Implicit Point Function for LiDAR Super-Resolution in Autonomous Driving

Minseong Park, Haengseon Son, and Euntai Kim

***Abstract*—LiDAR super-resolution is a relatively new problem in which we seek to fill in the blanks between measured points when a low-resolution LiDAR is given, making a high-resolution LiDAR or even a resolution-free LiDAR. Recently, several research works have been reported regarding LiDAR super-resolution. However, most of the works on LiDAR super-resolution have the drawback that they first transform 3D LiDAR point cloud into 2D depth map and upsample the LiDAR output by applying the image super-resolution method, ignoring the 3D geometric information of the point cloud obtained from a LiDAR. To solve the above problem, we propose a new deep learning network named as implicit point function (IPF). The basic idea of IPF is that when we are given low-resolution point cloud and a query ray, we generate the 3D target point embeddings on the query ray using *on-the-ray positional embedding* and local features, preserving the 3D geometric information of the given point cloud. Then, we aggregate them into one target point via the attention mechanism. IPF enables us to learn continuous representation of 3D space from low-resolution LiDAR and upsample a small number of layers to any number that we want. Finally, our IPF is applied to large-scale synthetic dataset and real dataset, and its validity is demonstrated by comparing with the previous methods.**

***Index Terms*—Autonomous vehicle navigation, deep learning for visual perception, LiDAR.**

I. Introduction

LIDAR is receiving attention as a primary sensor in the field of mobile robotics and autonomous vehicles because it outputs the relatively precise distance to the obstacles compared with other sensors such as a camera. But the main drawback of the LiDAR is that the output from the LiDAR is quite sparse. In particular, the farther the distance is, the sparser the measured data points are, making the perception using the LiDAR very difficult when the obstacles are slightly far from the LiDAR. The evidence for this can be found in widely-used public LiDAR datasets [1], [2], [3]. Understandably, the high-end LiDAR with a large number of layers will mitigate the problem, but it will increase the cost significantly. Thus, LiDAR upsampling and super-resolution is receiving attention as the solution to the problem.

Over the last few years, several research works have been reported on the LiDAR super-resolution. Early works on LiDAR super-resolution were motivated by image super-resolution. Basically, they employed convolution-deconvolution frameworks and sought to learn an explicit function which directly maps a given low-resolution image to target resolution image of RGB (or depth) in image super-resolution [4], [5], [6], [7] (or LiDAR super-resolution [8], [9], [10]). However, these methods have the disadvantage that since the resolution of the output is fixed, new training is required when the target resolution is changed.

Recently, implicit function methods are being reported regarding the LiDAR super-resolution. The key difference between explicit [8], [9], [10] and implicit function methods is that the explicit function methods learn to predict the depth for the query image, whereas the implicit function methods learn to output not the depth but the *component* of the depth, which will be combined with other component to make the final depth prediction. As a pioneering work in image super-resolution, Local Implicit Image Function (LIIF) [11] was reported to learn local signal function from the query ray as the component of the final RGB value prediction. It outputs the final RGB value by applying the linear interpolation to the local RGB values predicted from each neighboring coordinates. Motivated by LIIF, Implicit LiDAR Network (ILN) [12] was reported as a pioneering work in LiDAR super-resolution for upsampling the LiDAR input. The key difference between them is that LIIF learns the local signals and aggregate them using linear interpolation, while ILN learns relative weight for measured signals and blend them using the learned weights via nonlinear interpolation. ILN showed almost the state-of-the-art (SOTA) performance in LiDAR super-resolution.

Despite the recent success in ILN, we believe that there is still a room that we can exploit in ILN to improve the performance. Specifically, 1) even though the ILN combines the measured points via nonlinear interpolation, the output is still weighted average (=convex combination) of the measured points. Thus, the output cannot be less than the minimum distance of the measured points and cannot be greater than the maximum distance of the measurement points, reducing the ability to cope

Manuscript received 18 May 2023; accepted 28 August 2023. Date of publication 11 September 2023; date of current version 19 September 2023. This letter was recommended for publication by Associate Editor Y. Jia and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) through the Korea Government (MSIT) under Grant 2021-0-00800 (Development of Driving Environment Data Transformation and Data Verification Technology for the Mutual Utilization of Self-driving Learning Data for Different Vehicles). *(Corresponding author: Euntai Kim.)*

Minseong Park and Euntai Kim are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea (e-mail: msp922@yonsei.ac.kr; etkim@yonsei.ac.kr).

