times H \times 3$ tensor and processed with a classical feed-forward convolutional network to expand into a $W \times H \times 11$ tensor $\mathcal{V}$ representing the relative weight of each CIDW output for each pixel. Since some of the CIDW outputs might be invalid (especially with small kernels), we must force the corresponding weights to zero. The validity information corresponds to pixels where the local density for a certain kernel size is greater than zero. Therefore, we compute the sign of local densities and multiply it element-wise to $\mathcal{V}$. Finally, CIDW outputs are rescaled with the element-wise product $\mathcal{C} \odot \mathcal{V}$, and the resulting I_o is returned with the per-element validity information M_o.

3.4 Adding IDWBlock into Sparsity Invariant CNNs

We propose two different ways to arrange IDWBlocks in a sparsity invariant CNN. The first (IDWNet) is sketched in Fig. 2 and contains just a single IDWBlock used in parallel with a sequence of sparse convolutions (SparseConv) followed by ReLU activations, as described in [32]. Output images of the IDWBlock are concatenated with the output feature maps of the SparseConvs, as well as the respective output masks. To match the channelwise dimension of the IDWBlock mask, the one-channel SparseConv mask is replicated before concatenation. After the concatenation, the resulting multi-channel I and M

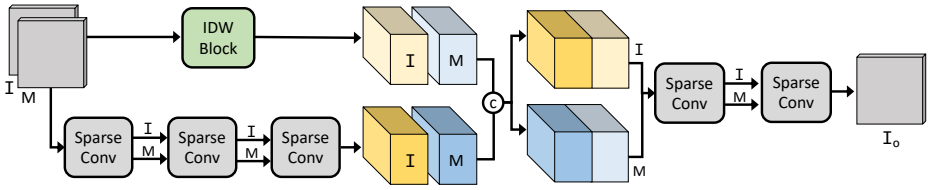

Fig. 2. IDWNet architecture. The input is processed by our IDW block and by a sequence of three trainable sparse convolution layers. Results are concatenated and processed by a second sequence of sparse convolutions.

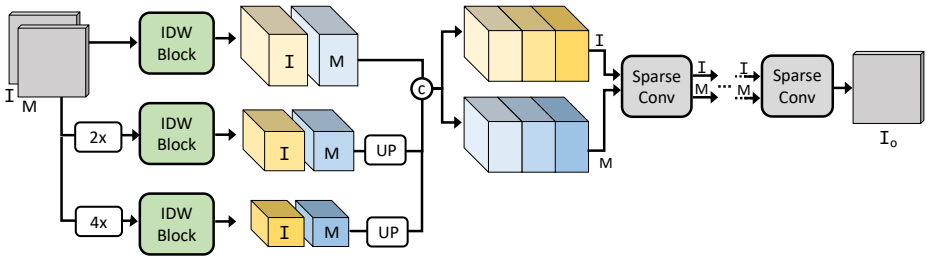

Fig. 3. MS-IDWNet architecture. Input is processed at different scales (original, 2× and 4×) and concatenated. Dense output is produced by a sequence of sparse convolutions.

tensors are processed with additional SparseConv blocks to obtain the final interpolated image. Note that, at this point, within the SparseConv module the input multi-channel mask is squeezed into a single-channel mask by channel-wise summation followed by sign operation: this is done because a single-channel mask with max-pooling operation is propagated from layer to layer as proposed in [32].

The second architecture (MS-IDWNet) is shown in Fig. 3. This time we investigated a multi-scale arrangement in which three parallel IDWBlocks are fed with the original input image and a 2×, 4× down-scaled version respectively. Down-scaling is performed by doing average pooling on both the image and the mask and then normalizing the obtained image with the down-scaled mask. The effect is equivalent as computing the average only on valid samples. For the up-scaling, we perform a nearest-neighbour interpolation on both the image and the mask. The two operations are not trainable. Finally, outputs of all the IDWBlocks are concatenated and fed to a sequence of sparse convolutions as in the previous architecture.

