Distance Transform Pooling Neural Network for LiDAR Depth Completion

Yiming Zhao, *Member, IEEE*, Mahdi Elhousni, *Student Member, IEEE*, Ziming Zhang, *Member, IEEE*, and Xinming Huang, *Senior Member, IEEE*

***Abstract*— Recovering dense depth maps from sparse depth sensors, such as LiDAR, is a recently proposed task with many computer vision and robotics applications. Previous works have identified input sparsity as the key challenge of this task. To solve the sparsity challenge, we propose a recurrent distance transform pooling (DTP) module that aggregates multi-level nearby information prior to the backbone neural network. The intuition of this module is originated from the observation that most pixels within the receptive field of the network are zero. This indicates a deep and heavy network structure has to be used to enlarge the receptive field aiming at capturing enough useful information as most processed signals are uninformative zeros. Our recurrent DTP module can fill in empty pixels with the nearest value in a local patch and recurrently transform distance to reach farther nearest points. The output of the proposed DTP module is a collection of multi-level semi-dense depth maps from original sparse to almost full. Processing this collection of semi-dense depth maps alleviates the network from the input sparsity, which helps a lightweight simplified ResNet-18 with 1M parameters achieve state-of-the-art performance on the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) depth completion benchmark with LiDAR only. Besides the sparsity, the input LiDAR map also contains some incorrect values due to the sensor error. Thus, we further enhance the DTP with an error correction (EC) module to avoid the spreading of the incorrect input values. At last, we discuss the benefit of only using LiDAR for nighttime driving and the potential extension of the proposed method for sensor fusion and the indoor scenario. The code has been released online at https://github.com/placeforyiming/DistanceTransform-DepthCompletion.**

***Index Terms*— Depth completion, distance transform (DT), neural networks, sensor fusion.**

I. Introduction

MAPPING LiDAR points on the image plane will create a sparse depth map with values for only about 5% pixels. The LiDAR-based depth completion is a task to complete this sparse depth map to dense [1], [2]. The initial challenge of this task is claimed as input sparsity, which creates unique difficulty for regular convolution layers [3]. Most early-stage works were focusing on this sparsity challenge by designing special convolution kernels [3], [4] or structures [5]. However, it was demonstrated in [2] that directly sending sparse input into a large encoder-decoder structure with ResNet-34 as the backbone network can solve the problem with good performance. This stimulates us to think—Is sparsity really a challenge for regular convolution neural networks? Although depth completion attracts many successive research works [6], [7], few of them are still focusing on this sparse property. In this article, we first investigate the initially claimed sparsity challenge by conducting statistical analysis. We find the long-tail distributed sparse pattern that some empty pixels are extremely far away from input points. This explains why sparsity is a challenge for early-stage small networks but seems not a problem when the network is large enough to cover a long distance. Inspired by the sparsity pattern, we further design a distance transform (DT) module with error correction (EC). The proposed module significantly improves the ability of small networks on depth completion task and helps a lightweight ResNet-18 with only 1M parameters achieve state-of-the-art performance.

A. Input Sparsity and Outliers

LiDAR and camera have different sensing mechanisms to perceive the world. This difference makes those mapped depth values unevenly distribute on the image plane. As shown in Fig. 1(a), some points cluster together and create large empty regions in the orange dash box. The displacement of sensor position will also map some points from the occluded background on the foreground object. Those wrongly mapped values should be identified and corrected. We give an example of those incorrect values in the green dash box of Fig. 1(a). Directly filling empty pixels with those values will spread the error to the nearby region.

B. Statistical Analysis

To understand more about the sparsity and outliers, we conduct statistical analysis on the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) validation set [8] by analyzing the nearest value of each empty pixel. We first calculate the cumulative percentage of the city-block distance between each empty pixel and its nearest value on the image plane. As shown in Fig. 1(b), the green line depicts the long-tail effect that some empty pixels have a large distance to even the nearest value. We further use the nearest value to predict each empty pixel and calculate the average RMSE with the ground truth shown as the blue bar in Fig. 1(b). In general, the nearest value prediction will have a smaller error if the city-block or l_1 distance is smaller. The exception happens when the city-block distance is close to zero. This

Manuscript received April 15, 2021; revised September 8, 2021; accepted November 10, 2021. This work was supported by the U.S. National Science Foundation (NSF) under Grant CCF-2006738. *(Corresponding author: Xinming Huang.)*

The authors are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: xhuang@wpi.edu).

Digital Object Identifier 10.1109/TNNLS.2021.3129801

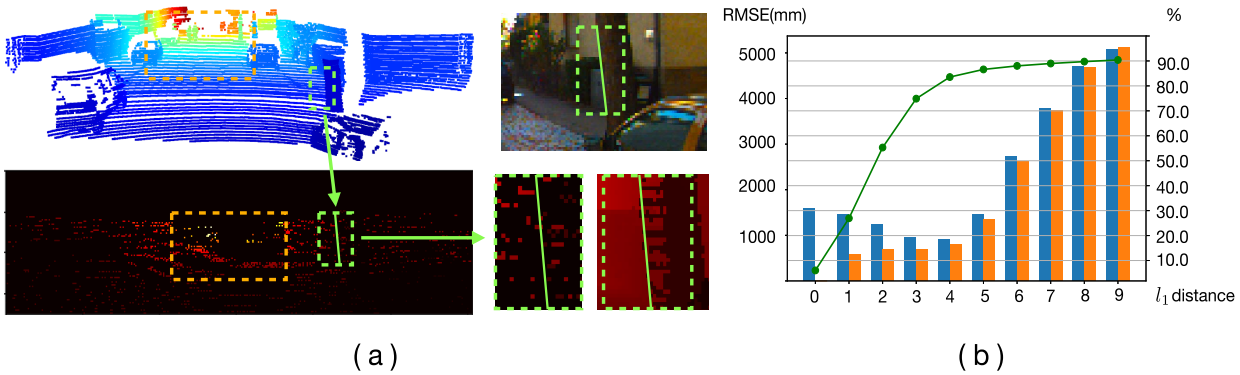

Fig. 1. (a) Pattern of input sparsity and outliers. The sparse pattern is unevenly distributed on the image plane. Simply using the nearest value to fill in the empty pixels will spread the error in the nearby region as shown in the green dash box. (b) Statistical analysis of the KITTI validation set. The Green line is the accumulative percentage of empty pixels which have smaller l_1 distances to their nearest point. The Blue bar is the average root mean square error (RMSE) error by filling in the empty pixel with the nearest value. Due to the sensor error on the KITTI dataset, some points of the sparse input have large errors compared with ground truth. To show this as the Orange bar, we replace those input values with the ground truth, then calculate the average error of the nearest filling.

exception is because of the incorrect depth values in the sparse input. Therefore, we replace the sparse input's value with the corresponding ground truth and show the statistical result again as the orange bar. The trend is clear that the closer nearest value has a smaller RMSE error as the initial guess of the empty pixel.