Haengseon Son is with Korea Electronics Technology Institute, Seongnam 13509, South Korea (e-mail: hsson@keti.re.kr).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3313925, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3313925

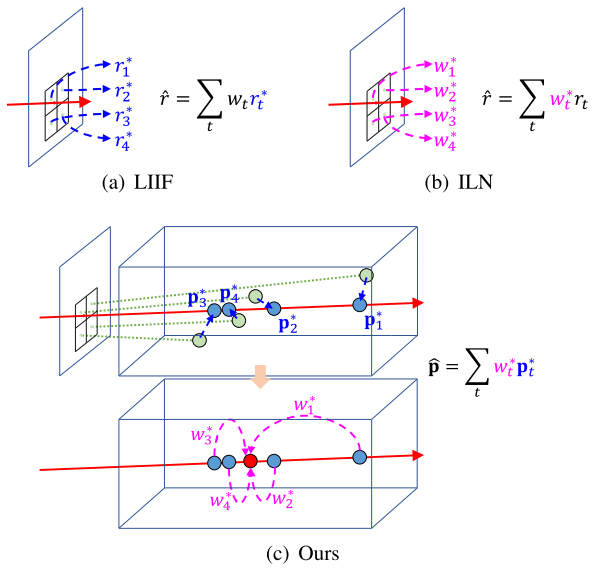

Fig. 1. Comparison between LIIF [11], ILN [12] and ours. * indicates the output predicted by the network. (a) LIIF learns the local depths; (b) ILN learns relative weights; (c) Our IPF learns 3D points and weights preserving 3D geometric information.

with the abrupt depth change. 2) The key characteristics of the output of a LiDAR is the 3D geometric information of the point cloud. However, ILN degenerates the 3D point cloud to 2D depth map and upsamples the depth map in a 2D-specific manner, not fully exploiting the 3D geometric relation between the measured points and the query ray. Specifically, ILN does not directly force the measured points to move to the 3D query ray but indirectly estimates the depth of the query ray, not imposing the constraint that the predicted point lies on the 3D query ray, as shown in the Fig. 1.

To solve the problem in ILN, we propose a new deep learning network named Implicit Point Function (IPF). The comparison between previous works, LIIF and ILN, and our IPF is summarized in Fig. 1. The key idea of IPF is that when low-resolution LiDAR point cloud and a query ray are given, we generate the target point embedding not in the 2D depth image but on the query ray in the 3D space using *on-the-ray positional embedding* and the local feature of each neighboring measured point, as shown in Fig. 1(c). Then we compute the relative weight for each target point embedding via attention mechanism and aggregate the target point embeddings using the computed weights. When we compute the weight for each target embedding, we use Transformer [13], [14] and it is motivated by ILN [12].

In this letter, we propose a novel implicit-based network for LiDAR super-resolution, which is named as Implicit Point Function (IPF). The technical contributions of our IPF are twofold. 1) When low-resolution point cloud and a query ray are given, we directly force the measured points to move onto the 3D query and impose the constraint that the predicted point lies on the 3D query ray, fully exploiting the geometric relationship between the measured points and the query ray. 2) We also propose on-the-ray positional embedding that enables IPF to capture global spatial context and the local geometric relationship between the measured point and the query ray in a 3D-specific manner. Using the above two technical contributions, our IPF achieves state-of-the-art performance on the synthetic large-scale CARLA dataset [12], [15] and the SemanticKITTI dataset [16].

II. Related Works

In the field of upsampling or super-resolution, there are three different yet similar tasks: (1) point cloud upsampling, (2) image super-resolution, and (3) LiDAR super-resolution. To tackle these tasks, there are two different solutions: (i) explicit methods and (ii) implicit function methods.

1) Explicit methods: Explicit methods involve training a network to generate virtual data that matches the given real data. Regarding the application of explicit methods to image super-resolution, several works have used convolutional neural networks [4], [5], [6], [7], [17] to generate high-resolution images.

Regarding the application of explicit methods to point cloud upsampling, PU-Net [18] can be considered as a pioneering work that directly upsamples low-resolution point clouds. Building on PU-Net, other works have employed explicit methods, such as PU-GAN [19], PU-GCN [20], LiUpNet [21], and PU-Dense [22], to upsample point clouds. PU-GAN [19] uses a generative adversarial network, while PU-GCN [20] employs a graph convolutional network and NodeShuffle module. LiUpNet [21] utilizes a transformer-based feature extractor and density-invariant feature consistency loss, and whereas PU-Dense [22] processes a diverse set of point clouds with variable input size by employing a 3D multi-scale architecture using sparse convolution.

Finally, regarding the application of explicit methods to LiDAR super-resolution, several works have been reported [8], [9], [10], and they are motivated by image super-resolution because the image SR methods can be applied to LiDAR SR by representing 3D LiDAR data on a 2D plane by replacing color or intensity with depth. These works use convolution-deconvolution frameworks to explicitly map low-resolution LiDAR data to high-resolution LiDAR data. However, they have the limitation of requiring retraining when the target resolution changes.

2) Implicit function methods: Implicit function methods involve training a network to learn a function that represents the sensor signals. Compared to explicit methods, the key advantages of implicit function methods are their flexibility, compressibility, and generality [23]. When applied to upsampling or super-resolution, the network in implicit function methods can be trained to learn the continuous representation of the input data. Thus, once a network is trained, it enables upsampling to any desired target resolution, including resolutions up to $\times 30$ or $\times 40$. In contrast, explicit function methods require separate training for each target resolution, which can be less effective. However, the drawback of implicit function methods is their limited interpretability, as their trained results are difficult for humans to understand.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

