FastPCI: Motion-Structure Guided Fast Point Cloud Frame Interpolation

Tianyu Zhang[1], Guocheng Qian[2], Jin Xie[3†], and Jian Yang[1]

[1] Nankai University, Tianjin, China
[2] Snap Research
[3] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China School of Intelligence Science and Technology, Nanjing University, Suzhou, China
tianyu.zhang@mail.nankai.edu.cn, guocheng.qian@kaust.edu.sa
{csjxie, csjyang}@njust.edu.cn

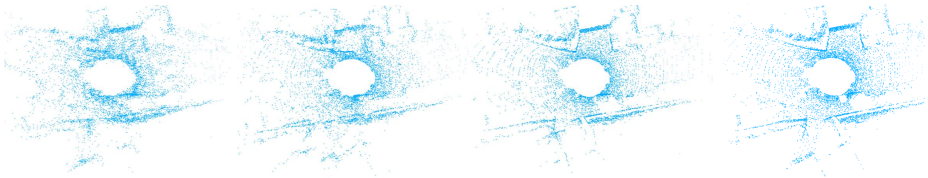


Fig. 1: Our FastPCI performs accurate point cloud frame interpolation in ~ 0.1 seconds per frame, being more accurate and more than $600\times$ and $10\times$ faster than the state-of-the-art NeuralPCI [40] and PointINet [14].

Abstract. Point cloud frame interpolation is a challenging task that involves accurate scene flow estimation across frames and maintaining the geometry structure. Prevailing techniques often rely on pre-trained motion estimators or intensive testing-time optimization, resulting in compromised interpolation accuracy or prolonged inference. This work presents FastPCI that introduces Pyramid Convolution-Transformer architecture for point cloud frame interpolation. Our hybrid Convolution-Transformer improves the local and long-range feature learning, while the pyramid network offers multilevel features and reduces the computation. In addition, FastPCI proposes a unique Dual-Direction Motion-Structure block for more accurate scene flow estimation. Our design is motivated by two facts: (1) accurate scene flow preserves 3D structure, and (2) point cloud at the previous timestep should be reconstructable using reverse motion from future timestep. Extensive experiments show that FastPCI significantly outperforms the state-of-the-art PointINet and NeuralPCI with notable gains (*e.g.* 26.6% and 18.3% reduction in Chamfer Distance in KITTI), while being more than $10\times$ and $600\times$ faster, respectively. Code is available at https://github.com/genuszty/FastPCI.

Keywords: Point Cloud Frame Interpolation · Point Cloud Processing · Motion Estimation

† Corresponding author.

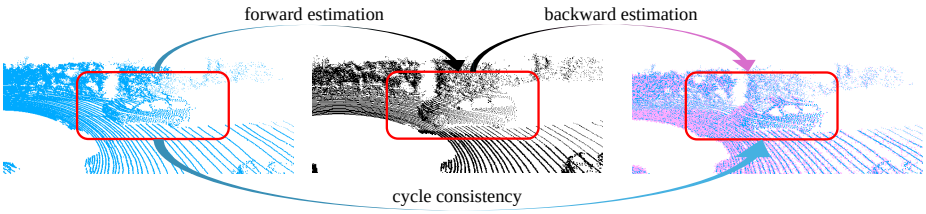


Fig. 2: (left) : frame $t-1$; (middle) : frame t ; (right) : the overlapping original frame $t-1$ and the frame $t-1$ estimated from frame t. FastPCI produces structure-aware motion and performs dual-direction motion estimation motivated by two facts: (1) **structure consistency**: an accurate motion preserves structure of objects, *e.g.* the car in red box; (2) **cycle consistency**: point cloud at frame t (right) is predicted from frame $t-1$ (left) by the estimated motion, meanwhile frame $t-1$ can be reconstructed from frame t (middle) by applying the reverse estimated motion (right).

1 Introduction

LiDAR is widely used in autonomous driving, virtual/augmented reality, and robotics [4, 35, 38]. However, due to hardware limitations, LiDAR usually captures low frame-rate scans that can lead to temporal discontinuities. Point Cloud Frame Interpolation (PCI) addresses this challenge by synthesizing intermediate frames, which has the wide application in enhancing the temporal resolution of the data and improving the efficacy of applications reliant on continuous environmental representation, such as real-time tracking and navigation [16, 35].

