DiffPCI: Large Motion Point Cloud frame Interpolation with Diffusion Model

Supplementary Material

Here, we provide additional information to supplement the main paper. Firstly, we provide a detailed derivation of the interpolation-specific variational lower bound in §A.1. Secondly, we present more details on the network structures of the Global Matching module, the Local-Global Attention Weighting module, and the Point Fusion module in §A.2, §A.3, and §A.4. Then, the additional dataset details information is described in §B. We conduct additional ablation studies in §C. Moreover, we provide additional qualitative results in §D.

A. Methods Details

A.1. Derivation of the interpolation-specific variational lower bound

In this section, we derive an interpolation-specific variational lower bound $\mathcal{L}_{intp}$, the optimization objective of DiffPCI. The main paper mentions that DiffPCI consists of a forward interpolation diffusion process and a reverse interpolation denoising process. The forward interpolation diffusion process is defined by a Markov chain that gradually adds noise to a ground truth intermediate point cloud frame $\mathbf{P}^n$ using a pre-defined noise schedule $\{\beta_t\}_{t=1}^T$ in T steps, with conditional probabilities, as follows:

$$q(\mathbf{P}^n_{1:T}|\mathbf{P}^n) := \prod_{t=1}^{T} q(\mathbf{P}^n_t|\mathbf{P}^n_{t-1}), \tag{S.1}$$

$$q(\mathbf{P}^n_t|\mathbf{P}^n_{t-1}) := \mathcal{N}(\mathbf{P}^n_t; \sqrt{1-\beta_t}\mathbf{P}^n_{t-1}, \beta_t\mathbf{I}). \tag{S.2}$$

Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, according to the conditional independence property of Markov chain, one can sample from the forward interpolation diffusion process at an arbitrary time step t with:

$$q(\mathbf{P}^n_t|\mathbf{P}^n) = \mathcal{N}(\mathbf{P}^n_t; \sqrt{\bar{\alpha}_t}\mathbf{P}^n, (1-\bar{\alpha}_t)\mathbf{I}). \tag{S.3}$$

To sample $\mathbf{P}^n_t$ from this distribution in practice, we can use a reparameterization:

$$\mathbf{P}^n_t(\mathbf{P}^n, \epsilon) = \sqrt{\bar{\alpha}_t}\mathbf{P}^n + \sqrt{1-\bar{\alpha}_t}\epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{S.4}$$

Here the noise schedule $\{\beta_t\}_{t=1}^T$ is the designed such that $\bar{\alpha}_t \approx 0$ and $q(\mathbf{P}^n_T|\mathbf{P}^n) \approx \mathcal{N}(\mathbf{P}^n_T; \mathbf{0}, \mathbf{I})$. That is, as the forward interpolation diffusion process ends, the last state of the ground truth point cloud frame becomes close to pure Gaussian noise.

Given the forward interpolation diffusion process, one could generate new samples by starting from pure Gaussian noise and sampling from the reverse conditionals $q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t)$, $\mathbf{P}^0$, and $\mathbf{P}^1$. As $q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t)$ is intractable, so one can use a θ-parameterized Gaussian distribution

$$p_\theta(\mathbf{P}^n_{0:T}) := p(\mathbf{P}^n_T) \prod_{t=1}^{T} p_\theta(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^0, \mathbf{P}^1), \tag{S.5}$$

$$p_\theta(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^0, \mathbf{P}^1) := \mathcal{N}(\mathbf{P}^n_{t-1}; \mu_\theta(\mathbf{P}^n_t, \mathbf{P}^0, \mathbf{P}^1, n), \tilde{\beta}_t\mathbf{I}), \tag{S.6}$$

to approximate the reverse Markov chain (provided that β_t is sufficiently small in each forward step [4]). Here μ_θ denotes a deep neural network, and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Then the interpolation-specific variational lower bound is derived as follows:

$$\begin{aligned} &\mathbb{E}\left[-\log p_\theta(\mathbf{P}^n)\right] \\ &\leq \mathbb{E}\left[-\log\left(\frac{p_\theta(\mathbf{P}^n_{0:T} \mid \mathbf{P}^0, \mathbf{P}^1)}{q(\mathbf{P}^n_{1:T} \mid \mathbf{P}^n)}\right)\right] =: \mathcal{L}_{intp}, \end{aligned} \tag{S.7}$$

and training can be done by minimizing $\mathcal{L}_t(\theta)$, it can be decomposed [1, 3, 5] as

$$\begin{aligned} \mathcal{L}_{intp} = \mathbb{E}_q\Big[&\underbrace{D_{KL}(q(\mathbf{P}^n_T|\mathbf{P}^n)||p_\theta(\mathbf{P}^n_T))}_{L_T} \\ &+ \sum_{t>1}^{T} \underbrace{D_{KL}(q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^n)||p_\theta(\mathbf{P}^n_{t-1}|\mathbf{P}^0, \mathbf{P}^1))}_{L_{t-1}} \\ &- \underbrace{\log p_\theta(\mathbf{P}^n|\mathbf{P}^n_1)}_{L_0}\Big]. \end{aligned} \tag{S.8}$$

The L_T in Equ. S.8 can be ignored because the prior $p_\theta(\mathbf{P}^n_T)$ can be set to a standard normal distribution. The *reconstruction term* can be approximated and optimized using a Monte Carlo estimate [3], leaving the L_{t-1} as the main focus for learning the reverse interpolation denoising process. The $q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^n)$ in the L_{t-1} is tractable and it can be derived that

$$q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^n) = \mathcal{N}(\mathbf{P}^n_{t-1}; \tilde{\mu}_t(\mathbf{P}^n_t, \mathbf{P}^n), \tilde{\beta}_t\mathbf{I}), \tag{S.9}$$

Here,

$$\tilde{\mu}_t(\mathbf{P}^n_t, \mathbf{P}^n) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{P}^n + \frac{\sqrt{\alpha}_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{P}^n_t, \tag{S.10}$$

As both $q(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t, \mathbf{P}^n)$ and $p_\theta(\mathbf{P}^n_{t-1}|\mathbf{P}^n_t)$ are Gaussian distributions, the KL-divergence in the L_{t-1} in Equ. S.8

*Corresponding author.

takes the form

$$\mathcal{L}_{t-1} = \mathbb{E}_q \left[\frac{1}{2\bar{\beta}_t} || \underbrace{\tilde{\mu}(\mathbf{P}_t^n, \mathbf{P}^n)}_{①} - \underbrace{\mu_\theta(\mathbf{P}_t^n, \mathbf{P}^0, \mathbf{P}^1, n)}_{②} ||^2 \right]. \tag{S.11}$$

It is noted that plugging Equ. S.4 into Equ. S.11, the ① can be written as

$$\tilde{\mu}(\mathbf{P}_t^n, \mathbf{P}^n) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\mathbf{P}_t^n + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\mathbf{P}^n}{1-\bar{\alpha}_t}, \tag{S.12}$$

As ② also conditions on $\mathbf{P}_t^n$, we can match ① closely by setting it to the following form:

$$\mu_\theta(\mathbf{P}_t^n, \mathbf{P}^0, \mathbf{P}^1, n) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\mathbf{P}_t^n}{1-\bar{\alpha}_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t) f_\theta(\mathbf{P}_t^n, \mathbf{P}^0, \mathbf{P}^1, n)}{1-\bar{\alpha}_t}, \tag{S.13}$$

where $f_\theta(\mathbf{P}_t^n, \mathbf{P}^0, \mathbf{P}^1, n)$ is parameterized by a deep neural network that seeks to predict $\mathbf{P}^n$ from noisy point cloud $\mathbf{P}_t^n$ and time index t. Then, the optimization problem simplifies to:

$$\mathcal{L}_{intp} = \text{loss}(f_\theta(\mathbf{P}_t^n, \mathbf{P}^0, \mathbf{P}^1, n), \mathbf{P}^n). \tag{S.14}$$

where we employ the Chamfer distance as the loss function (see Sec. 3.3 in the main paper).