4 Experimental Section

To evaluate our approach, we conducted our experiments on both the KITTI depth completion dataset and NYU-depth-v2 dataset.

KITTI-Depth dataset [32] includes sparse depth maps (5% of the pixels available) from projected LiDAR point clouds that were matched against the stereo

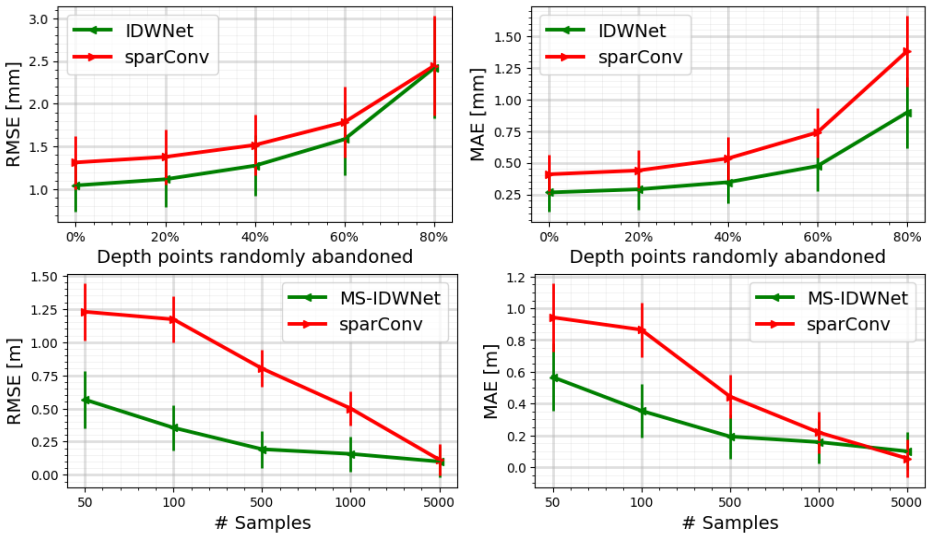

Fig. 4. Testing robustness of our method varying input sparsity levels. First row shows results on KITTI depth completion benchmark: the values on x-axis show the percentage of randomly abandoned input points. Second row shows the same evaluation on NYU dataset for different sparsity levels (number of input samples).

cameras. The dataset has 86k training images, 7k validation images, and 1k test set images with no access to the ground truth. We used all the 86k depth maps for training and a validation subset of 1k images for evaluation. As the top part of the images have no valid values, we removed the top 103 rows and center crop for training and validation. For testing, we used the original image size.

NYU-depth-v2 [30] is an RGB-D dataset for indoor scenes, captured with a Microsoft Kinect with size 640×480. Similarly to [17, 4], we used the official split with roughly 48k images for the training and 654 for testing. Each input image was randomly sampled with a uniform distribution.

We trained both our models on an NVIDIA GeForce RTX 4080 using the Mean Squared Error (MSE) loss function. All parameters were randomly initialized and updated with ADAM optimizer configured with an initial learning rate of 0.01. During the train we applied the Learning rate decaying equation described in [10]. In all our experiments we used the proposed IDWNet (Fig. 2) for the KITTY dataset and MS-IDWNet (Fig. 3) for NYU-depth-v2.

4.1 Evaluation

We started by testing the effect of different input sparsity for our technique. Figure 4 shows the reconstruction error varying the test set sparsity. Plots on the left show Root Mean Squared Error (RMSE), while plots on the right display the MAE. The first row analyses the behaviour when abandoning an increasing

Table 1. Comparisons of different methods against the validation set of the KITTI dataset [32].Data of SegGuided [18] is as reported in the paper, Other competing methods are as reported in [10].