Early work [12, 13] relied on stereoscopic images to generate pseudo-LiDAR point cloud interpolation. Recent works [11, 14, 32, 36, 40] adopt neural networks to predict interpolated point clouds end-to-end in a data-driven manner. IDEA-Net [36] refines the linear interpolation results by learning a one-to-one alignment matrix and modeling a trajectory compensation module. However, the one-to-one correspondence assumption in IDEA-Net limits its application on large outdoor point cloud datasets. PointINet [14] predicted the interpolated point cloud through learning fusion and refinement networks with a fixed pretrained motion estimator. However, it relies merely on pre-trained motion estimation and cares less about structure consistency, leading to noisy prediction as illustrated in Figure 1. Most recently, NeuralPCI [40] adopted a dynamic neural radiance field [17, 21] and achieved state-of-the-art accuracy in PCI. Unfortunately, NeuralPCI requires testing-time per-scene optimization, which is rather expensive.

In contrast to pioneering PCI methods, our work is based on two facts, as illustrated in Figure 2: (1) *structure consistency*: an accurate motion preserves the structure of objects; (2) *cycle consistency*: point cloud at frame t is predicted by the estimated motion $M_{t-1,t}$ from the frame $t-1$, and a reversed motion $M_{t,t-1}$ should reconstruct the point cloud in the frame $t-1$. Respecting these two consistencies, we propose a Dual-Direction Motion-Structure Transformer block. Given the point features from frame 0 and frame 1, we concatenate them in forward and backward order and use them for a bidirectional input in the block to encour-

age the model be aware of the motion direction. We perform a cross-attention between the bidirectional input to estimate the motion and structure jointly. We hierarchically stack this Motion-Structure block, and propose our Pyramid Motion-Structure Estimation Network. The pyramid structure provides multi-level features and reduces the computational burden. The usage of Transformer enhances the global reasoning ability due to its attention mechanism. We finally propose FastPCI consisting of a Pyramid Motion-Structure estimation network, a motion compensation network, and an interpolation refinement network for point cloud frame interpolation. Through extensive experiments in three large-scale automated driving datasets, we demonstrate significant improvements over the state-of-the-art in both accuracy and speed, as shown in Figure 1.

Contributions of our work are listed as follows:

- We present Dual-Direction Motion-Structure Transformer block. This block estimates motion in a structure-aware manner through the mixed information from forward and backward point features.
- We propose FastPCI that adopts a Pyramid Convolution-Transformer architecture tailored for rapid and precise point cloud frame interpolation.
- We propose to optimize FastPCI with reconstruction loss and additional pyramid loss and dual-direction losses. Comprehensive evaluations across diverse autonomous driving datasets demonstrate FastPCI's superiority over the state-of-the-art.

2 Related Work

Point Cloud Frame Interpolation. Existing point cloud interpolation methods try to find point-to-point correspondences between two point cloud reference frames. They can be divided into two categories. One is to explicitly predict continuous interframe motion by estimating scene flow. PointINet [14] warps two input frames with bi-directional flows to the intermediate frame, then adaptively samples the two warped results and fuses them with attentive weights. This method is implemented based on pre-trained scene flow, focusing more on local motion. The effectiveness of frame interpolation of PointINet highly depends on the accuracy of the scene flow. The other is an implicit field estimation technique that requires testing-time optimization. NeuralPCI [40] creates a spatial-temporal neural field to fit the given frames, and is able to output interpolated frames due to its continuous modeling. The fatal disadvantage of NeuralPCI is the prolonged inference time. In our work, we still train a feedforward point cloud frame interpolation network that runs efficiently in inference. We present a hybrid convolution-transformer network with a pyramid architecture. Our model incorporates a structure-aware motion estimation and a dual-direction cross-attention as inspired by the structure consistency and cycle consistency. We also provide explicit losses to enhance pyramid feature learning through hierarchical supervision and to enforce cycle consistency.

Video Frame Interpolation The goal of the video frame interpolation (VFI) task is that the predicted frame at any time should not only satisfy the conti-

nuity of the dynamic objects but also have the same smooth appearance as the input frame. Therefore, the existing VFI methods can be regarded as the process of capturing the motion between successive frames, or explicitly or implicitly mixing the appearance consistent with the input frame to synthesize the intermediate frame. Many VFI methods perform motion estimation and appearance extraction in a hybrid manner [1, 6, 9, 15], warp the input frame with the help of optical flow, and then mix the appearance information to enhance the detail and generate an intermediate frame. Due to the need to handle both motion and appearance, this approach requires well-designed and high-capacity extraction modules. There are also many ways to design separate modules to extract motion and appearance information [7, 18, 19, 28], and the addition of additional modules often creates a high computational overhead. To deal with complex motions and reduce the complexity of the model, EMA-VFI [37] proposes an interframe attention module, which is integrated into the hybrid architecture of CNN and transformer, and reuses attention mapping to extract appearance features and motion features, saving the computational overhead of a single model. These works inspire us to design a point cloud interpolation network that extracts both motion and structure-consistent information end-to-end. However, extending these approaches to unordered and unstructured point clouds remains a challenge.