C. Observations and Motivation

The statistical analysis shows several properties of sparsity. First, some empty pixels are far away from the useful information as they have a large distance to even the nearest input value. Second, if the nearest value is close, it can already provide a good initial guess for the empty pixel. Third, in some cases, the sparse input itself is erroneous. The first property explains why those early-stage models have the sparsity challenge but a simple encoder–decoder structure can solve the problem once the backbone network is large enough [2]–[4]. This is because predicting depth values for those pixels far away from other input values needs many convolution layers with pooling to reach the far away from useful information. The second property suggests a straightforward solution to this sparsity challenge by filling in the empty pixel with close nearest value. The third one indicates the extra EC needs to be considered for this real-world challenge.

All those three observations build our motivation. We propose a distance transform pooling (DTP) module that recurrently transforms the distance to locate the nearest value within various distance masks. To avoid the spreading of the incorrect input value, we further design an EC module ahead of the DTP module. The DT with EC can generate multi-level sparsity depth maps as the input of the network, thus enlarging the receptive field of the network to ease the sparsity challenge. The multi-scale input also implicitly encodes the distance information that informs the network how the nearest initial can be trusted. One can refer to our previous publication [9] for more discussions from the geometric perspective.

D. Proposed Approach and Contributions

Based on the discussion above, we propose our final solution for the task of depth completion or sparse-to-dense. It consists of three parts: an EC module to correct the sensor error, a DTP module to utilize the nearest value, and a final refinement network to smooth the dense depth map. Specifically, we define as follow.

1) We propose a specific DTP operator for LiDAR data that can be implemented on GPU effectively and efficiently. We convert each LiDAR scan into a binary image and apply (truncated) city-block distance to locate the nearest LiDAR point of each pixel in the depth image. Recurrently applying this operator will get depth maps with various density levels. We process each depth map with a regular convolution layer and concatenate outputs together. The backbone network is expected to help each pixel learn the true depth value from the nearest value within different distance thresholds. After this operator, the original sparse input will be replaced with multi-level semi-dense depth maps, thus there is no more sparsity challenge.
2) We propose an EC module before the DTP module to correct outlier values in LiDAR data while preserving its sparsity. The EC module consists of a small convolutional network working on the modification of the nonempty pixels from LiDAR data. The input error is generated by mapping the real point on the wrong surface due to the displacement of two sensors. Therefore, incorrect values stay with other normal values to create a denser region. This explains why a small network is sufficient to serve as an EC module as it only needs to perceive a denser local region. The training of parameters in the EC module relies on our DTP to pass the gradient.
3) State-of-the-art performance with a lightweight backbone. We simplify a standard ResNet-18 by reducing the number of filters in each layer to 64 so that the number of network parameters is ∼1M, leading to fast inference speed (∼0.04 s per sample in our experiments). We achieve state-of-the-art performance by using LiDAR only with this backbone. Since our DTP overcomes the sparsity challenge, we demonstrate the resilience of the model performance to smaller networks. We further discuss the benefit of only using LiDAR sensor on nighttime driving, and also show simply adding the RGB branch in our proposed pipeline

can achieve good performance for sensor fusion and indoor scenario. All those benefits indicate our proposed DT with EC is a practical solution to depth completion that can work for various demands.

II. Related Work

A. Depth Completion With Classical Methods

Before the wide adoption of outdoor LiDAR for autonomous driving, depth completion was researched to fill up the depth map generated from the RGB-D camera. Some classical methods were proposed to solve the image inpainting or depth quality improvement problem for RGB-D depth map [10]–[14]. The ratio of missing data from RGB-D depth map is roughly around 10%–30%, which is different from LiDAR points' sparsity. Besides completing the generated depth map from the RGB-D camera, some researchers assumed only a few points have been matched by the stereo camera and complete this sparse input to dense by using optimization methods like compressive sensing [15]–[17]. The natural difference between LiDAR sensors and those RGB-D sensors urges researchers to design new outdoor LiDAR depth completion methods. Recently, Ku *et al.* [18] proposed a fast method for LiDAR-based depth completion with simple image processing operators, including dilation, Gaussian blur, etc. This simple method even obtained comparable result with some learning-based methods.

B. Depth Completion With Self-Supervised Learning

Self-supervised learning can train the network without labels. Self-supervised depth prediction relies on the image warping and photometric loss to penalize the error [19], [20]. Ma *et al.* [2] extended the self-supervised depth prediction framework to depth completion by feeding the sparse points into the network and treating them as the ground truth for corresponding pixels. Moreover, Yang *et al.* [21] integrated their conditional prior network to the self-supervised depth completion pipeline and got a better performance. Wong *et al.* [22] further considered pose consistency and geometric compatibility for performance improvement. Yao *et al.* [23] proposed a binary anisotropic diffusion tensor to eliminate smoothness constraint at intended positions and directions. Those self-supervised methods do not need labels, which allows them to update weights online. This convenience is useful for many advanced situations like federated learning and user privacy protection.

C. Depth Completion With Supervised Learning for Sparse LiDAR Only

The recent success of deep neural networks on computer vision tasks has inspired researchers to solve the depth completion task using deep learning. Uhrig *et al.* [3] claimed sparsity as the main challenge for this task. They also set up the depth completion task on KITTI, which attracts many successive studies. Chodosh *et al.* [24] provided a solution for depth completion by combining compressed sensing and deep learning using the alternating direction neural network (ADNN) framework. This method is inspired by how people handle sparse signals. Eldesokey *et al.* [4] designed a normalized convolution layer that only consists of the depth map and confidence map as two channels. This method is also extendable with RGB images as extra guidance [25]. Huang *et al.* [5] proposed a model consisting of both the specially designed structure and the specially designed kernel that can slightly outperform a standard encoder-decoder structure [2] where the raw sparse depth images are fed into a large 34-layer network with a residual module and multiple transpose convolution layers. This work demonstrates that regular 2-D convolution can process the sparse LiDAR input if the network is deep enough. They also provided a self-supervised solution with photo-consistency loss guided by images. To overcome the computational issues in large networks, Eldesokey *et al.* [26] proposed a specific lightweight network. The idea of co-learning with image reconstruction has also be explored [27].

D. Depth Completion With Supervised Learning for Sensor Fusion

The simplest way to do sensor fusion is concatenating RGB images with LiDAR [2], [28]. More advanced solutions are also proposed recently in different directions. From the view of geometry, several papers considered surface normal or the 3-D points into the model design [6], [29], [30]. One paper [21] modeled the depth completion from the view of probability. By considering the purpose of object detection, researchers [31] focused on the depth edge's clearance. Inspired by spatial propagation, some papers achieved good performance by designing some post-processing modules [7], [32]. The idea of graph network is also considered by one recent paper [33].

E. In Summary of the Depth Completion Task

From the view of input modality and if the label is available, LiDAR depth completion solution can be summarized as the non-learning solution, self-supervised learning solution, supervised learning for LiDAR only and supervised learning for sensor fusion. Each kind of solution has its own benefits and limitations. In this article, we mainly focus on supervised depth completion for LiDAR only. Since solving the sparsity challenge is our primary goal, we believe working on the single modality is more persuasive as those sensor fusion models with RGB often need complicated network structures and specific fusion strategies, which will distract the effect of sparsity. As a LiDAR only solution, our model also has the unique benefit that can work without worrying about the lighting condition, shown as a nighttime driving example.