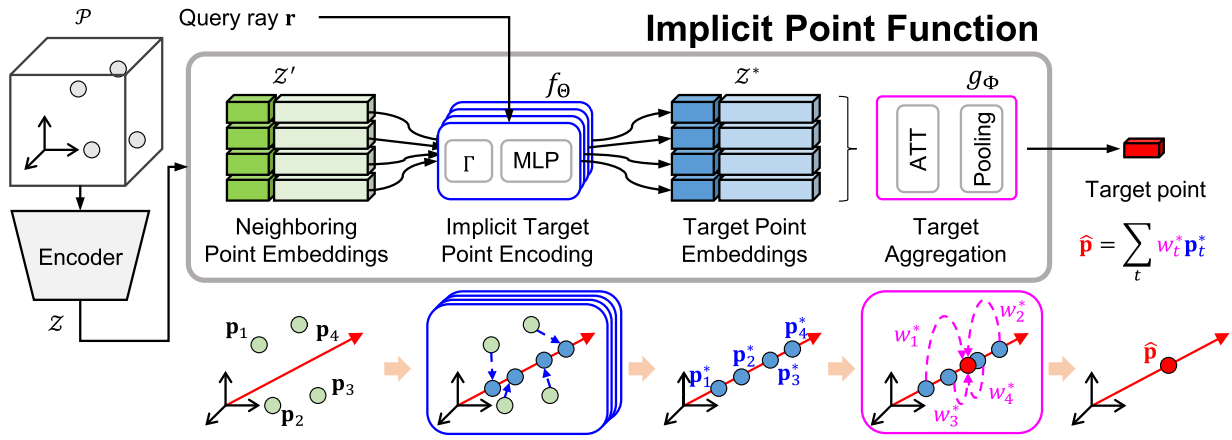

Fig. 2. Overview. When a set of low-resolution LiDAR point cloud is given, an encoder in the left of the figure extracts a set of point embeddings. From the set, four point embeddings $\mathcal{Z}' = \{[\mathbf{p}_t; \mathbf{z}_t]\}_{t=1}^{4}$ which are closest to the given query ray are selected. The implicit target point encoding function takes the four neighboring point embeddings $\mathcal{Z}' = \{[\mathbf{p}_t; \mathbf{z}_t]\}_{t=1}^{4}$ and the query ray as input, and returns target point embeddings $\mathcal{Z}^* = \{[\mathbf{p}_t^*; \mathbf{z}_t^*]\}_{t=1}^{4}$ which lie on the query ray. Finally, the target points are aggregated to one target points by target point aggregation function.

Regarding the application of implicit function methods to image super-resolution, Local Implicit Image Function (LIIF) [11] was reported as a pioneering work to learn local RGB function as an implicit function. In LIIF, the 2D image was upsampled regardless of the target resolution by learning a mapping from the low-resolution image to the component of the target image, rather than to the target image itself. Regarding the application of implicit function methods to LiDAR super-resolution Implicit LiDAR Network (ILN) [12] was reported as an implicit network. ILN [12] was motivated by [11]. The basic idea of [11] was to represent LiDAR data on a 2D plane by replacing color or intensity with depth and to leverage the techniques developed for "Image Super-Resolution" for up-sampling LiDAR data. However, ILN has the weakness point that it does not fully use the 3D geometric relation between the measured points and the query ray, and that is the point that we will exploit in this letter.

III. Implicit Point Function

A. Overview

In this letter, our goal is to fill in the blank pointed by a query ray $\mathbf{r} \in \mathbb{R}^3$ (= the 3D geometry of $\mathbf{r}$) based on a set of low-resolution LiDAR point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$. Here, the query ray $\mathbf{r}$ is assumed to start from the origin of the LiDAR. To achieve this goal, we propose a new deep learning network, Implicit Point Function (IPF), which enables us to predict the continuous representation of the given environment. The IPF is a function that takes a set of point embeddings $\mathcal{Z} = \{[\mathbf{p}_i; \mathbf{z}_i] \in \mathbb{R}^{3+D}\}_{i=1}^{N}$, and a query ray $\mathbf{r}$ as inputs, and returns a target point $\hat{\mathbf{p}} \in \mathbb{R}^3$ which lies on the query $\mathbf{r}$ as output, where $\mathbf{z}_i \in \mathbb{R}^D$ is the point feature; $\mathbf{p}_i \in \mathbb{R}^3$ is the 3D position of $\mathbf{z}_i$; D is the feature dimension; N is the number of points. Therefore, the IPF is defined as follows:

$$\hat{\mathbf{p}} = \mathrm{IPF}\left(\mathcal{Z}, \mathbf{r}\right). \tag{1}$$

As illustrated in Fig. 2, Our IPF is a composition of two functions: 1) implicit target point encoding function f_Θ and 2) target point aggregation function g_Φ. With these functions, $\mathrm{IPF} = g_\Phi \circ f_\Theta$ is reformulated as follows:

$$\mathrm{IPF}\left(\mathcal{Z}, \mathbf{r}\right) = g_\Phi\left(\bigcup_{[\mathbf{p}_t;\mathbf{z}_t]\in\mathcal{Z}'}\left\{f_\Theta\left(\begin{bmatrix}\mathbf{p}_t\\ \mathbf{z}_t\end{bmatrix}, \mathbf{r}\right)\right\}\right), \tag{2}$$

where $\mathcal{Z}' \subset \mathcal{Z}$ is a set which consists of four neighboring point embeddings $[\mathbf{p}_t; \mathbf{z}_t]$ around the query ray $\mathbf{r}$. Details of these functions, f_Θ and g_Φ, are described in Sections III-C and III-E, respectively.