General Point Cloud Processing With the popularity of three-dimensional scanning devices such as lidar, laser, or RGB-D scanners, the capture of point clouds has become easier, and point cloud processing technology has gained great attention. Earlier work transformed point clouds into regular grids or 3D voxels, which are input into a typical convolutional architecture. However, this data representation transformation loses the point cloud structure and introduces quantization artifacts. The pioneering point-based work is PointNet [22], which builds a framework capable of directly processing point cloud data using operators with permutation invariance, such as point-by-point MLP and pooling layers, to aggregate features across sets. PointNet++ [23] compensates for the shortcomings of pointnet and achieves global and local detail feature extraction. PointConv [31], DGCNN [30] and Point transformer [39] are pioneering frameworks based on point convolution, graph convolution, and transformer, respectively. Followups [10, 24–26, 29] propose different designs to improve the efficacy or efficiency of point cloud learning. Our work is inspired by these 3D point cloud processing [33]. We introduce a hybrid Convolution-Transformer framework with a pyramid structure into the point cloud frame interpolation field, along with novel filed-specific designs to push the boundary for state-of-the-art point cloud frame interpolation.

3 Method

In this section, we present FastPCI, a Pyramid Convolution-Transformer architecture for fast point cloud frame interpolation. FastPCI adopts Motion-Structure Transformer along with a Dual-Direction Cross-Attention to estimate

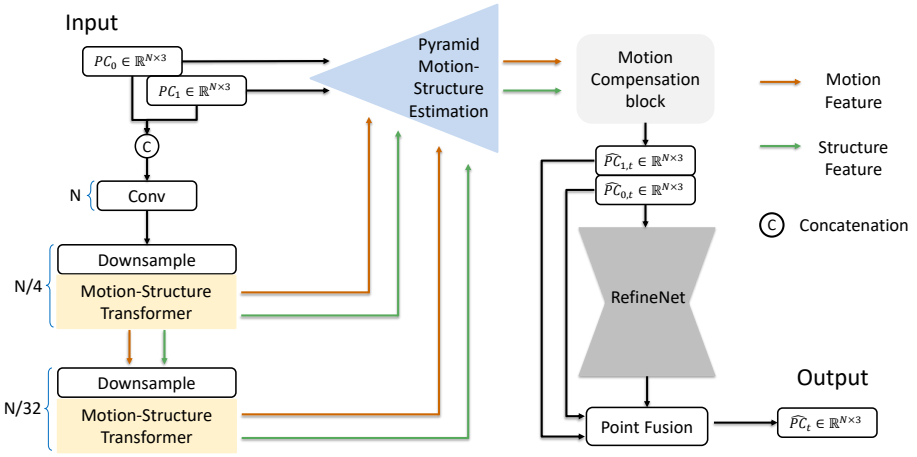


Fig. 3: Overview of FastPCI pipeline. Given two input frames $PC_0 \in \mathbb{R}^{N\times 3}$ and $PC_1 \in \mathbb{R}^{N\times 3}$, FastPCI estimates both motion and structure using a Pyramid Convolution-Transformer network. The estimated motion is used to warp the input frame to produce interpolated frames. RefineNet further refines the interpolated frames, and outputs the final frames from the fused forward and backward estimates.

structure-aware motion using bidirectional features. We also propose architecture-related loss functions to further improve performance and will elaborate next.