F. DT Operator and Our Implementations

DT operator is one of the most classic computer vision and machine learning techniques [34]. The research around this useful operator keeps alive nowadays [35], [36]. In many recent application systems, the DT plays an important role. For example, it has been used to localize spinal cord from the MRI data [37] toward fully automated clinical utilization.

It also has been tried to solve the association problem for dense semantic information [38] in map-based robot localization. Not limited to Euclidean distance, the DT also can work with other distance definitions. The MBD (minimum barrier distance) is a commonly used distance definition on RGB pixels [39]. The MBD transform is a crucial part of some salient object detection models [40], [41]. Most DT algorithms are implemented on CPU, and some recent works start to explore how to utilize GPU to make the DT run in parallel [36], [42]. In this article, instead of focusing on low-level CUDA programming, we are using modern deep learning frameworks, such as Tensorflow and Pytorch, to implement DT. Therefore, our design can easily be integrated with deep learning models to allow both forward and backward propagation.

III. Our Approach

A. Problem Setup

Fundamentally, LiDAR-based depth completion is a nonlinear regression problem. Let $\{(\mathbf{x}_i, \mathbf{y}_i)\} \in \mathcal{X} \times \mathcal{Y}$ be the training set where $\mathbf{x}_i, \forall i$ denotes a sparse LiDAR image and $\mathbf{y}_i, \forall i$ denotes its semi-dense ground-truth depth image. Then the depth completion problem can be formulated as

$$\min_{\psi \in \Psi} \sum_i \ell\big(\mathbf{y}_i, \psi(\mathbf{x}_i) \odot \mathbf{1}_{\mathbf{y}_i}\big) \tag{1}$$

where $\psi : \mathcal{X} \to \mathcal{Y}$ denotes the depth completion mapping function from the feasible space Ψ, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denotes a loss function such as the least-square, $\mathbf{1}_{\mathbf{y}_i}$ denotes the binary image of $\mathbf{y}_i$ consisting 1 for non-zeros in $\mathbf{y}_i$ and 0 otherwise, and $\odot$ denotes the entry-wise multiplication.

B. Error Correction (EC)

We treat the EC as a function $\mathbf{x}'_i = \phi(\mathbf{x}_i) \odot \mathbf{1}_{\mathbf{x}_i}$, where $\phi(\cdot)$ is modeled using regular convolutional neural networks (CNNs). The entry-wise multiplication with a binary mask $\mathbf{1}_{\mathbf{x}_i}$ helps $\mathbf{x}'_i$ share the same sparsity as $\mathbf{x}_i$. In this article, we propose modeling $\phi(\cdot)$ using four regular convolutional layers with kernel sizes $7 \times 7, 5 \times 5, 3 \times 3, 3 \times 3$ and channel numbers 16, 16, 16, 1. We design such a lightweight network based on the error analysis of recent papers [6]. The input sensor error of LiDAR is generated by mapping some occluded points from the background to the foreground. An example is shown in Fig. 1(a) that mapping the point cloud on the image plane will map some points from the background wall on the foreground tree. This creates a denser region with a mixture of incorrect values and correct values. The displacement of LiDAR and camera creates this error mode that suggests the EC module should focus on identifying outliers from the clue in the local region, thus inspires us to approximate $\phi(\cdot)$ with a small network. We show our EC module in Fig. 2. A mask is used to help the small network only focus on existing sparse values.

1) Training of the EC Module: Note, the training of this EC module does not need any extra settings. The next DTP operator works like other pooling operators, such as max pooling, that allow the pass of both the forward and backward propagation. Parameters in this EC module will be updated with other parameters along the gradient descent of the final loss function.

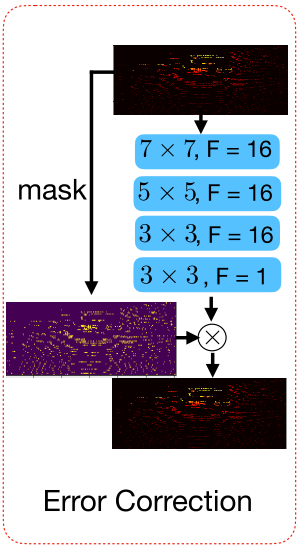

Fig. 2. Proposed error correction (EC) module.

C. Distance Transform Pooling (DTP)

1) Distance Transform (DT): DT [34] is an operator usually applied to binary images to generate distance maps, which are grayscale images of the same size of the original images consisting of the distance as well as the offset to the closest available signal as each pixel value. In deep learning, DT is usually used as an off-shelf data pre-processing step. For instance, in [43] DT with Euclidean distance was applied to segmentation label masks whose outputs were directly used in training deep models for better regularization. In this article, we propose integrating DT with the training of deep models to locate the nearest neighbors for raw depth estimation. The city-block distance is used to keep consistency with our statistical analysis. The implementation also supports the approximation of Euclidean distance, we show a performance comparison in ablation study. However, compared with the Euclidean distance, the city-block distance has the following nice properties:

Proposition 1 (Recurrent Formula for City-Block Distance): Given arbitrary pair of pixel locations $\mathbf{x}, \mathbf{y} \in \mathcal{I}$ from an image where $\mathcal{I}$ denotes the collection of all pixel locations, their city-block distance, $d_{\ell_1}(\mathbf{x}, \mathbf{y})$, can be computed recursively using a DT kernel, $\mathcal{H}$, as follows:

$$d_{\ell_1}(\mathbf{x}, \mathbf{y}) = \begin{cases} \|\mathbf{x} - \mathbf{y}\|_1, & \mathbf{x} \in \mathcal{H}_{\mathbf{y}} \\ \min\limits_{\mathbf{z} \in \mathcal{I}} \{d_{\ell_1}(\mathbf{x}, \mathbf{z}) + d_{\ell_1}(\mathbf{z}, \mathbf{y})\}, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{x} \in \mathcal{H}_{\mathbf{y}}$ indicates that the pixel at $\mathbf{x}$ falls into the kernel centered at $\mathbf{y}$.

Proposition 2 (Receptive Field With Recurrent Formula): Given the size of the city-block DT kernel as $k \times k$, we can exactly compute the city-block DT outputs with kernel size $(l(k-1)+1) \times (l(k-1)+1)$ with l repeats.

Therefore, a city-block DT operator can accurately compute the outputs in a recurrent way. As one example shown in Fig. 3, with a 3×3 kernel as $[2\,1\,2; 1\,0\,1; 2\,1\,2]$, we can repeatedly use a small DT kernel to compute the outputs of larger DT kernels, thanks to the nice recursive properties.