B. Point Feature Encoding

We feed the set of low-resolution points $\mathcal{P}$ to a backbone network to obtain the set of point embeddings $\mathcal{Z}$. In this letter, we utilize the same backbone network EDSR [7], which has been previously employed in works such as [11] and [12], EDSR is known for its utilization of simplified ResNet blocks achieved by removing unnecessary batch normalization layers from the ResNet blocks of SRResNet [5]. The reason for removing batch normalization (BN) in EDSR is that 1) BN layers have the potential to limit model flexibility by normalizing the features, and 2) eliminating BN can reduce GPU consumption. It is worth noting that other backbone networks, such as PointNet++ [24], can also be used in this context.

C. Implicit Target Point Encoding Function

Different from the previous works [11], [12] which transform point cloud $\mathcal{P}$ into 2D depth map and upsample the depth image for each query, our network upsamples the LiDAR point cloud using the 3D geometric relationship between point and ray without projection onto the 2D image. Specifically, we directly project the measured points onto the 3D query and and impose the restriction that the predicted point lies on the 3D query ray, fully leveraging the geometric relationship between the measured points and the query ray. To realize the idea, we

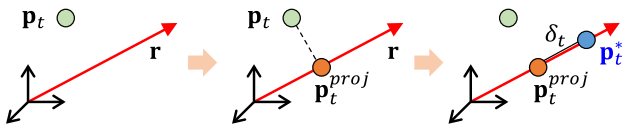

Fig. 3. Process of implicit target point encoding function. A neighboring point embedding $\mathbf{p}_t$ is orthogonally projected onto a query ray $\mathbf{r}$. Then the depth offset δ_t is added to the orthogonal projection $\mathbf{p}_t^{proj}$ and the feature offset $\Delta\mathbf{z}_t$ is added to the point embedding feature $\mathbf{z}_t$.

design the implicit target point encoding function f_Θ using a deep learning network with a set of learnable parameters Θ.

When a query ray $\mathbf{r}$ is given, we simply select four neighboring point embeddings $\mathcal{Z}' = \{[\mathbf{p}_t; \mathbf{z}_t]\}_{t=1}^4$ around the query ray $\mathbf{r}$ as in [11], [12]. Neighboring point embeddings $\mathcal{Z}'$ are selected so that pixel coordinates are closest to the query ray.

The goal of the implicit target point encoding function f_Θ is to generate the four target point embedding $\mathcal{Z}^* = \{[\mathbf{p}_t^*; \mathbf{z}_t^*]\}_{t=1}^4$ on the query ray $\mathbf{r}$ from the neighboring point embedding $\mathcal{Z}' = \{[\mathbf{p}_t; \mathbf{z}_t]\}_{t=1}^4$ and to capture 3D spatial context in the point cloud $\mathcal{P}$. A target point embedding To this end, we define the implicit target point encoding function $f_\Theta : \mathbb{R}^{3+D} \times \mathbb{R}^3 \mapsto \mathbb{R}^{3+D}$:

$$\begin{bmatrix} \mathbf{p}_t^* \\ \mathbf{z}_t^* \end{bmatrix} = f_\Theta \left(\begin{bmatrix} \mathbf{p}_t \\ \mathbf{z}_t \end{bmatrix}, \mathbf{r} \right), \tag{3}$$

which takes each neighboring point embedding $[\mathbf{p}_t; \mathbf{z}_t]$ and the query ray $\mathbf{r}$ as input, and returns the corresponding target point embedding $[\mathbf{p}_t^*; \mathbf{z}_t^*]$ as output. As shown in Fig. 3, the implicit target point encoding function f_Θ is realized by:

$$\begin{bmatrix} \delta_t \\ \Delta\mathbf{z}_t \end{bmatrix} = \text{MLP}\left(\begin{bmatrix} \Gamma(\mathbf{p}_t, \mathbf{r}) \\ \mathbf{z}_t \end{bmatrix} | \Theta \right), \tag{4}$$

$$\mathbf{p}_t^* = \mathbf{p}_t^{proj} + \delta_t \mathbf{r}, \tag{5}$$

$$\mathbf{z}_t^* = \mathbf{z}_t + \Delta\mathbf{z}_t, \tag{6}$$

where $\delta_t \in \mathbb{R}$ is the depth offset on the query ray $\mathbf{r}$ from $\mathbf{p}_t^{proj}$; $\Delta\mathbf{z}_t \in \mathbb{R}^D$ is the feature offset; Γ is the on-the-ray positional encoding function, which encodes the information about the relative position of each neighboring point $\mathbf{p}_t$ and a query ray $\mathbf{r}$; $\mathbf{p}_t^{proj} \in \mathbb{R}^3$ is the orthogonal projection point $\mathbf{p}_t$ onto the ray $\mathbf{r}$. The on-the-ray positional encoding function will be explained in the next subsection.