Method	RMSE (mm)	MAE (mm)
U-Net [28]	1387.35	445.73
FRRN [25]	1148.27	338.56
PSPNet [40]	1185.39	354.21
FPN [15]	1441.82	473.65
He et al. [9]	1056.39	293.86
HMS-Net [10]	994.14	262.41
SegGuided [18]	1146.78	278.75
SparseConv [32]	1314.23	409.17
IDWNet (our)	1045.34	265.08

number of points on KITTI dataset. The subsampling was performed by randomly deleting samples with an increasing probability (from 0 to 0.8), keeping the points distribution consistent with the original data by performing the operation with a sliding window. We compared our IDWNet (combining CIDW blocks with sparse convolutions) with the sparsity invariant CNN as described in [32]. Our proposed model exhibits a consistent improvement in terms of both MAE and RMSE at any density level. The second row shows the same experiment performed on NYU dataset: since ground truth is dense, we uniformly sampled the data to obtain different sparsity levels (from 50 up to 5000 points per image). Also in this case, our approach performs better with respect to the sparse convolutions approach, since IDWBlock is able to adapt to different sparsity levels, even for significantly sparser samples.

Table 2. Errors by different methods on NYU-DEPTH-V2 test set. The values are taken from their respective papers. "w/RGB" indicates RGB image used.

	Method	RMSE (m)	REL (m)	δ_1	δ_2	δ_3
200 points	SparseConv [32]	1.065	0.257	0.550	0.752	0.880
	Sparse-to-dense [16]	0.259	0.054	0.963	0.992	0.998
	HMS-Net [10]	0.233	0.044	—	—	—
	MS-IDWNet (our)	0.255	0.055	0.955	0.991	0.988
500 points	SparseConv [32]	0.801	0.159	0.739	0.861	0.933
	Sparse-to-dense [16] w/RGB	0.230	0.044	0.971	0.994	0.998
	SPN [4] w/RGB	0.162	0.027	0.985	0.997	0.999
	MS-IDWNet (our)	0.190	0.038	0.975	0.996	0.999

Table 1 reports quantitative results of our IDWNet versus other encoder-decoder and unguided depth completion techniques on the KITTI validation set. We report the RMSE and MAE values (in mm) for U-Net [28], FRRN [25],

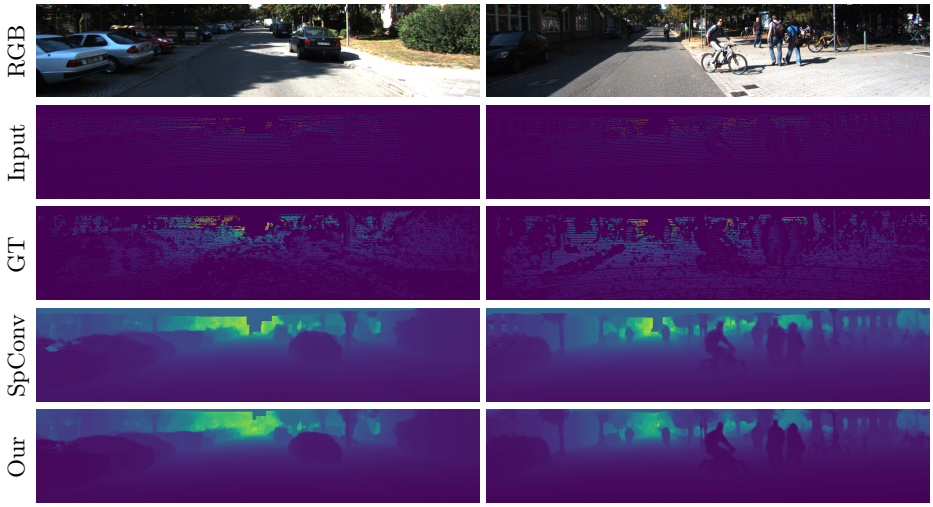

Fig. 5. Visual qualitative example of the result obtained on the KITTI validation set by our method and the Sparsity Invariant CNN (SpConv) [32].