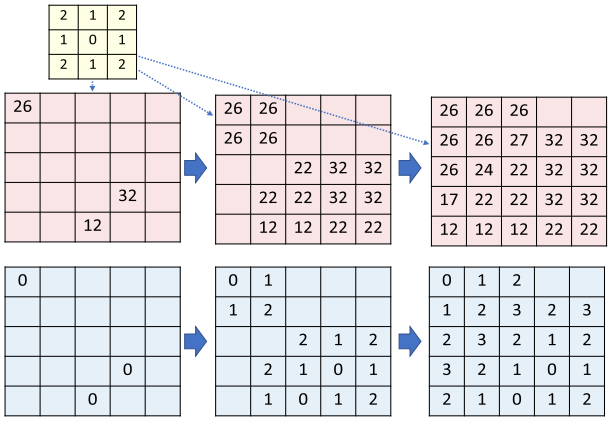

Fig. 3. Example of city-block distance transform. (Top) 3×3 truncated city-block kernel. (Middle) depth maps with LiDAR input. (Bottom) distance maps. Equivalently the sequential repeats output the results with kernels 3×3, 5×5, and 7×7 accordingly.

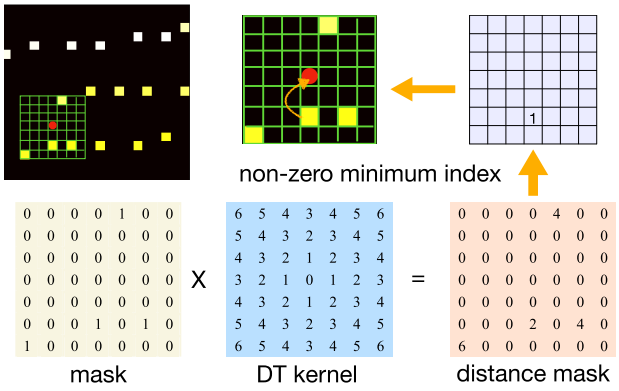

Fig. 4. Example of distance transform pooling on the sparse depth input. There is a 7×7 kernel working for the empty pixel marked with a red spot.

2) DTP: With those nice properties, we further illustrate how to utilize DT to locate the nearest value for each empty pixel as a pooling operator, named DTP. In DTP Algorithm 1, p denotes the pixel location in the patch, $\mathbf{U}_{ij}(p), \mathbf{V}_{ij}(p)$ denote the depth value and distance mask value at p in the patches, respectively.

Algorithm 1 DTP: Distance Transform Pooling Operator

Input : LiDAR image $\mathbf{X}$, DT kernel $\mathbf{H}$, kernel size k
Output: Depth map $\mathbf{Z}$

$\mathbf{Z} \leftarrow \mathbf{X}$;
$\mathbf{M} \leftarrow$ a binary mask to indicate if $\mathbf{X} > 0$;
foreach *empty pixel at* (i, j) *in* $\mathbf{Z}$ **do**
  Crop the patches $\mathbf{U}_{ij}$ and $\mathbf{V}_{ij}$ from $\mathbf{M}$ and $\mathbf{Z}$, respectively, with the same size as $\mathbf{H}$ centered at location (i, j);
  $\mathbf{U}_{ij} \leftarrow (k - \mathbf{H}) \odot \mathbf{U}_{ij}$; // $\odot$: Entry-wise product
  $t \leftarrow 0$;
  foreach *index* p *in matrix* $\mathbf{U}_{ij}$ **do**
    if $\max_p \mathbf{U}_{ij}(p) == \mathbf{U}_{ij}(p)$ **then**
      $\mathbf{Z}_{ij} \leftarrow \mathbf{Z}_{ij} + \mathbf{V}_{ij}(p), t \leftarrow t + 1$;
    end
  end
  $\mathbf{Z}_{ij} \leftarrow \mathbf{Z}_{ij}/t$;
end
return $\mathbf{Z}$

One example of how this operator work on sparse depth input is shown in Fig. 4. For the empty pixel marked with red color, the 7×7 kernel will locate the nearest non-zero value within it and put the value to the center empty pixel. The same as the other pooling operators, the DTP is also differentiable

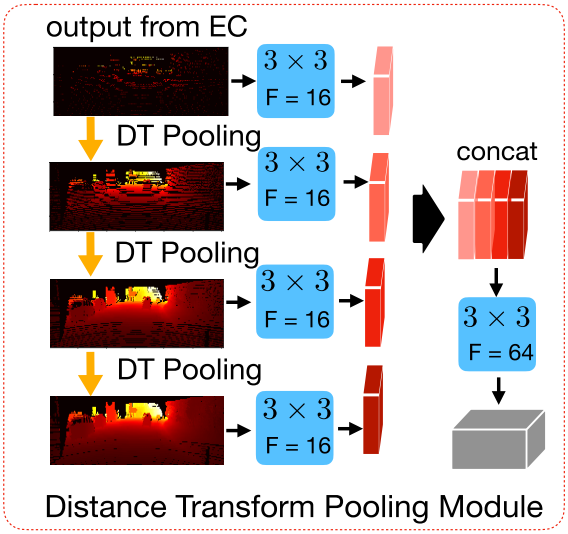

Fig. 5. Structure of our distance transform pooling (DTP) module. The DTP module takes in the output of the EC and generates a tensor to the final refinement network.

which makes the whole pipeline end-to-end trainable. Note that, the index of non-zero minimum value is not a function supported by all the modern deep learning frameworks, so we use an inverse kernel $H' = k - H$ equivalently to find the maximum index on distance mask in Algorithm 1, where the k is the kernel size and the H is the original distance kernel. This helps the pooling algorithm only rely on basic mathematical calculations supported by all the modern deep learning frameworks.

3) DTP Module: The DTP recurrently finds the nearest value within a distance mask to generate multi-level semi-dense depth maps. We propose to process each semi-dense depth map with a convolution layer and concatenate the output together. From the statistical analysis, we know the error of the nearest initial guess is proportional to the distance. Therefore, individually processing all the middle-level depth maps can inform the distance information to the network. We visualize the structure of our DTP module in Fig. 5. This module keeps the original sparse input as well as eases the sparsity challenge.

D. Network Architecture

We illustrate our framework in Fig. 6, where the backbone network can be any existing convolutional network structure.

To demonstrate our capability of completing depth using lightweight networks, we use a simplified ResNet-18 encoder-decoder structure whose number of filters in each layer is reduced to 64, leading to about 1M parameters. This light-weight backbone structure, shown in Fig. 7, gives us a favor to help all the experiments in this article can be conducted on a single RTX 2080ti GPU with 11 GB memory. Note that many papers use an encoder-decoder ResNet-34 as the backbone in the field of depth completion [2], [7], [32]. Compared with it, the only difference in our structure is the fewer layers and filters. We use the mean square error (MSE) loss during training.

As a neural network model, many factors will affect the test performance. In order to better illustrate our proposed DTP module does solve the sparsity challenge, we use the most

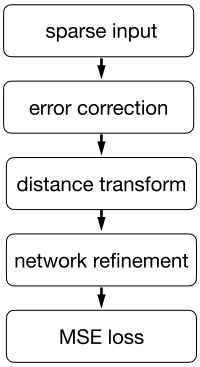

Fig. 6. Model pipeline.

common MSE loss although other loss functions may achieve better performance [26], [44]. The backbone network is also the commonly used ResNet structure. In the ablation study, we further reduce the number of layers to investigate the model resilience to a smaller structure with a limited receptive field. The model with our DTP module is much more robust than the baseline.