D. On-the-Ray Positional Encoding Function

The local geometric relationships among four neighboring points $\mathbf{p}_t$ on the ray $\mathbf{r}$ should be encoded into a target point embedding $[\mathbf{p}_t^*; \mathbf{z}_t^*]$ to perform super-resolution. To this end, we define the on-the-ray positional encoding function $\Gamma : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^{18L}$ that takes one of the four neighboring points $\mathbf{p}_t$ and $\mathbf{r}$ as input and returns the on-the-ray positional embedding as output, using high frequency positional encoding function $\gamma : \mathbb{R}^3 \mapsto \mathbb{R}^{6L}$ [25] which maps input to a higher dimension space, where L is a hyper parameter and $L = 10$ in this letter.

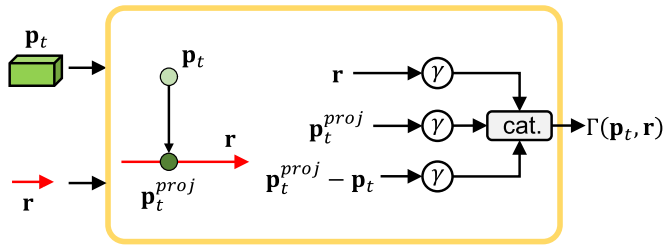

Fig. 4. On-the-ray positional encoding function. Three vectors query ray $\mathbf{r}$, projection $\mathbf{p}_t^{proj}$, and rejection $\mathbf{p}_t - \mathbf{p}_t^{proj}$ are mapped by γ to a higher dimension space, and then concatenated.

The on-the-ray positional encoding function Γ is defined by

$$\Gamma(\mathbf{p}_t, \mathbf{r}) = \begin{bmatrix} \gamma(\mathbf{r}) \\ \gamma\left(\mathbf{p}_t^{proj}\right) \\ \gamma\left(\mathbf{p}_t - \mathbf{p}_t^{proj}\right) \end{bmatrix} \in \mathbb{R}^{18L}, \tag{7}$$

where $\mathbf{p}_t^{proj}$ is the orthogonal projection of a neighboring point $\mathbf{p}_t$ onto the query ray $\mathbf{r}$ by

$$\mathbf{p}_t^{proj} = (\mathbf{p}_t \cdot \mathbf{r})\,\mathbf{r}. \tag{8}$$

The on-the-ray positional encoding function Γ is depicted in Fig. 4. The three vectors, query ray $\mathbf{r}$, projection $\mathbf{p}_t^{proj}$, and rejection $\mathbf{p}_t - \mathbf{p}_t^{proj}$, include the geometric information to capture the 3D relation among the ray and the neighboring point embedding $[\mathbf{p}_t; \mathbf{z}_t]$. The query ray $\mathbf{r}$ provides directional guide for super-resolution; the projection $\mathbf{p}_t^{proj}$ provides coarse information about the location of neighboring points on the ray $\mathbf{r}$; and the rejection $\mathbf{p}_t - \mathbf{p}_t^{proj}$ provides fine information between the query ray and the neighboring point. By combining the information from the three vectors (=$\mathbf{r}$,$\mathbf{p}_t^{proj}$,$\mathbf{p}_t - \mathbf{p}_t^{proj}$), we can capture the 3D geometric relationship between the query ray and each neighboring points in 3D space.

E. Target Point Aggregation Function

Given a set of target point embeddings each of which is obtained from the corresponding neighboring point, we aggregate the target point embeddings to determine the final target point. To achieve this goal, we define the target point aggregation function $g_\Phi : \mathbb{R}^{4\times(3+D)} \mapsto \mathbb{R}^3$:

$$\hat{\mathbf{p}} = g_\Phi(\mathcal{Z}^*), \tag{9}$$

which takes the set of generated target point embeddings $\mathcal{Z}^* = \{[\mathbf{p}_t^*; \mathbf{z}_t^*]\}_{t=1}^4$ as input and returns the final target point $\hat{\mathbf{p}}$ as output. The target point aggregation function is a composition of the weight prediction via attention mechanism (ATT) and the target point pooling. Then, the target aggregation function g_Φ is reformulated as follows:

$$\{w_t^*\}_{t=1}^4 = \text{ATT}(\mathcal{Z}^*), \tag{10}$$

$$\hat{\mathbf{p}} = \text{pooling}\left(\{w_t^*\}_{t=1}^4, \{\mathbf{p}_t^*\}_{t=1}^4\right). \tag{11}$$

1) Weight Prediction via Attention Mechanism: Since each target point embedding $[\mathbf{p}_t^*; \mathbf{z}_t^*]$ obtained from the original embedding $[\mathbf{p}_t; \mathbf{z}_t]$ only captures the limited local information about the target prediction, we have to aggregate the four target point embeddings to get reliable target prediction. Understandably, we have to focus on (or attend to) a certain target point embedding rather than the remaining three embeddings. To this end, we apply the self-attention mechanism to select a target point embedding $[\mathbf{p}_t^*; \mathbf{z}_t^*]$ that we have to focus on, or to predict the weights of the target points. The self-attention is realized using a Transformer [13], [14] to compute the weight of each target point embedding $[\mathbf{p}_t^*; \mathbf{z}_t^*]$ and it is similar to [12]. The only difference from [12] is that the on-the-ray positional embedding is used to capture the 3D information between the four target point embeddings. Combining the local structure obtained from four target point embeddings enable us to use enriched spatial context about local regions and to make final prediction.

