Supplementary Material: Implicit Point Function for LiDAR Super-Resolution

Minseong Park[1], Haengseon Son[2], Euntai Kim[1]

A. Implementation details

As in the previous works [11], [12], we use EDSR [24] as a feature encoder in IPF. When we apply the point cloud to EDSR, the input depth map is normalized from $[0, 100]$ to $[-1, 1]$ and the length of the output feature is set to $D = 64$. Each coordinate of the 3D points is also normalized from $[-100, 100]$ to $[-1, 1]$. The implicit target point encoding function f_Θ is realized by using a four layers MLP containing three hidden layers with ReLU activations and one output linear layer. The Transformer used in target aggregation function is implemented by stacking eight layers followed by two heads. The on-the-ray positional embedding used for Transformer consists of a linear layer and high frequency positional encoding function γ with $L = 10$. Our IPF is trained for 400 epochs with the batch size 16. An Adam optimizer is used with the initial learning rate 10^{-4} which decays by a rate of 0.5 every 200 epochs.

B. Ablation Studies

TABLE I
EFFECTIVENESS OF IMPLICIT TARGET POINT ENCODING FUNCTION. A TARGET AGGREGATION FUNCTION g_Φ IS REMOVED FOR FAIR COMPARISON BETWEEN LIIF AND IPF.

Target res.	Method	MAE [m]↓	IoU↑	F1↑
64×1024	LIIF [11]	**1.531**	0.262	0.410
	Ours (f_Θ)	1.741	**0.315**	**0.469**
128×2048	LIIF [11]	**1.689**	0.248	0.393
	Ours (f_Θ)	1.917	**0.363**	**0.526**
256×4096	LIIF [11]	**1.737**	0.213	0.348
	Ours (f_Θ)	1.988	**0.243**	**0.387**

1) Implicit target point encoding function: To see the effectiveness of the implicit target point encoding function f_Θ, we trained f_Θ while selecting only one nearest neighbor point $[\mathbf{p}_t; \mathbf{z}_t]$ instead of four points $\mathcal{Z}' = \{[\mathbf{p}_t; \mathbf{z}_t]\}_{t=1}^{4}$ for each query ray. Thus, we removed a target aggregation function g_Φ in this ablation study. Thus, IPF directly predicts the single nearest neighbor point from the query ray as the target point. For fair comparison, we also implemented LIIF so that it directly outputs the depth from the nearest neighbor pixel of the query ray. The comparison results are summarized in Table I. Our IPF shows lower performance than LIIF in 2D related metric MAE, but it outperforms LIIF in 3D related metrics such as IoU and F1-Score. This implies that the implicit target point encoding function f_Θ is not good for capturing the 2D context of the depth map, but shows good results in capturing the 3D spatial context which are essential in 3D points reconstruction.

[1]Minseong Park and Euntai Kim are with School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea {msp922, etkim}@yonsei.ac.kr
[2]Haengseon Son is with Korea Electronics Technology Institute hsson@keti.re.kr

TABLE II
QUANTITATIVE RESULT OF ON-THE-RAY POSITIONAL EMBEDDING ANALYSIS. THREE GEOMETRIC VECTORS, QUERY RAY (RAY), PROJECTION (PROJ.), AND REJECTION (REJ.) ARE USED.

Target res.	ray	proj.	rej.	MAE [m]↓	IoU↑	F1↑
64×1024	✓			1.690	0.218	0.351
		✓	✓	1.528	0.335	0.494
	✓	✓	✓	**1.527**	**0.346**	**0.505**
128×2048	✓			1.750	0.303	0.459
		✓	✓	1.683	0.355	0.517
	✓	✓	✓	**1.676**	**0.374**	**0.537**
256×4096	✓			2.032	0.185	0.309
		✓	✓	1.761	0.242	0.386
	✓	✓	✓	**1.758**	**0.246**	**0.392**

TABLE III
EFFECT OF L ON ACCURACY. L IS VARIED FROM 5 TO 30.

L	5	10	15	20	30
MAE [m]↓	1.755	**1.676**	1.700	1.780	4.339
IoU↑	0.325	**0.374**	0.355	0.248	0.022

2) On-the-ray positional embedding analysis: We conducted further experiments to determine which geometric information matters in on-the-ray positional embedding by changing the combinations of geometric vectors given in Eq. 7. Three geometric vectors are used, and they are query ray $\mathbf{r}$ (ray), projection $\mathbf{p}_t^{proj}$ (proj.), and rejection $\mathbf{p}_t - \mathbf{p}_t^{proj}$ (rej.). The quantitative results are summarized in Table II and the qualitative results are illustrated in Fig. 1. As shown in the table, when only query ray $\mathbf{r}$ is used in on-the-ray positional embedding, the performance is seriously degraded, and the reason is that local relationship between the query ray $\mathbf{r}$ and the neighboring point $\mathbf{p}_t$ is not considered. When all the three geometric vectors are used, the best performance is obtained, and this implies that the projection and rejection are key information for capturing the spatial context and local relationship between the neighboring point and the query ray.