PSPNet [40], FPN [15], He et al. [9], HMS-Net [10] and SparseConv [32]. Our approach performs better with respect to other methods, and it is comparable with the HMS-Net architecture (especially the MAE), which however involves a complex multi-scale structure. Table 2 reports comparisons on NYU test set while handling very sparse inputs (200 and 500 samples). In this case, we included the Sparse-to-dense [16] and SPN [4] that were designed to work well with a low number of samples (with and without RGB guidance). Also for this case, our method shows good results against other state of the art methods while exhibiting a substantially simpler architecture.

Finally, Fig. 5 and Fig. 6 display qualitative outputs on KITTI and NYU datasets respectively. In both the cases, we compare our approach with the sparsity invariant CNN as we did for the experiment shown in Fig. 4. In general, we observe that our multi-scale IDW architecture offers sharper edges with respect to the results obtained from SparseConv.

5 Conclusions

In this paper we proposed two CNN architectures for unguided depth completion. Both models includes the novel IDWBlock, which embeds Shepard's interpolation with sparse convolutions. We show that by mixing algorithmic and learning-based interpolation approaches can offer better performances with a minimal increase in the number of training parameters. Moreover, our approach predicts accurate depths, without requiring different treatment for different sparsity levels. Experimental results showed the advantage of the proposed method for depth completion without adding too much complexity.