IV. Experiments

Datasets: We conduct our experiments on KITTI [8] dataset. KITTI is a well-known public dataset for autonomous driving, as a benchmark for depth completion [3]. This benchmark contains a collection of sparse LiDAR scans as well as corresponding ground-truth semi-dense depth maps. It consists of 85 898 training samples, 1000 validation samples, and 1000 test samples. The test ground truth is unavailable, requiring researchers to submit the predicted test results to the server to get the final metric scores.

Training and Evaluation: Without heavy fine-tuning, in all the experiments we train our method using the MSE loss for 1. We utilize Adam as our optimizer with a learning rate 0.0001 as initial, decreasing by half after every 2 epochs. For the experiments on KITTI using LiDAR only, we train 8 epochs, and for the remaining experiments, we train 20 epochs in total. We set the batch size to 2 on KITTI and 4 on NYU-Depth-v2. All the experiments are conducted on a single NVIDIA RTX 2080Ti with 11 GB memory.

Following [5], [25], we report our results in terms of RMSE, mean absolute error (MAE), RMSE of the inverse depth (iRMSE), and MAE of the inverse depth (iMAE). By default, the unit measurement for each metric is mm, mm, 1/km, 1/km, respectively.

Hyperparameters in DTP Operator: There are two hyperparameters, the DT kernel size k and the number of repeats l which determines the range of truncated city-block distance. According to the statistical analysis in Fig. 1(b), above 90% pixels have the nearest value within city-block distance 9 on the KITTI validation set. Thus we choose $k = 7$ and $l = 3$, which can cover a minimum distance as 9. Here the hyperparameter choosing is heuristic, a small kernel size needs more repeats with more parameters, and a large kernel size will lose distance details. Those two hyperparameters are able to be tuned case-by-case. When it comes to other sensor settings, like sparser LiDAR or camera with other resolutions, those two hyperparameters can be changed to meet the new sparsity scenario. In this article, we demonstrate this heuristic chosen parameter set can already achieve state-of-the-art performance on KITTI's test leaderboard with LiDAR only.

TABLE I
State-of-the-Art Performance Comparison on KITTI Test Dataset Using LiDAR Only

Methods	#params	RMSE	MAE	iRMSE	iMAE	ExInfo
Spade-sD[44]	5.3M	1035.29	**248.32**	**2.60**	**0.98**	Synthetic
Glob-guide[45]	2.0M	922.93	249.11	2.80	1.07	Semantic
IR_L2[27]	11.0M	**901.43**	292.36	4.92	1.35	RGB
SparseConv[3]	25k	1601.33	481.27	4.94	1.78	No
ADNN [24]	1.7k	1325.37	439.48	59.39	3.19	No
IP-Basic[18]	-	1288.46	302.60	3.78	1.29	No
NConv-CNN[25]	0.5k	1268.22	360.28	4.67	1.52	No
pNCNN[26]	0.7M	960.05	251.77	3.37	**1.05**	No
SparseToDense[2]	5.5M	954.36	288.64	3.21	1.35	No
HMS-Net[5]	-	937.48	258.48	**2.93**	1.14	No
Ours (32 filters)	0.3M	956.44	260.93	3.44	1.19	No
Ours (64 filters)	1.0M	**937.27**	**247.81**	2.94	1.07	No

TABLE II
Effectiveness of the Error Correction Module on KITTI Validation Set With LiDAR Only

EC Module	RMSE	MAE	iRMSE	iMAE
w/o.	991.52	260.28	2.97	1.09
w.	984.19	250.86	2.94	1.06

A. Performance Comparison on KITTI

In Table I, we compare our model performance with all the other LiDAR only methods on KITTI's leaderboard. Our method achieves the best MAE with a small number of parameters. There are some methods of using extra information during training. For instance, Glob-guide [45] needs a pre-trained model with semantic segmentation label, Spade-sD [44] uses synthetic data, and IR_L2 [27] requires the auxiliary information from RGB. If we exclude those three methods, our solution also has the best RMSE. The other two indicators iRMSE and iMAE of our method are still close to the state-of-the-art. Since those four indicators evaluate different aspects of the method (iRMSE and iMAE are more sensitive to the close depth), we see that the performance of our method is systematically better than the others without extra information during training.

In particular, pNCNN [26] is a method with lightweight networks for depth completion. To demonstrate the robustness of our method to lightweight networks as well, we reduce the number of filters in each layer of ResNet-18 from 64 to 32. This helps us decrease the total number of parameters to 0.3M, leading to inferior performance that still has a better RMSE than pNCNN.

B. Ablation Study

1) Effectiveness of EC Module: To verify the effectiveness of the EC, we conduct experiments on KITTI using LiDAR only. Table II lists our results, where we can see the improvement. We further visualize the corrected values in Fig. 8. We use a mask to indicate those input values which have been corrected larger than 3 meters. It is clear to see all those

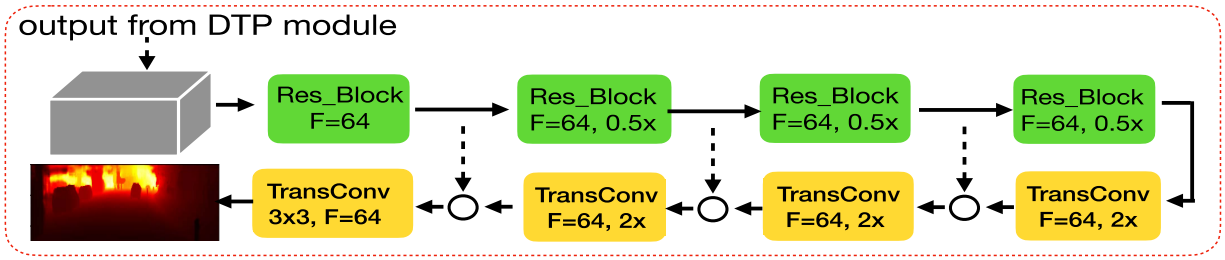

Fig. 7. Structure of the backbone network. ○ represents simple concatenation. The ResNet block is the same as the block used in the standard ResNet-18 with fewer channels.

Fig. 8. Outliers with values which have been corrected larger than 3 m by our error correction module. Almost all those points are coming from object boundaries.

identified outliers are distributed on the boundary of objects. This consolidates the error source that background points are mapped on the foreground object due to the different views of LiDAR and camera.

Besides visualization, we also specifically evaluate the sparse input after correction with some numerical indicators. Here the correction aims to reduce those large erroneous values instead of giving accurate predictions, so we choose δ_i as the indicator which reflects this goal better. δ_i is the percentage of predicted pixels where the relative error is within a threshold. Specifically

$$\delta_i = \frac{\text{card}\left(\left\{\hat{y} : \max\left\{\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right\} < 1.25^i\right\}\right)}{\text{card}(\{y_i\})}$$

where y_i and $\hat{y}_i$ are respectively the ground truth and the prediction, and card is the cardinality of a set. A higher δ_i indicates better prediction. This metric is also used in many related papers, such as [1], [46]. RMSE(mm) is also used to keep consistency with the previous evaluation. Note, the evaluation in Table III only considers those pixels with values in both the input and the ground truth. We can see our EC module largely improve the accuracy of the sparse input. This helps the DTP module to avoid the spreading of few errors from the input.