2) Target Point Pooling: Given a set of predicted weights $\{w_t^*\}_{t=1}^4$ and the corresponding set of target points $\{\mathbf{p}_t^*\}_{t=1}^4$, we apply to get the final prediction

$$\hat{\mathbf{p}} = \sum_{t=1}^{4} w_t^* \mathbf{p}_t^*. \tag{12}$$

Here, it should be noted that unlike ILN in which the prediction is limited to the interval defined by the minimim and maximum values of $\{\mathbf{p}_t\}_{t=1}^4$, the range of values produced by our IPF is not constrained by the minimum and maximum distances to the measured points. The reason is that $\mathbf{p}_t^* = \mathbf{p}_t^{proj} + \delta_t \mathbf{r}$ and the offset parameter δ_t is computed from the MLP.

F. Loss

We aim at minimizing the error between predicted and ground-truth depth while ensuring that fine details are preserved, and the network remains robust to small outliers. To achieve this goal, IPF is trained in a supervised way using the L1 loss given by

$$\mathcal{L} = \sum_{i=1}^{M} \|\hat{\mathbf{p}}_i \cdot \mathbf{r}_i - d_i^{gt}\|, \tag{13}$$

where i is the index of query ray; M is the number of query rays used in training; d_i^{gt} is the ground-truth depth of query ray $\mathbf{r}_i$. In our letter, we chose to use L1 loss instead of L2 loss due to its robustness to outliers and lower sensitivity to high differences between predicted and actual values. If L2 loss is used, it will "square" the effect of high differences between predicted and actual values, thereby significantly being affected by some outliers.

IV. Experiments

A. Experimental Setting

1) Dataset: To validate our IPF, we use CARLA dataset as a benchmark dataset. The CARLA dataset contains eight scenarios recorded on urban roads by previous research [12]. Six scenarios (Town 01 - 06) are used for training and the remaining two scenarios (Town 07 and 10) are used for testing. Thus, we can say that the training and test sets are completely distinct and different from each other, except that both are captured on urban roads. The total numbers of scenes for train set and test set are 22,244 and 2,847, respectively. The LiDAR specification is that vertical Field of View (FoV) is $-15^\circ \sim 15^\circ$; horizontal FoV is $0^\circ \sim 360^\circ$; maximum distance is 80 m. To check the continuous representation and test the scalability of the competing methods, we train the networks to upsample 16×1024 to only one resolution 128×2048 but test the networks to upsample 16×1024 to three different resolutions: 64×1024, 128×2048, and 256×4096. The testing environment with a target resolution equal to or lower than the trained resolutions are called *in-distribution* (64×1024, 128×2048), and the testing environment with higher resolutions is called *out-of-distribution* (256×4096).

2) Evaluation Metric: We use the same evaluation metrics with the previous work [12]. For the 2D representation, we measure Mean Absolute Error (MAE) in the form of 2D depth map. For the 3D representation, we measure the Intersection of Union (IoU), Precision (Prec.), Recall (Rec.), and F1-Score (F1) in the form of voxel scene with a voxel size of 0.1 m.

B. Experimental Results

1) Quantitative Comparison Results: Our IPF is compared with LiDAR-SR [9], bilinear interpolation, LIIF [11], and ILN [12]. Since LiDAR-SR is an explicit function method, it directly learns the depth for each resolution and thus, it is trained separately for each resolution using the GT images. On the other hand, LIIF, ILN and IPF are implicit function frameworks, and they are trained only on 128×2048. We also compared our network with bilinear interpolation as a baseline.

Our IPF is compared with the previous works for both in-distribution and out-of-distributions in Table I. The results of LiDAR-SR, LIIF-LE, and ILN are taken from [12]. Additionally, we also trained the LIIF without LE (Local Ensemble). For both in-distribution (64×1024, 128×2048) and out-of-distribution test (256×4096) IPF achieves state-of-the-art results in almost all metrics and scales. In particular, IPF significantly outperforms all the previous methods in all 3D related metrics such as IoU, Precision, Recall, and F1-Score. Only in MAE for target resolution 256×4096, LIIF is slightly better than ours.

2) Qualitative Comparison Results: Fig. 5 shows the comparison of qualitative results on CARLA dataset. We marked the area that we have to focus on using magenta boxes. In the first and second rows, our IPF shows better results in detailed representation compared to other methods. Specifically, since ILN only blends the depth values of neighboring pixels, it cannot cope with the case in which the GT depth is not represented by the weighted sum (=convex combination) of the depth values of four neighboring points. For example, please see the ground point under the vehicle in first row or the empty space at the bottom of the structure in second row. On the other hand, our IPF recovers them better than ILN by capturing spatial context. In the third row, it is shown that our IPF is more robust to noise