Additionally, we conducted some experiments to investigate the effect of the L which is a hyper parameter for how high a dimension the geometric vector is mapped to. We varied

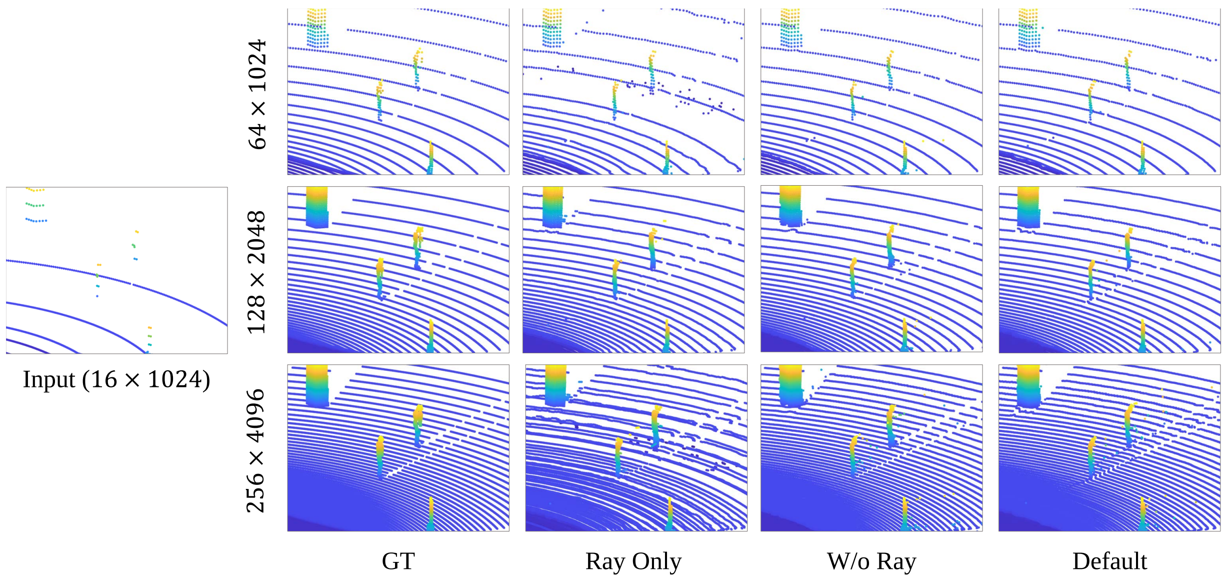

Fig. 1. Qualitative result of on-the-ray positional embedding analysis. Default setting is the combination of ray, projection, and rejection. Compared to Ray Only (2nd column), the performance of W/o Ray and Default (3rd and 4th columns) is improved. This improvement comes from the projection and rejection, which provide essential information for capturing the spatial context and local relationship between the neighboring point and the query ray.

the value of L and evaluated the resulting performance. The quantitative results are summarized in Table III. As shown in the table, increasing L from 5 to 10 improves performance, but increasing L from 10 to 20 does not, and $L = 30$ leads to a significant degradation in performance. This implies that mapping the geometric vectors to higher dimension space helps the network to capture the geometric information, but too high L makes it difficult the network to converge stably during training.

TABLE IV
TARGET POINT POOLING ANALYSIS.

Target res.	Pooling	MAE [m]↓	IoU↑	F1↑
64×1024	arg max	1.561	0.344	0.502
	Weighted Sum	**1.527**	**0.346**	**0.505**
128×2048	arg max	1.710	0.345	0.505
	Weighted Sum	**1.676**	**0.374**	**0.537**
256×4096	arg max	1.792	0.242	0.385
	Weighted Sum	**1.758**	**0.246**	**0.392**

3) Target point pooling analysis: We can consider two ways about how to use the weights obtained after attention to aggregate the four target embeddings, and they are $\arg\max$ and weighted sum. We tested our IPF using the two ways, and the results are summarized in Table IV. In our experiments, the weighted sum operator performs better than the $\arg\max$ operator. This implies that the single most promising target point does not capture the relationship between neighboring points as well as the combination of the four target points.