3.1 Overall Pipeline

Our overall pipeline is illustrated in Figure 3. Given two consecutive point clouds with low temporal resolution $PC_0 \in \mathbb{R}^{L\times 3}$ and $PC_1 \in \mathbb{R}^{L\times 3}$ in the time step $t = 0$ and $t = 1$, the goal of FastPCI is to generate the point cloud frame $PC_t \in \mathbb{R}^{L\times 3}$ in arbitrary time step $t \in (0, 1)$. Here L denotes the number of points in the current point cloud. FastPCI first estimates the motion feature $M \in \mathbb{R}^{L\times C}$ and structure feature $S \in \mathbb{R}^{L\times C}$ from the proposed Pyramid Convolution-Transformer Estimation network. The motion feature M is the high-dimensional representation of the scene flow SF that represents the displacement between the corresponding points of consecutive input frames. Throughout the paper, M and $M_{0,1}$ represent forward motion from time 0 to time 1, and $M_{1,0}$ represent reverse motion from time 1 to 0. Then, we refine our motion estimation through a compensation block taking M and S as input and output the scene flow SF. A dual-direction estimation of interpolated point clouds $\widehat{PC}_{0,t}$ and $\widehat{PC}_{1,t}$ can be obtained by adopting forward and reverse motions from time steps 0 and 1, respectively. Here, $\widehat{PC}_{0,t}$ is the forward estimate of $\widehat{PC}_t$ from PC_0 and the backward estimate of $\widehat{PC}_t$ from PC_1. Then, a RefineNet is optimized to update the forward estimate $\widehat{PC}_{0,t}$. Finally, we follow the same fusion strategy as PointINet [14] to get the final output from the refined forward estimate and

the backward estimate. The entire system is trained end-to-end with ground truth PC_t.

We highlight two unique designs in our pipeline which we elaborate on next: (1) Pyramid Motion-Structure Network for structure-aware motion estimation. Keeping the structure of the front and back frames consistent is crucial for point cloud frame interpolation. This motivates our design of structure-aware motion estimation, which differs from the state-of-the-art considering only motion. Our network also employs Pyramid Convolution-Transformer for fast processing and multilevel feature learning. Our novel use of Transformer for point cloud frame interpolation also adds a global reasoning ability to the network and overcomes the limitation of local processing of point convolutions. (2) Motion-Structure Transformer Block that introduces dual-direction cross-attention for improved motion estimation by considering both forward and backward point features.

3.2 Pyramid Motion-Structure Estimation

Our Pyramid Motion-Structure Estimation Network takes two frames at time 0 and 1 as input and estimates the forward motion features and structure features from frame 0 to frame t. The output of this network is fed into a Motion Compensation to get the scene flow, *i.e.* motion in coordinate space.

Pyramid Motion-Structure Estimation consists of three pyramid stages. The first stage is built using a three-layer MLP (a mini-PointNet [22]). The second and third stages are built using a downsampling layer to reduce the initial spatial resolution N by $4\times$ and $32\times$ respectively. They are all followed by a cost layer and an SF Predictor (details in **Appendix**). Multilevel motion and structure features estimated by our Motion-Structure Transformer are fed into the SF Predictor of the second and third stages. At each stage, motion and structure features are captured from the point features F_0^l, F_1^l by the Motion-Structure Transformer block. Here, F and l denote the point features and the l^{th} stage.

3.3 Motion-Structure Transformer

As illustrated in Figure 4, our Motion-Structure Transformer estimates motion features and structure features from point features. In detail, given features $F_0^l \in \mathbb{R}^{L\times C}$ and $F_1^l \in \mathbb{R}^{L\times C}$ of the timestep 0 and 1 at stage l, we normalize the features, and stack them in forward and backward order, which gives a dual-direction features $F_{0,1}^l \in \mathbb{R}^{L\times C} = [F_0^l; F_1^l]$ and $F_{1,0}^l \in \mathbb{R}^{L\times C} = [F_1^l; F_0^l]$, where $L, C, ;$ denote the number of points, channel size, and concatenation operation in the batch dimension, respectively. A subsequent Dual-Direction Cross Attention module is used to output forward motion and structure features by using the forward feature as query and the backward feature as key and value. Mathematically, query, key, and value in the attention is given by:

$$q_{0,1} = F_{0,1}^l W_q, k_{1,0} = F_{1,0}^l W_k, v_{1,0} = F_{1,0}^l W_v, \tag{1}$$