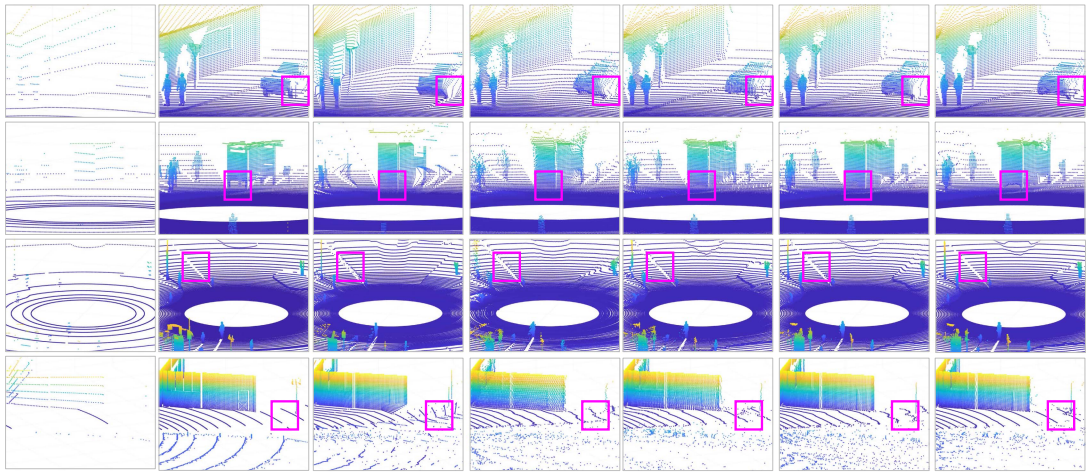

Fig. 5. Qualitative comparison results. We marked the area that we have to focus on using magenta boxes. Compared to other methods, our IPF shows better results in detailed representation, robustness to noise and reliability of ground point prediction.

TABLE I
LIDAR SUPER-RESOLUTION RESULTS ON CARLA DATASET. BOLD AND UNDERLINED INDICATE BEST AND SECOND-BEST PERFORMANCE FOR EACH METRIC, RESPECTIVELY. *CONVOLUTION-DECONVOLUTION FRAMEWORK THAT IS TRAINED WITH TARGET RESOLUTION GROUND TRUTH

Method	MAE [m]↓	IoU↑	Prec.↑	Rec.↑	F1↑
Target resolution: 64×1024					
LiDAR-SR [9]*	1.560	0.233	0.370	0.377	0.373
Bilinear	2.372	0.202	0.322	0.328	0.325
LIIF [11]	1.531	0.262	0.407	0.413	0.410
LIIF-LE [11]	1.558	0.258	0.403	0.409	0.406
ILN [12]	1.536	0.329	0.483	0.486	0.484
Ours	**1.527**	**0.346**	**0.505**	**0.506**	**0.505**
Target resolution: 128×2048					
LiDAR-SR [9]*	1.746	0.161	0.262	0.288	0.274
Bilinear	2.591	0.165	0.268	0.287	0.277
LIIF [11]	1.689	0.248	0.384	0.403	0.393
LIIF-LE [11]	1.714	0.236	0.372	0.388	0.379
ILN [12]	1.690	0.331	0.483	0.498	0.491
Ours	**1.676**	**0.374**	**0.533**	**0.541**	**0.537**
Target resolution: 256×4096					
LiDAR-SR [9]*	1.753	0.127	0.207	0.245	0.224
Bilinear	2.646	0.163	0.256	0.303	0.277
LIIF [11]	**1.737**	0.213	0.329	0.370	0.348
LIIF-LE [11]	1.923	0.158	0.221	0.356	0.272
ILN [12]	1.763	0.232	0.353	0.396	0.373
Ours	1.758	**0.246**	**0.386**	**0.398**	**0.392**

at the boundary of the occluded area than other methods. In the fourth row, IPF predicts the ground points more reliably than other methods, even when the low-resolution input point cloud has only a few points. This result shows that IPF has strength in capturing 3D spatial context over the previous methods.

3) Qualitative Results on Different Target Resolutions: Fig. 6 shows the qualitative results on the scalability of the competing networks. This figure demonstrates that IPF shows the robust upsampling results even when the target resolution changes. From the figure, it can be seen that IPF successfully recover the geometry of missing points from the low-resolution to high-resolution in both in-distribution and out-of-distribution tests.

TABLE II
LIDAR SUPER-RESOLUTION RESULTS ON SEMANTICKITTI DATASET. BOLD AND UNDERLINED INDICATE BEST AND SECOND-BEST PERFORMANCE FOR EACH METRIC, RESPECTIVELY. ALL METHODS WERE TESTED BY QUERYING ALL RAYS WHOSE GEOMETRY WAS OBSERVED

Method	MAE [m]↓	IoU↑	Prec.↑	Rec.↑	F1↑
Bilinear	2.463	0.123	0.207	0.225	0.216
LIIF [11]	1.941	0.136	0.234	0.240	0.237
LIIF-LE [11]	2.152	0.093	0.135	0.221	0.168
ILN [12]	2.040	**0.146**	0.254	**0.247**	**0.251**
Ours	**1.897**	**0.146**	**0.261**	0.241	**0.251**

C. Real Application

What really matters is the application to the real-world problem. Thus, we applied all the competing methods to SemanticKITTI [16] validation set (08). Specifically, we first down-sampled the point cloud from Velodyne HDL-64E and generated a 16 × 1024 depth map. Then, we trained all the competing models on the CARLA dataset during 400 epochs and tested them for SemanticKITTI by predicting the depth of all the rays whose measurements are available.