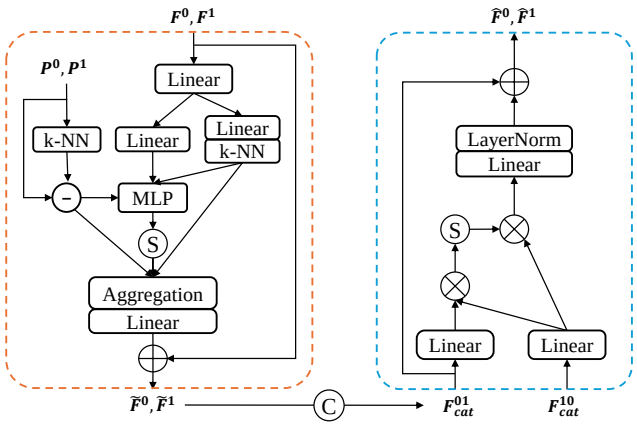
A.2. Global Matching module

This module is introduced from our baseline GMSF [7] for calculating global correlation and estimating initial scene flow. First, the global cross-correspondence matrix between the input frames $\mathbf{P}^0, \mathbf{P}^1$ is given by multiplying the feature matrices $\mathbf{F}^0, \mathbf{F}^1$ and then normalizing over the second dimension with softmax to get the matching confidence. Secondly, the self-correspondence matrix of $\mathbf{P}^0$ is generated using a self-attention mechanism. The estimated matching point cloud is computed as a weighted average of the 3D coordinates based on the matching confidence. Third, the initial scene flow is obtained by subtracting the estimated matching point cloud from $\mathbf{P}^0$.

A.3. Local-Global Attention Weighting module

The local-global attention weighting module applies both local and global attention weighting to the secondary encoded features of the full-scale module, enhancing the learning of more prominent and engaging features.

As shown in Fig. S1 (a), local information is initially integrated into a compact region using k-nearest neighbors (k-NN) and subsequently derived through aggregation, as follows:



(a) Local attention weighting (b) Global attention weighting

Figure S1. **The illustration of Local-global attention weighting module.** (a) Local attention weighting (Orange dashed box); (b) Global attention weighting (Blue dashed box). The inputs are two contiguous frames $\mathbf{P}^0, \mathbf{P}^1$ and their features $\mathbf{F}^0, \mathbf{F}^1$ encoded once by the full-scale module respectively. Attention features $\hat{\mathbf{F}}^0, \hat{\mathbf{F}}^1$ are output after attention weighting module. Here $\textcircled{S}, \oplus, \otimes$ and $\ominus$ denote the softmax function, multiplication, addition element-wise, and subtraction element-wise operations, respectively.

$$\tilde{\mathbf{F}}^0 = MLP(\psi_q(\mathbf{F2}^0) - \psi_k(\mathbf{F2}^0) + \delta) \odot (\psi_v(\mathbf{F2}^0) + \delta),$$
$$\tilde{\mathbf{F}}^1 = MLP(\psi_q(\mathbf{F2}^1) - \psi_k(\mathbf{F2}^1) + \delta) \odot (\psi_v(\mathbf{F2}^1) + \delta), \tag{S.15}$$

The input features are initially processed through linear layers ψ_q, ψ_k, and ψ_v to produce the query, key, and value representations. $\mathbf{F2}$ is the feature obtained from the secondary encoding of the full-scale encoder. The symbol δ denotes the relative position embedding.

Global information is gathered through two Global attention weighting layers (Fig. S1 (b)): one utilizing self-attention and the other employing cross-attention, both of which have similar structures. The self-attention is formulated as

$$\mathbf{F}_{self}^{01} = \langle \phi_q(\mathbf{F}_{cat}^{01}), \phi_k(\mathbf{F}_{cat}^{01}) \rangle \, \phi_v(\mathbf{F}_{cat}^{01}), \tag{S.16}$$

The cross-attention is given as

$$\hat{\mathbf{F}}^0 = TC[\langle \phi_q(\mathbf{F}_{self}^{01}), \phi_k(\mathbf{F}_{self}^{01}) \rangle \, \phi_v(\mathbf{F}_{self}^{01})]. \tag{S.17}$$

Here we take generating $\hat{\mathbf{F}}^0$ as an example, where ϕ_q, ϕ_k, and ϕ_v are the linear layers to generate query, key, and value. $\mathbf{F}_{cat}^{01}$ is the cascading in the batch dimension of $\tilde{\mathbf{F}}^0$ and $\tilde{\mathbf{F}}^1$. TC is a tensor chunking operation.