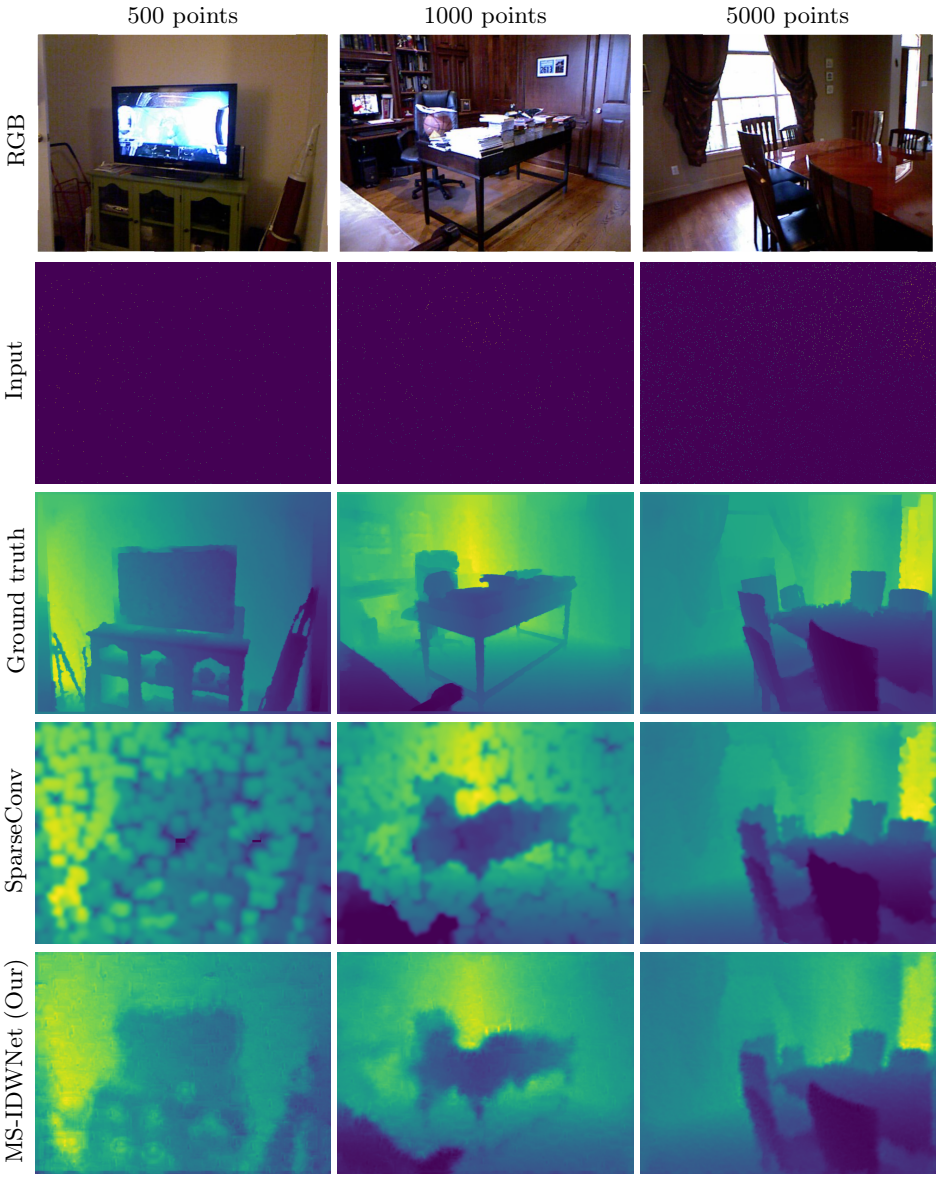

Fig. 6. Visual results for IDWNet and sparseConv [32] from NYU dataset. Our IDWNet produces objects with sharp and clear boundaries.

References

1. Alhashim, I., Wonka, P.: High quality monocular depth estimation via transfer learning. ArXiv **abs/1812.11941** (2018)
2. Barron, J.T., Poole, B.: The fast bilateral solver. ArXiv **abs/1511.03296** (2015)
3. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2147–2156 (2016). https://doi.org/10.1109/CVPR.2016.236
4. Cheng, X., Wang, P., Yang, R.: Depth estimation via affinity learned with convolutional spatial propagation network. In: European Conference on Computer Vision (2018)
5. Chodosh, N., Wang, C., Lucey, S.: Deep convolutional compressed sensing for lidar depth completion. ArXiv **abs/1803.08949** (2018)
6. Choi, K., Chong, K.: Modified inverse distance weighting interpolation for particulate matter estimation and mapping. Atmosphere **13**(5) (2022). https://doi.org/10.3390/atmos13050846, https://www.mdpi.com/2073-4433/13/5/846
7. Franke, R.: Scattered data interpolation: tests of some methods. Mathematics of computation **38**(157), 181–200 (1982)
8. Gasparetto, A., Ressi, D., Bergamasco, F., Pistellato, M., Cosmo, L., Boschetti, M., Ursella, E., Albarelli, A.: Cross-dataset data augmentation for convolutional neural networks training. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 910–915. IEEE (2018). https://doi.org/10.1109/ICPR.2018.8545812
9. He, L., Wang, G., Hu, Z.: Learning depth from single images with deep neural network embedding focal length. IEEE Transactions on Image Processing **27**(9), 4676–4689 (2018)
10. Huang, Z., Fan, J., Cheng, S., Yi, S., Wang, X., Li, H.: Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion. IEEE Transactions on Image Processing **29**, 3429–3441 (2018)
11. Ku, J., Harakeh, A., Waslander, S.L.: In defense of classical image processing: Fast depth completion on the cpu. In: 2018 15th Conference on Computer and Robot Vision (CRV). pp. 16–22 (2018). https://doi.org/10.1109/CRV.2018.00013
12. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. ArXiv **abs/1608.07916** (2016)
13. Li, J., Heap, A.D.: A review of comparative studies of spatial interpolation methods in environmental sciences: Performance and impact factors. Ecological Informatics **6**(3), 228–241 (2011). https://doi.org/https://doi.org/10.1016/j.ecoinf.2010.12.003, https://www.sciencedirect.com/science/article/pii/S1574954110001147
14. Li, Y., Ibanez-Guzman, J.: Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. IEEE Signal Processing Magazine **37**(4), 50–61 (2020)
15. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
16. Ma, F., Cavalheiro, G.V., Karaman, S.: Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. 2019 International Conference on Robotics and Automation (ICRA) pp. 3288–3295 (2018)