As in the CARLA dataset, in the SemanticKITTI dataset, IPF achieved state-of-the-art performance compared to previous methods. This implies that our proposed 3D-specific approach works with real dataset as well as synthetic dataset. The quantitative results are summarized in Table II and the qualitative result is illustrated in Fig. 7

V. IMPLEMENTATION DETAILS & ABLATION STUDIES

This section is provided in a supplementary file due to the page limit. The supplementary file is available at https://github.com/MSP922/IPF

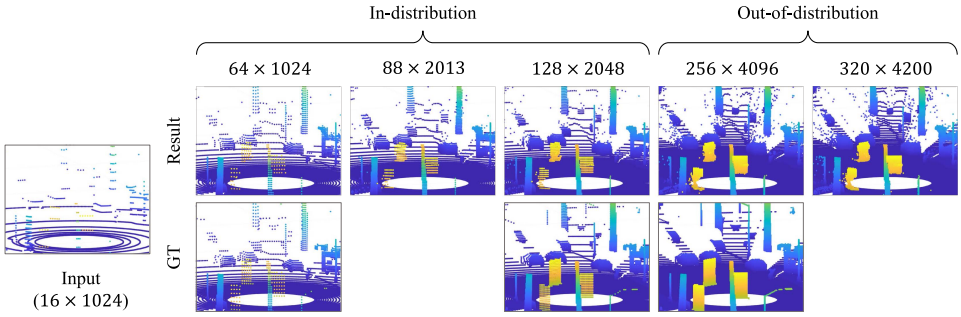

Fig. 6. Qualitative results on different target resolutions. Our IPF effectively restores the geometry of missing points in both in-distribution and out-of-distribution tests.

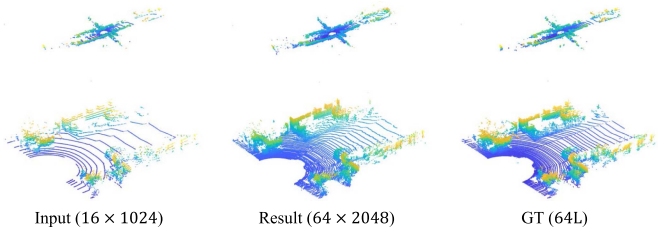

Fig. 7. Qualitative result on SemanticKITTI validation set. The input is 16 × 1024 depth map down-sampled from GT point cloud (64 L). Our IPF works with real dataset as well as synthetic dataset without training on real dataset.

VI. Conclusion

In this letter, we have proposed a novel implicit function for representing continuous 3D space and upsampling low-resolution LiDAR point cloud. Our function is a composition of two functions: implicit target point encoding function defined on the point embedding and the query ray and target point aggregation function for generated target points. We also propose a on-the-ray positional embedding for capturing global spatial context and local geometric relationship between the neighboring point and the query ray. Because of the limitation that there is no sensor that can measure continuous space the real world, our approach is only trained on synthetic dataset, but it can be extended to real world scenario.

References

[1] 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.*, 2012, pp. 3354–3361.
[2] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11618–11628.
[3] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2443–2451.
[4] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.
[5] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 105–114.
[6] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5835–5843.
[7] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140 .
[8] L. T. Triess, D. Peter, C. B. Rist, M. Enzweiler, and J. M. Zöllner, "CNN-based synthesis of realistic high-resolution LiDAR data," in *Proc. IEEE Intell. Veh. Symp.*, 2019, pp. 1512–1519.
[9] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, "Simulation-based LiDAR super-resolution for ground vehicles," *Robot. Auton. Syst.*, vol. 134, 2020, Art. no. 103647.
[10] G. Eskandar, S. Sudarsan, K. Guirguis, J. Palaniswamy, B. Somashekar, and B. Yang, "HALS: A height-aware LiDAR super-resolution framework for autonomous driving," 2022, *arXiv:2202.03901*.
[11] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8628–8638.
[12] Y. Kwon, M. Sung, and S. Yoon, "Implicit LiDAR network: LiDAR super-resolution via interpolation weight prediction," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 8424–8430.
[13] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[14] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.
[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.
[16] J. Behley et al., "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9296–9306.
[17] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[18] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.
[19] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7202–7211.
[20] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point cloud upsampling using graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11683–11692.
[21] T. Y. Chen, C. C. Hsiao, and C.-C. Huang, "Density-imbalance-eased LiDAR point cloud upsampling via feature consistency learning," *IEEE Trans. Intell. Veh.*, vol. 8, no. 4, pp. 2875–2887, Apr. 2023.
[22] A. Akhtar, Z. Li, G. V. d. Auwera, L. Li, and J. Chen, "PU-dense: Sparse tensor-based point cloud geometry upsampling," *IEEE Trans. Image Process.*, vol. 31, pp. 4133–4148, 2022.
[23] H. Kato et al., "Differentiable rendering: A survey," 2020, *arXiv:2006.12057*.
[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.
[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.