A.4. Point fusion module

See Fig. S2 for details.

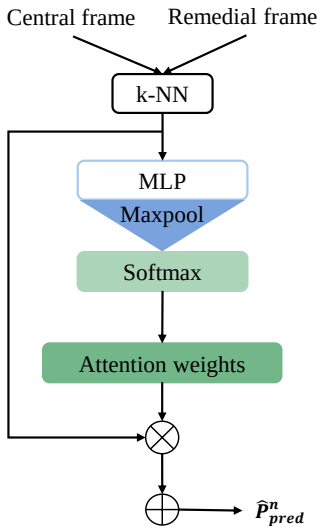


Figure S2. **The illustration of Point Fusion module.** The center frame and remedial frame obtained from the adaptive selection module are processed by the point fusion module, which utilizes k-nearest neighbors (k-NN), a multi-layer perceptron (MLP), max pooling, and softmax operations to produce the final predicted time n point cloud frame $\hat{\mathbf{P}}^{\mathbf{n}}_{\mathbf{pred}}$.

B. Additional dataset details

As described in Sec. 4.1 of the main paper, we adhere to the dataset segmentation and preprocessing methods outlined in NeuralPCI [8]. Specifically, the KITTI odometry dataset comprises 22 LiDAR point cloud sequences, with 11 sequences containing ground truth (00 to 10). We use sequences 00 to 06 for training and sequences 09 to 10 for testing. The Argoverse 2 sensor dataset includes 1,000 scenarios, averaging 150 LiDAR sweeps per scenario, whereas the Nuscenes dataset consists of 1,000 driving scenes, each containing approximately 400 LiDAR frames. For both datasets, we utilize the top 700 scenes for training and the scenes numbered 851 to 1000 for testing. The KITTI odometry and Argoverse 2 sensor data are point cloud frames captured at a frame rate of 10Hz, while the Nuscenes data are captured at 20Hz. To ensure consistency across datasets, the Nuscenes data are downsampled to a frequency of 10Hz, which inherently leads to increased motion between the input frames.

C. Additional Ablation Studies

C.1. Number of Time Steps.

We investigate the influence of the different numbers of time steps during training and sampling on the results of the Nuscenes dataset in Table. S1 and S2. Although increasing the number of denoising steps can produce cleaner intermediate frames, we observe that in real-world autonomous driving PCI tasks, the GT frames themselves often contain noise. As a result, over-denoising (with too many denoising steps) may yield overly clean outputs that deviate from the noisy GT, leading to degraded scores. By contrast, the slightly noisy frames generated with fewer denoising steps (as in our method) better align with the GT and achieve higher scores.

Table S1. **The number of time steps during training on three automatic driving datasets.** The best results are in **bold**.

Dataset	The number of steps	Metrics	
		CD↓	EMD↓
Nuscenes	5	1.17	182.83
	20	**0.84**	**165.04**
	100	0.99	172.61
	200	1.15	186.67
Argoverse 2 sensor	5	0.81	61.47
	20	**0.67**	**56.79**
	100	0.73	62.08
	200	0.86	68.43
KITTI odometry	5	0.58	58.24
	20	**0.50**	**53.67**
	100	0.55	58.51
	200	0.59	60.35

Table S2. **The number of time steps during inferencing on three automatic driving datasets, where the training time step is set to 20.** The best results are in **bold**.