2) Effectiveness of DTP Module and Robust to Smaller Networks: As we discussed, input sparsity is a particular challenge when the network is small. Here we demonstrate our DTP module will help the small network keep resilient to the sparse input.

TABLE III

COMPARISON THE ERROR OF SPARSE INPUT WITH/WITHOUT OUR ERROR CORRECTION MODULE

	RMSE(mm)	$\delta_1(\%)$	$\delta_2(\%)$	$\delta_3(\%)$
Original Input	1595.24	98.9	99.3	99.6
After Error Correction Module	838.67	99.4	99.7	99.9

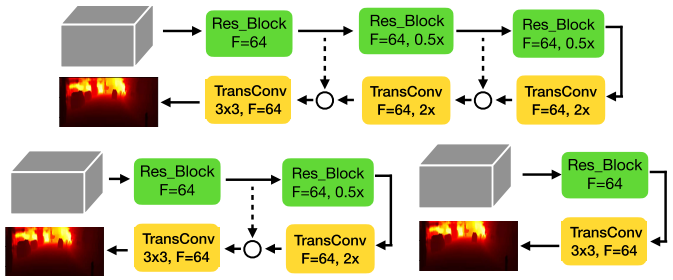

Fig. 9. Smaller structures with fewer ResNet blocks.

TABLE IV

COMPARISON OF DIFFERENT DT KERNEL ON KITTI VALIDATION SET WITH LIDAR ONLY

DT kernel	RMSE	MAE	iRMSE	iMAE
Euclidean	986.52	251.28	2.95	1.05
City Block	984.19	250.86	2.94	1.06

The backbone network showed in Fig. 7 has four repeating residual blocks which are the same as the block used in ResNet-18. To further explore the model performance with a smaller network, we reduce the number of residual blocks displayed in Fig. 9. Then, we have four network structures with different repeating numbers from one to four. The comparison is shown in Fig. 10, where we can see the smaller the network the larger improvement our method can achieve.

3) Various Distance Kernels: To cope with the statistical analysis and keep the exact same distance with recurrent DT, we use city-block distance in this article. Our distance kernel also supports the recurrent approximation of other distances, such as Euclidean distance. We list the performance comparison here in Table IV. The performance difference between those two distance kernels is small.

4) Comparison With Direct Nearest Filling: Instead of our DTP, directly filling in all the empty pixels with the nearest value before the network is a straightforward way to utilize the nearest neighbor. This idea has been explored by some recent papers with the corresponding OpenCV operator [47], [48]. Compared with this straightforward idea, our DTP has two advantages: 1) the recurrent pooling with kernels can serve as part of the network structure to allow the EC module end-to-end trainable and 2) as unveiled by our statistical analysis,

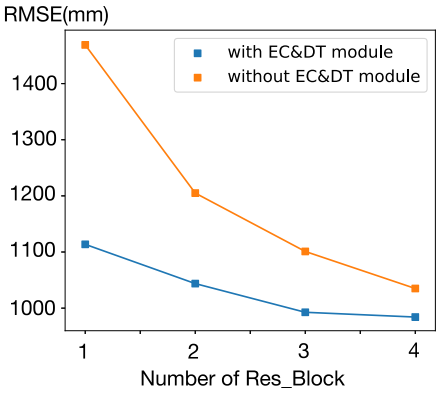

Fig. 10. Performance comparison with different number of residual blocks on the KITTI validation dataset.

TABLE V

COMPARISON OF OUR PROPOSED MODULE WITH DIRECT NEAREST FILLING WITH A REFINEMENT NETWORK USED IN OTHER PAPERS

DT kernel	RMSE	MAE	iRMSE	iMAE
Nearest filling + Backbone	1011.82	258.74	3.06	1.10
Our DTP + Backbone	984.19	250.86	2.94	1.06

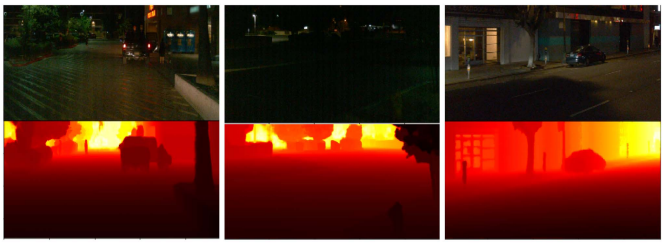

Fig. 11. Examples of model performance on nighttime samples from Waymo dataset with weights trained on KITTI.

filling in all the pixels with the nearest value will involve significantly different initial guesses for those empty points that have a large distance to input values. Here, we further compare the performance of our DTP module with the baseline proposed by two recent papers [47], [48] in Table V.

V. DISCUSSION

A. Robust to Lighting Condition

LiDAR sensor provides accurate measurement without being affected by lighting conditions. This feature compensates for the flaw of the camera that makes LiDAR become an almost inevitable choice in the setting of modern autonomous cars. Since the designed DTP module is working on LiDAR only, it keeps the ability to work on various lighting conditions.

Unfortunately, the current KITTI benchmark does not support the quantitative evaluation of this feature. To demonstrate this ability, we quantitatively visualize some nighttime driving samples in Fig. 11. Those samples are picked from the Waymo dataset [49] with the model trained on KITTI. We believe the consistency of generating depth maps in a whole day is important for fully autonomous driving.

B. Extension With RGB Image

Although our method is built by analyzing properties on the sparse depth map, we show our module is also able to be easily fused into the pipeline with RGB images in Fig. 12. The output tensor from our DTP module is concatenated with different stages of the network for better fusion. The backbone

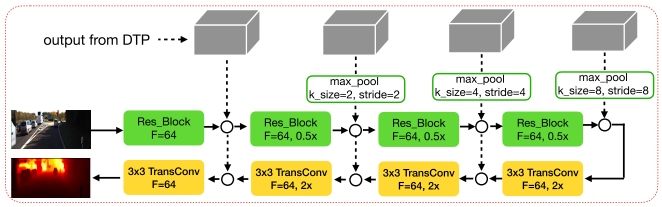

Fig. 12. Network structure to fuse the RGB image.