4) Analysis of various K: The number of neighboring point embeddings K is a key parameter that has a significant impact on both performance and time, and understanding its influence is crucial for optimizing IPF. The quantitative results are summarized in Table V. Here, $K = 1$ is f_Θ (without target aggregation function g_Θ). As shown in the table, $K = 4$ achieved best performance in 3D related metrics compared to all other settings. $K = 1$ is too local, so it cannot capture the spatial context. $K = 16$ is similar to $K = 4$ in 3D related metrics and outperforms all other settings in MAE. This implies that capturing information from many neighboring points can improve performance, but it is difficult to implement due to the amount of computation.

TABLE V
QUANTITATIVE RESULTS WITH VARIOUS K IN TARGET AGGREGATION FUNCTION FOR 128 × 2048. K IS VARIED FROM 1 TO 16. $K = 1$ CORRESPONDS TO THE CASE IN WHICH A TARGET AGGREGATION FUNCTION IS NOT USED.

K	MAE [m]↓	IoU↑	F1↑	Runtime [ms]
1	1.917	0.363	0.526	**46.0**
4	1.676	**0.374**	**0.537**	<u>160.6</u>
9	<u>1.665</u>	0.367	0.530	359.7
16	**1.658**	<u>0.373</u>	<u>0.536</u>	634.6

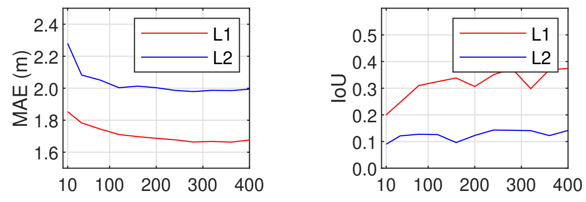

Fig. 2. Loss function analysis for 128 × 2048. The x-axis represents the training epochs. L1 loss (red line) works better than L2 loss (blue line) in terms of MAE and IoU.

5) Loss function analysis: We conducted an experiment for training IPF using different loss functions, L1 and L2. The results are illustrated in Fig. 2. We evaluated every 40 epochs. From the figure, our L1 loss works better than L2 loss in terms of MAE and IoU. The reason is clear from the fact that L1 and L2 losses serve different functions: If minimizing very seldom large outliers is our goal, we must choose L2, as L2 loss penalizes large outliers more due to the squaring. Yet, if minimizing regular noise is our goal, we have to select L1 since it penalizes small outliers more than L2 loss. Understandably, as our goal is to minimize regular noise over all the space, L1 works better than L2.

TABLE VI
QUANTITATIVE RESULTS ON NOISY INPUTS WITH DIFFERENT σ. GAUSSIAN NOISE WITH A MEAN OF 0 IS ADDED TO EACH PIXEL OF THE INPUT DEPTH MAP WHILE VARYING STANDARD DEVIATION σ FROM 0 TO 1.0 M.

	σ [m]	0	0.1	0.2	0.5	1.0
MAE [m]↓	ILN [12]	1.690	1.708	1.725	1.780	1.873
	Ours	**1.676**	**1.691**	**1.710**	**1.767**	**1.871**
IoU↑	ILN [12]	0.331	0.240	0.186	0.102	0.074
	Ours	**0.374**	**0.266**	**0.204**	**0.110**	**0.076**

6) Robustness to noise and outliers: We conduct some additional experiments to evaluate the robustness of the competing method to noise and outliers. We add Gaussian noise with a mean of 0 to each pixel of the input depth map while varying standard deviation σ from 0 to 1.0 m, as shown in Table VI. The Gaussian noise is added during the inference with a fixed random seed for the same input. In Table VI, $\sigma = 0$ corresponds to the case of original ILN and IPF. From the table, we can see that our IPF is just as impacted by additional noise as ILN is, but IPF consistently surpasses ILN in terms of MAE and IoU, regardless of the level of noise.