Dataset	The number of steps	Metrics	
		CD↓	EMD↓
Nuscenes	1	0.85	165.96
	2	**0.84**	165.72
	3	**0.84**	**165.04**
	4	0.86	165.97
	5	0.86	165.98
	6	0.86	166.01
	7	0.86	166.04
Argoverse 2 sensor	1	**0.67**	57.06
	2	**0.67**	56.98
	3	**0.67**	**56.79**
	4	0.68	57.16
	5	0.68	57.22
	6	0.69	57.25
	7	0.70	57.34
KITTI odometry	1	0.50	53.99
	2	**0.50**	53.70
	3	**0.50**	**53.67**
	4	**0.50**	**53.67**
	5	0.50	53.94
	6	0.50	53.99
	7	0.50	54.00

Table S3. **Runtime and Parameter Comparison.**

Methods	Parameter(M)	Runtime(ms)	CD↓ / EMD↓
PointINet [2]	0.002	1458	1.55/190.54
NeuralPCI [8]	1.848	60822	1.15/179.11
FastPCI [6]	5.793	109	1.11/173.64
DiffPCI(ours)	4.124	2815	0.84/165.04

C.2. Runtime and Parameter Comparison.

We report runtime (ms per frame) during inference on a single NVIDIA GeForce RTX 3090 GPU, the Number of parameters (M), and the quantitative comparison of the Nuscenes dataset in Table. S3. It is evident that while we incur a certain inference time cost for a relatively modest number of parameters, the interpolation performance gains we achieve are significantly superior.

D. Additional Visual Results

Fig. S3, S4, and S5 show more of our qualitative results. Among them, yellow, blue, and red correspond to three interpolated intermediate frames.

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020. 1

[2] Fan Lu, Guang Chen, Sanqing Qu, Zhijun Li, Yinlong Liu, and Alois Knoll. Pointinet: Point cloud frame interpolation network. In *AAAI*, page 2251–2259, 2021. 4

[3] Calvin Luo. Understanding diffusion models: A unified perspective, 2022. 1

[4] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, Lille, France, 2015. PMLR. 1

[5] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, 56(4), 2023. 1

[6] Tianyu Zhang, Guocheng Qian, Jin Xie, and Yang Jian. Fastpci: Motion-structure guided fast point cloud frame interpolation. In *ECCV*, 2024. 4

[7] Yushan Zhang, Johan Edstedt, Bastian Wandt, Per-Erik Forssen, Maria Magnusson, and Michael Felsberg. Gmsf: Global matching scene flow. In *Advances in Neural Information Processing Systems*, pages 64415–64427. Curran Associates, Inc., 2023. 2

[8] Zehan Zheng, Danni Wu, Ruisi Lu, Fan Lu, Guang Chen, and Changjun Jiang. Neuralpci: Spatio-temporal neural field for 3d point cloud multi-frame non-linear interpolation. In *CVPR*, 2023. 3, 4