17. Ma, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. 2018 IEEE International Conference on Robotics and Automation (ICRA) pp. 1–8 (2017)
18. Märkert, F., Sunkel, M., Haselhoff, A., Rudolph, S.: Segmentation-guided domain adaptation for efficient depth completion. ArXiv **abs/2210.09213** (2022), https://api.semanticscholar.org/CorpusID:252918440
19. Mulkal, M., Wandi, R.: Inverse distance weight spatial interpolation for topographic surface 3d modelling. TECHSI - Jurnal Teknik Informatika **11**, 385 (10 2019). https://doi.org/10.29103/techsi.v11i3.1934
20. Nielson, R., Franke, R.: Scattered data interpolation and applications: A tutorial and survey. Geometric Modeling: Methods and Their Applications pp. 131–160 (2012)
21. Pistellato, M., Albarelli, A., Bergamasco, F., Torsello, A.: Robust joint selection of camera orientations and feature projections over multiple views. In: Proceedings - International Conference on Pattern Recognition. p. 3703 – 3708 (2016). https://doi.org/10.1109/ICPR.2016.7900210
22. Pistellato, M., Bergamasco, F., Albarelli, A., Torsello, A.: Dynamic optimal path selection for 3d triangulation with multiple cameras. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9279**, 468 – 479 (2015). https://doi.org/10.1007/978-3-319-23231-7_42
23. Pistellato, M., Bergamasco, F., Albarelli, A., Torsello, A.: Robust cylinder estimation in point clouds from pairwise axes similarities. ICPRAM 2019 - Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods p. 640 – 647 (2019). https://doi.org/10.5220/0007401706400647
24. Pistellato, M., Cosmo, L., Bergamasco, F., Gasparetto, A., Albarelli, A.: Adaptive albedo compensation for accurate phase-shift coding. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 2450–2455. IEEE (2018). https://doi.org/10.1109/ICPR.2018.8545465
25. Pohlen, T., Hermans, A., Mathias, M., Leibe, B.: Full-resolution residual networks for semantic segmentation in street scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4151–4160 (2017)
26. Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., Black, M.J.: Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12232–12241 (2019). https://doi.org/10.1109/CVPR.2019.01252
27. Rho, K., Ha, J., Kim, Y.: Guideformer: Transformers for image guided depth completion. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6240–6249 (June 2022). https://doi.org/10.1109/CVPR52688.2022.00615
28. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
29. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference. pp. 517–524 (1968)
30. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: European Conference on Computer Vision (2012)
31. Skala, V.: Rbf interpolation with csrbf of large data sets. Procedia Computer Science **108**, 2433–2437 (2017)

32. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity invariant cnns. In: International Conference on 3D Vision (3DV) (2017)
33. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M.E., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 8437–8445 (2018)
34. Wei, P., Cagle, L., Reza, T., Ball, J., Gafford, J.: Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. Electronics **7**(6), 84 (2018)
35. Wong, A., Fei, X., Tsuei, S., Soatto, S.: Unsupervised depth completion from visual inertial odometry. IEEE Robotics and Automation Letters **5**(2), 1899–1906 (2020). https://doi.org/10.1109/LRA.2020.2969938
36. Wright, G.B.: Radial basis function interpolation: numerical and analytical developments. University of Colorado at Boulder (2003)
37. Ye, J., Ji, Y., Wang, X., Ou, K., Tao, D., Song, M.: Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2824–2833 (2019)
38. Zhang, Y., Guo, X., Poggi, M., Zhu, Z., Huang, G., Mattoccia, S.: Completionformer: Depth completion with convolutions and vision transformers. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 18527–18536 (2023)
39. Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., Yang, J.: Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4101–4110 (2019). https://doi.org/10.1109/CVPR.2019.00423
40. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)
41. Zhu, A.Z., Yuan, L., Chaney, K., Daniilidis, K.: Unsupervised event-based learning of optical flow, depth, and egomotion. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 989–997 (2018)
42. Zou, Y.L., Hu, F.L., Zhou, C.C., Li, C.L., Dunn, K.J.: Analysis of radial basis function interpolation approach. Applied Geophysics **10**(4), 397–410 (2013)