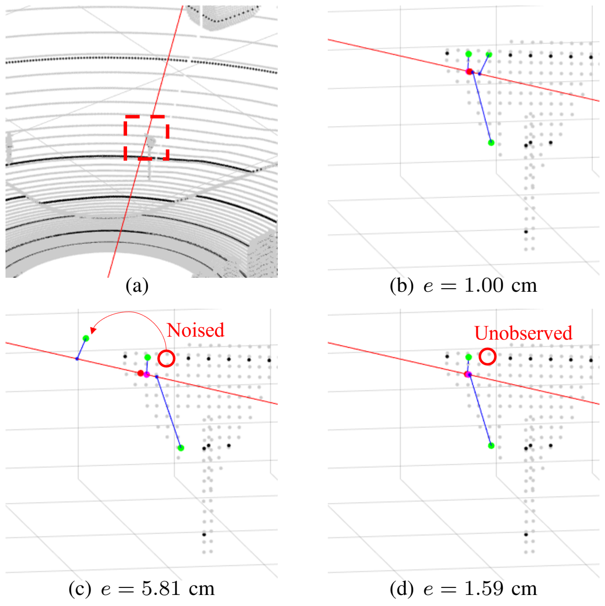

Fig. 3. Robustness to noise and outliers. Black points are the input low-resolution points and gray points are the GT high-resolution points. Red lines represent the query ray. Blue line shows predicting the target points (blue) from the neighboring points (green). The final prediction and ground truth is represented as red and magenta, respectively. (a) Scene and query ray; (b) Result from clean input; (c) Result from Gaussian noise-added neighboring points ($\sigma = 0.2$ m); (d) A neighboring point is unobserved.

Furthermore, we conduct an additional experiment for the case that the neighboring point is unobserved. We remove one of the neighboring points. The qualitative results for all cases (clean, noised, unobserved) are illustrated in Fig. 3. In Fig. 3(c), We can see that although one target point is predicted far away from the GT because of the noised neighboring point, it is aggregated with low weight and resulted in a final prediction that was close to the GT. In Fig. 3(d) we can see that the final prediction is not significantly affected even though one neighboring point is not observed.

TABLE VII
COMPARISON RESULT WITH PU-DENSE [22] FOR 64×1024. PU-DENSE IS TRAINED WITH 64×1024 AND THE RESULT IS FOR $\times 4$.

Method	IoU↑	Prec.↑	Rec.↑	F1↑
PU-Dense [22]	0.069	0.224	0.089	0.126
Ours	**0.346**	**0.505**	**0.506**	**0.505**

7) Comparison with point cloud upsampling: We conducted an experiment to compare our IPF with a direct point cloud upsampling method. The results of the comparison are summarized in Table VII. From the table, it is evident that IPF outperforms PU-Dense significantly. This table clearly demonstrates that general point cloud upsampling methods do not perform well when applied to LiDAR super-resolution. Conversely, our IPF effectively upsamples the point cloud. This is likely due to the fact that PU-Dense solely increases the point cloud density without considering the query ray of the target LiDAR.

TABLE VIII
ANALYSIS OF HYPER PARAMETER USED DURING TRAINING. WE REPORT THE RESULTS FOR 128×2048 ABOUT THE LEARNING RATE (LR) AND THE LEARNING RATE DECAY STEP SIZE (DS).

LR	DS	MAE [m]↓	IoU↑	Prec.↑	Rec.↑	F1↑
1e-3	200	1.732	0.341	0.496	0.508	0.502
5e-4	200	1.687	0.362	0.519	0.531	0.525
1e-4	100	1.686	**0.380**	**0.538**	**0.547**	**0.542**
1e-4	200	**1.676**	0.374	0.533	0.541	0.537

8) Hyper parameters for training: We consider the effects of the learning rate (LR) and decay step size (DS) used in the training. In the experiments, we varied LR from 0.0001 to 0.001 and DS from 100 to 200, respectively. The results are summarized in Table VIII. From the table, we can see that our IPF trained with learning rate 1e-4 and learning rate decay step size 200 achieved the best performance in terms of MAE.

TABLE IX
COMPUTATIONAL COMPLEXITY ANALYSIS FOR 128×2048.

Method	MAE [m]↓	IoU↑	GFLOPs (/frame)	Runtime [ms]
LIIF [11]	<u>1.689</u>	0.248	<u>151.953</u>	**17.6**
LIIF-LE [11]	1.714	0.236	488.169	51.3
ILN [12]	1.690	0.331	**93.635**	47.1
Ours (f_Θ)	1.917	<u>0.363</u>	179.888	<u>46.0</u>
Ours ($g_\Phi \circ f_\Theta$)	**1.676**	**0.374**	600.213	160.6

9) Computational complexity: The computational complexity analysis of competing methods is summarized in Table IX. In terms of runtime, the proposed IPF (f_Θ) works at the same level as LIIF-LE and ILN and has a performance improvement of at least about 10% over both in IoU which is really matter for 3D point cloud. In terms of GFLOPs, IPF (f_Θ) is similar to LIIF with a performance improvement of about 46% over it. IPF ($g_\Phi \circ f_\Theta$) has the biggest computational complexity, with performance improvement of at least about 13% over competing methods, but it works faster than 5 Hz.