TABLE VI

PERFORMANCE COMPARISON WITH THREE RECENT METHODS THAT CAN WORK ON BOTH LIDAR ONLY AND SENSOR FUSION

Methods	LiDAR only		LiDAR+RGB	
	RMSE	MAE	RMSE	MAE
SparseToDense[2]	954.36	288.64	814.73	249.95
HMS-Net [5]	937.48	258.48	841.78	253.47
NConv-CNN [25]	1268.22	360.28	829.98	233.26
Ours	**937.27**	**247.81**	**776.92**	**221.01**

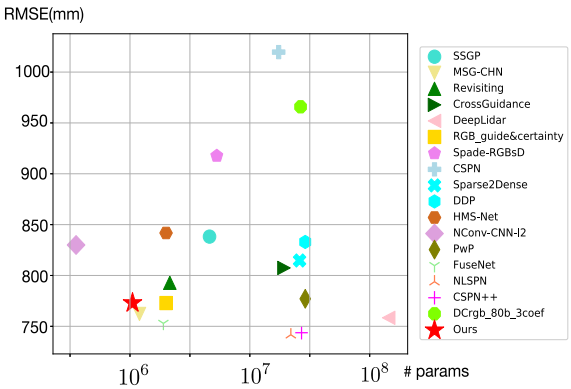

Fig. 13. Trade-off comparison between our method and other sensor fusion solutions with LiDAR and RGB images. Here, we consider the number of parameters as the indicator for model complexity and RMSE as model performance.

structure is still the lightweight ResNet-18 with 1M parameters in Fig. 7.

The same as reported in other papers [2], [5], [25], adding RGB information will improve the performance. We compare performance with these three recent methods which can work on both LiDAR only and LiDAR plus RGB in Table VI.

To further compare with all sensor fusion solutions on KITTI, we show our fusion performance with others by measuring the RMSE error and the number of parameters in Fig. 13. Two methods have better RMSE with a closer number of parameters. FuseNet [30] needs KD-tree before each continuous convolution layer, leading to slow inference time. MSG-CHN [50] explores how to use different combinations of hourglass networks to construct a better encoder-decoder structure, while we just use a standard ResNet-18 structure. Note that there are some potential ways to improve the performance, such as better fusion strategies [25], larger or more complicated networks [6], [50], other loss functions [45], etc. As the primary goal of this article is proposing DTP to solve the LiDAR sparsity challenge, we simply keep all those settings standard.

C. Performance on Indoor Scenario

Most LiDAR sensors are used for outdoor scenarios due to the high energy and long-range. Compared with outdoor, indoor scenarios are usually choosing cheaper and short-range

TABLE VII
RMSE(M) AND REL COMPARISON ON NYU-DEPTH-V2 DATASET WITH 200 SAMPLE POINTS

Methods	200 points		200 points + RGB	
	RMSE	REL	RMSE	REL
Ma [52]	0.259	0.054	0.230	0.044
Spade-sD [44]	0.246	0.051	-	-
SparseToDense [2]	0.245	0.049	-	-
HMS-Net [5]	0.233	0.044	0.212	0.041
DGCG [53]	-	-	0.225	0.046
InductiveFusion [54]	-	-	0.169	0.028
NConv-CNN [25]	-	-	0.171	0.026
Ours	**0.211**	**0.035**	**0.159**	**0.023**

depth sensors such as stereo cameras or RGBD cameras. Some recent papers [5], [46], [51] assume there are sparse depth maps already known, maybe from the matched features on epipolar line or solid-state LiDAR, then conduct a depth completion model to fill the sparse depth map to dense for the indoor scenario. Here we show the model proposed in this article also works well for the indoor scenario with NYU-Depth-v2 dataset [14].

Following Mal and Karaman [52] and other prior works [5], [46], [51], we resize each original image to 320 $\times$ 240 and crop the boundary region with inaccurate labels, leading to a resolution of 304 $\times$ 228. Then, we sample 200 points as the sparse depth input. Note, some papers sample 500 points to test the sensor fusion model with RGB images [6], [29], [32], we only sample 200 points because there are models work on both sparse depth only and sensor fusion with this setting.

To keep consistency with previous settings, we still use the 7 $\times$ 7 DT kernel with three repeats. Though this may lead to suboptimal performance, we find the result is good enough to demonstrate the effectiveness of our method on this dataset. Depth completion on NYU-Depth-v2 is an artificial task that the sparse depth input is sampled from the ground truth, so the outlier correction module is disabled as there is no input error. We list our comparison results with 200 points in Table VII, where REL refers to mean absolute relative error [52].

VI. CONCLUSION

In this article, we address the problem of LiDAR depth completion from the perspective of sparsity. By taking the data knowledge into the network design, we propose a DTP module that recurrently explores nearest neighbors within different distance thresholds. A trainable EC module is further proposed to work with the DTP to avoid the spread of incorrect values. The proposed approach demonstrates state-of-the-art performance on KITTI and it also brings some useful features, such as the robustness to smaller networks, the easy solution to removing blurred boundary points, and the ability to be extended with RGB. All those illustrate the practicability of our proposed depth completion solution.

REFERENCES

[1] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4796–4803.

[2] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3288–3295.

[3] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant CNNs," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 11–20.

[4] A. Eldesokey, M. Felsberg, and F. S. Khan, "Propagating confidences through CNNs for sparse data regression," in *Proc. 29th Brit. Mach. Vis. Conf. (BMVC)*. England, U.K.: BMVA Press, Sep. 2019, pp. 591.1–591.11.

[5] Z. Huang, J. Fan, S. Cheng, S. Yi, X. Wang, and H. Li, "HMS-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion," *IEEE Trans. Image Process.*, vol. 29, pp. 3429–3441, 2020.

[6] J. Qiu *et al.*, "DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3313–3322.

[7] X. Cheng, P. Wang, C. Guan, and R. Yang, "CSPN++: Learning context and resource aware convolutional spatial propagation networks for depth completion," in *Proc. AAAI*, 2020, pp. 10615–10622.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[9] Y. Zhao, L. Bai, Z. Zhang, and X. Huang, "A surface geometry model for LiDAR depth completion," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4457–4464, Jul. 2021.

[10] S. Lu, X. Ren, and F. Liu, "Depth enhancement via low-rank matrix completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3390–3397.

[11] J. Shen and S.-C.-S. Cheung, "Layer depth denoising and completion for structured-light RGB-D cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1187–1194.

[12] J. Park, H. Kim, Y. Tai, M. S. Brown, and I. S. Kweon, "High-quality depth map upsampling and completion for RGB-D cameras," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5559–5572, Dec. 2014.

[13] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin, "Shading-based shape refinement of RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1415–1422.

[14] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2012, pp. 746–760.

[15] S. Hawe, M. Kleinsteuber, and K. Diepold, "Dense disparity maps from sparse disparity measurements," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2126–2133.

[16] L.-K. Liu, S. H. Chan, and T. Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1983–1996, Jun. 2015.

[17] F. Ma, L. Carlone, U. Ayaz, and S. Karaman, "Sparse sensing for resource-constrained depth reconstruction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 96–103.

[18] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the CPU," in *Proc. 15th Conf. Comput. Robot Vis. (CRV)*, May 2018, pp. 16–22.

[19] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1851–1858.

[20] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 270–279.

[21] Y. Yang, A. Wong, and S. Soatto, "Dense depth posterior (DDP) from single image and sparse range," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3353–3362.

[22] A. Wong, X. Fei, S. Tsuei, and S. Soatto, "Unsupervised depth completion from visual inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1899–1906, Apr. 2020.