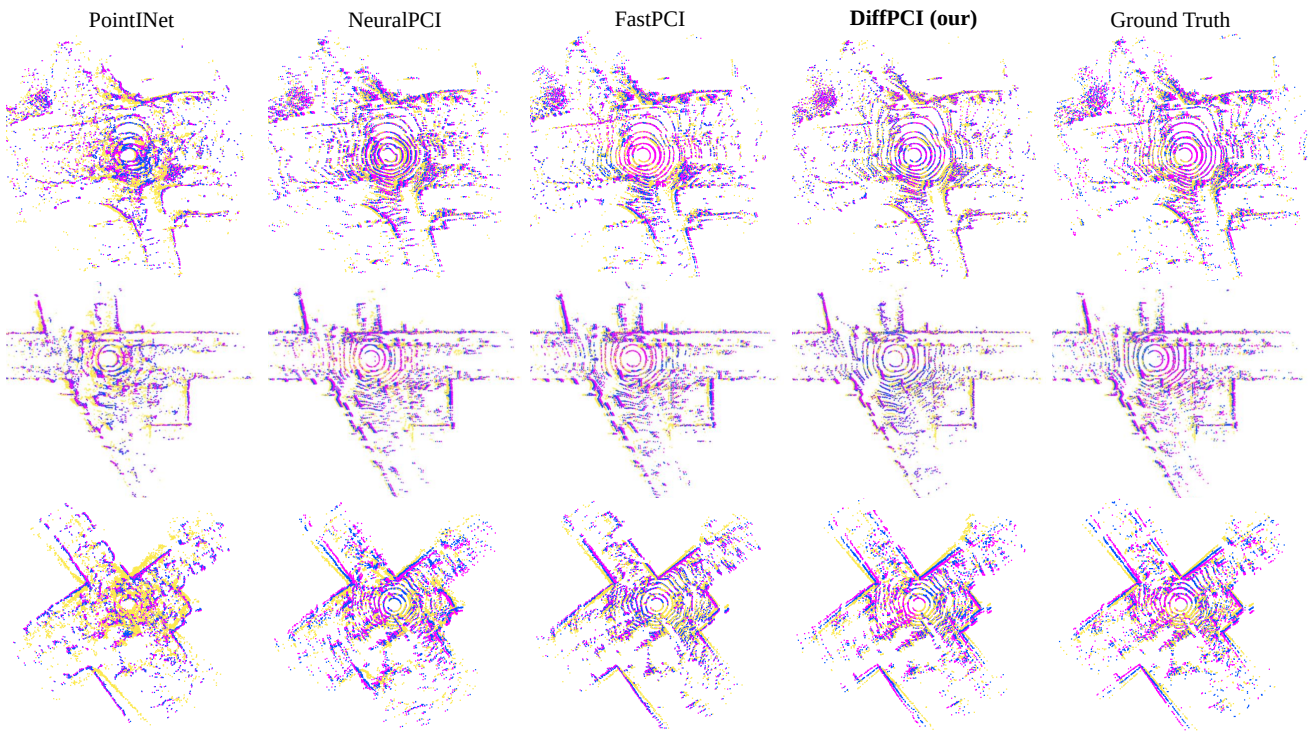


Figure S3. **More qualitative comparison on Nuscenes dataset.**

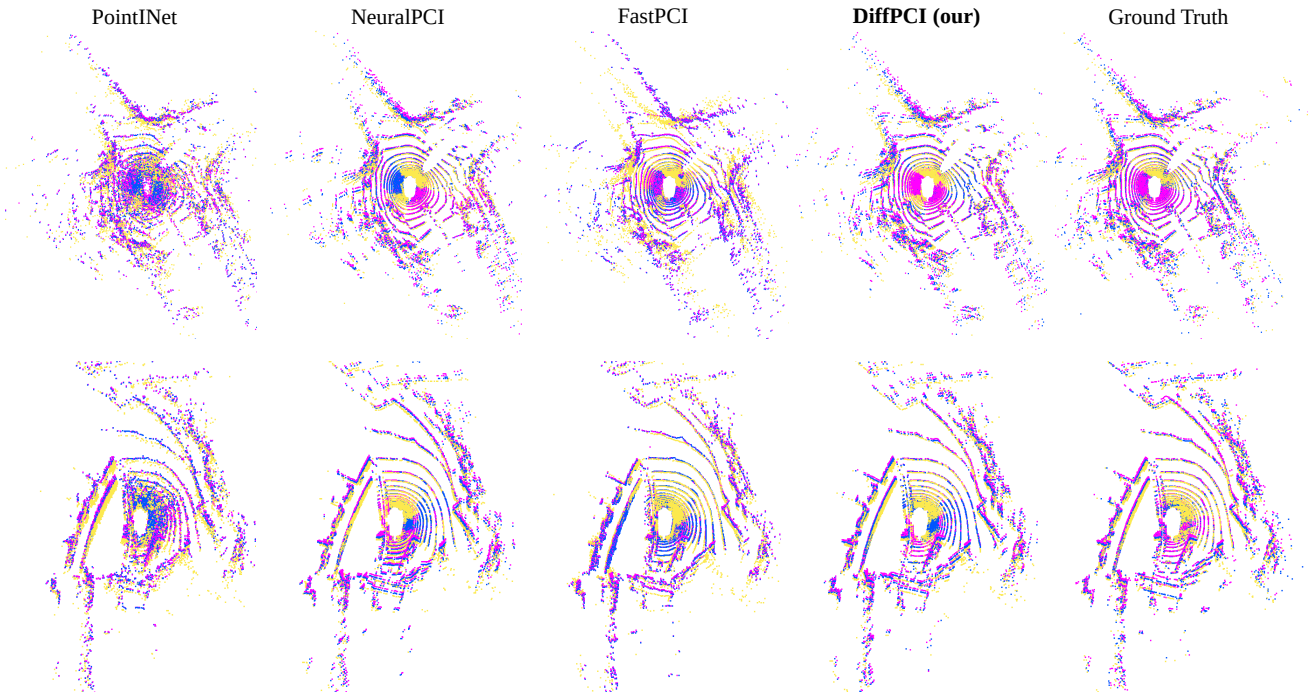