[23] Y. Yao, M. Roxas, R. Ishikawa, S. Ando, J. Shimamura, and T. Oishi, "Discontinuous and smooth depth completion with binary anisotropic diffusion tensor," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5128–5135, Oct. 2020.

[24] N. Chodosh, C. Wang, and S. Lucey, "Deep convolutional compressed sensing for LiDAR depth completion," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2018, pp. 499–513.

[25] A. Eldesokey, M. Felsberg, and F. S. Khan, "Confidence propagation through CNNs for guided sparse depth regression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2423–2436, Oct. 2020.

[26] A. Eldesokey, M. Felsberg, K. Holmquist, and M. Persson, "Uncertainty-aware CNNs for depth completion: Uncertainty from beginning to end," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12014–12023.

[27] K. Lu, N. Barnes, S. Anwar, and L. Zheng, "From depth what can you see? Depth completion via auxiliary image reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11306–11315.

[28] J. Tang, F.-P. Tian, W. Feng, J. Li, and P. Tan, "Learning guided convolutional network for depth completion," *IEEE Trans. Image Process.*, vol. 30, pp. 1116–1129, 2021.

[29] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse LiDAR data with depth-normal constraints," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2811–2820.

[30] Y. Chen, B. Yang, M. Liang, and R. Urtasun, "Learning joint 2D-3D representations for depth completion," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 10023–10032.

[31] S. Imran, Y. Long, X. Liu, and D. Morris, "Depth coefficients for depth completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12446–12455.

[32] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. So Kweon, "Non-local spatial propagation network for depth completion," 2020, *arXiv:2007.10042*.

[33] X. Xiong, H. Xiong, K. Xian, C. Zhao, Z. Cao, and X. Li, "Sparse-to-dense depth completion revisited: Sampling strategy and graph construction," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 682–699.

[34] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *J. ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[35] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Comput. Surv.*, vol. 40, no. 1, pp. 1–44, 2008.

[36] M. Manduhu and M. W. Jones, "A work efficient parallel algorithm for exact Euclidean distance transform," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5322–5335, Nov. 2019.

[37] C. Gros *et al.*, "OptiC: Robust and automatic spinal cord localization on a large variety of MRI data using a distance transform based global optimization," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2017, pp. 712–719.

[38] J.-H. Pauls, K. Petek, F. Poggenhans, and C. Stiller, "Monocular localization in HD maps by combining semantic segmentation and distance transform," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 4595–4601.

[39] R. Strand, K. C. Ciesielski, F. Malmberg, and P. K. Saha, "The minimum barrier distance," *Comput. Vis. Image Understand.*, vol. 117, no. 4, pp. 429–437, Apr. 2013.

[40] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech, "Minimum barrier salient object detection at 80 FPS," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1404–1412.

[41] A. Wang and M. Wang, "RGB-D salient object detection via minimum barrier distance transform and saliency fusion," *IEEE Signal Process. Lett.*, vol. 24, no. 5, pp. 663–667, May 2017.

[42] F. de Assis Zampirolli and L. Filipe, "A fast CUDA-based implementation for the Euclidean distance transform," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2017, pp. 815–818.

[43] N. Audebert, A. Boulch, B. Le Saux, and S. Lefèvre, "Distance transform regression for spatially-aware deep semantic segmentation," *Comput. Vis. Image Understand.*, vol. 189, Dec. 2019, Art. no. 102809.

[44] M. Jaritz, R. D. Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and dense data with CNNs: Depth completion and semantic segmentation," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 52–60.

[45] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy LiDAR completion with RGB guidance and uncertainty," 2019, *arXiv:1902.05356*.

[46] X. Cheng, P. Wang, and R. Yang, "Depth estimation via affinity learned with convolutional spatial propagation network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 103–119.

[47] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from RGB and sparse sensing," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 167–182.

[48] L. Bai, Y. Zhao, M. Elhousni, and X. Huang, "DepthNet: Real-time LiDAR point cloud depth completion for autonomous vehicles," *IEEE Access*, vol. 8, pp. 227825–227833, 2020.

[49] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2446–2454.

[50] A. Li, Z. Yuan, Y. Ling, W. Chi, S. Zhang, and C. Zhang, "A multi-scale guided cascade hourglass network for depth completion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 32–40.

[51] L. He, G. Wang, and Z. Hu, "Learning depth from single images with deep neural network embedding focal length," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4676–4689, Sep. 2018.

[52] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.

[53] B.-U. Lee, H.-G. Jeon, S. Im, and I. S. Kweon, "Depth completion with deep geometry and context guidance," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3281–3287.

[54] C. Fu, C. Dong, C. Mertz, and J. M. Dolan, "Depth completion via inductive fusion of planar LIDAR and monocular camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct./Jan. 2021, pp. 10843–10848.

Yiming Zhao (Member, IEEE) received the B.S. degree math, physics, and systems science from Lanzhou University, Lanzhou, China, in 2014, and the M.S. degree from Beijing Normal University, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Worcester Polytechnic Institute, Worcester, MA, USA.

His Ph.D. works mainly focus on deep learning and robotic vision. He has published several papers on premier conferences and journals in computer vision and robotics.

Mahdi Elhousni (Student Member, IEEE) received the B.S. degree in computer science and the M.S. degree in embedded systems from the National School for Computer Science, Rabat, Morocco, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Worcester Polytechnic Institute, Worcester, MA, USA.

His main research interests are computer vision, deep learning, and SLAM.

Ziming Zhang (Member, IEEE) received the Ph.D. degree from Oxford Brookes University, Oxford, U.K., in 2013, under the supervision of Prof. Philip H. S. Torr (now with the University of Oxford).

He is currently an Assistant Professor with Worcester Polytechnic Institute, Worcester, MA, USA. Before joining WPI, he was a Research Scientist with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA, from 2016 to 2019. Prior to that, he was a Research Assistant Professor at Boston University, Boston, MA. His research areas lie in computer vision and machine learning, especially in object recognition/detection, data-efficient learning (e.g., zero-shot learning) and applications (e.g., person re-identification), deep learning, and optimization. His works have appeared in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, CVPR, ICCV, ECCV, and NIPS.

Dr. Zhang serves as a reviewer/PC member for top conferences (e.g., CVPR, ICCV, NIPS, ICML, ICLR, AAAI, AISTATS, and IJCAI) and journals (e.g., PAMI, IJCV, and JMLR). He won the Research and Development 100 Award 2018.

Xinming Huang (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Virginia Tech, Blacksburg, VA, USA, in 2001.

Then, he joined Bell Labs of Lucent Technologies, Whippany, NJ, USA, as a Technical Staff Member, conducting research on wireless IC design. Since 2006, he has been a Faculty Member of the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA, where he is currently the Dean's Excellence Professor and a Satin Distinguished Fellow. He has published more than 150 referred technical articles and served as an associate editor for several IEEE journals. His current research interests are in the area of circuits and systems for computer vision, machine learning, LiDAR processing, autonomous vehicles, robotics, and the IoT applications.