Figure S4. **More qualitative comparison on Argoverse 2 sensor dataset.**

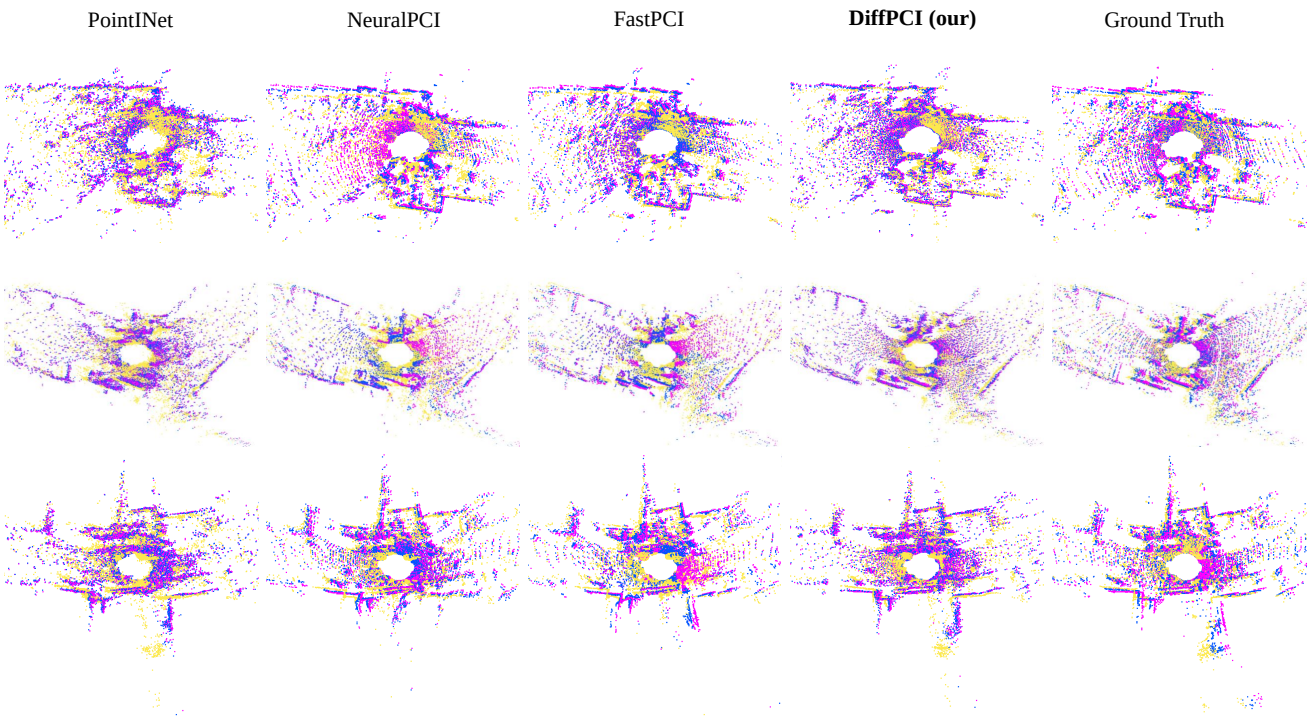


Figure S5. **More qualitative comparison on KITTI odometry dataset.**