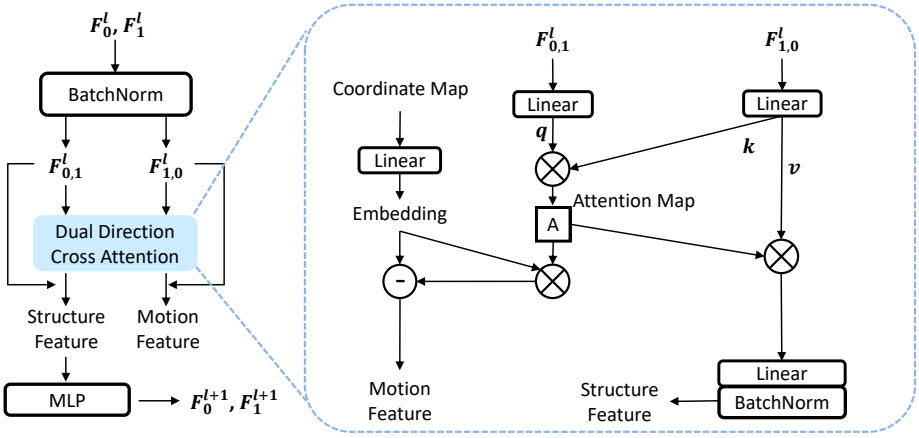


Fig. 4: The illustration of Motion-Structure Transformer. Where $\otimes$ and $\ominus$ denote matrix multiplication and element-wise subtraction, respectively. We highlight our Motion-Structure Transformer takes a bidirectional point features input, and perform a Dual-Direction Cross-Attention across forward and backward features. The structure and motion features are closely related to each other to learn a structure-aware motion.

where W_q, W_k, $W_v \in \mathbb{R}^{C\times\hat{C}}$ are linear projection weights. Then we calculate the dual-direction cross-attention map and the structure features as follows:

$$A = SoftMax(q_{0,1}k_{1,0}^{\mathrm{T}}/\sqrt{C}), \tag{2}$$

$$S = W_S(Av_{1,0}), \tag{3}$$

where W_S represents linear projection for the structure features. The structure features are passed to a two-layer MLP to update points features F_0^{l+1}, F_1^{l+1} for the next stage.

For motion estimation, we create a coordinate map $B \in \mathbb{R}^{L\times 3}$, where the value at each location indicates the unit position that expands uniformly along each direction and covers the entire point cloud frame. A higher dimensional embedding of B_1 is obtained in each cross-attention block by applying a single linear layer on B. The coordinate embeddings are multiplied by attention map A to estimate the corresponding position of the point after the motion. Finally, the motion vector $M_{0,1}$ is estimated by applying another linear projection on the difference between the original B_1 and the wrapped B_1 by attention map:

$$M_{0,1} = W_M(AB_1 - B_1), \tag{4}$$

where W_M represents linear projection for the motion features. It is worth noting that under the assumption of local linear motion, the motion features and structure features obtained by the Motion-Structure Transformer module can be multiplied by t to synthesize intermediate frames at any t moment, where $t \in [0, 1]$. We also highlight that the attention A comes from the point features

F^l, which are learned from the structure features. Therefore, the motion and structure features in our block are closely related. The motion features estimated from the Motion-Structure Transformer are structure-aware.

3.4 Motion Compensation Block

Inspired by video frame interpolation [15], we propose Motion Compensation Block to refine motion. In contrast to [15] and different from point cloud frame interpolation methods, we take the concatenation of motion and structure features as input. Our pivotal consideration of both motion and structure enhances the capture of motion relationships and the preservation of structural integrity between frames. Specifically, first, a three-layer MLP and sigmoid are leveraged to enhance nonlinear structure features, while input motion features are fed into one-dimensional convolution and LeakyReLU [34] for enhancement. Then, we concatenate the enhanced structure and motion features to a motion offset head built by two layers of convolution to obtain a motion correction term. The motion correction term is added back to the input motion to get the refined motion features. From the motion features, we use a scene flow prediction layer to map the motion features to $SF^l \in \mathbb{R}^{L\times 3}$.

Due to our hierarchical architecture, we can get a multilevel scene flow. A warping operation can be applied to obtain an interpolation frame at each timestep $t \in [0, t]$ for each stage by:

$$\widehat{PC}^l_{0,t} = \widehat{PC}^l_0 + SF^l * t, \quad \widehat{PC}^l_{1,t} = \widehat{PC}^l_1 + SF^l * (1-t). \tag{5}$$

Specifically, the final clean scene flow is used to predict the final output. The scene flows in each other level are used for supervised pyramid prediction.

3.5 RefineNet

We use an additional RefineNet to refine the forward estimate $\widehat{PC}_{0,t}$ since it is more sensitive to the final results. We simply leverage a three-stage U-Net [27] to encode and decode the input points. After each upsampling and downsampling layer, we use a Point Transformer block [39] to enhance features. The output of the UNet is concatenated with our backward estimate to yield a fused point cloud. We then closely follow PointINet [14] to get the final interpolated point cloud from the fused estimates. Refer to their paper for the details on estimating the final output from the fused estimation.

3.6 Objective Function

Following previous work [14,40], the entire system can be end-to-end trained by Chamfer Distance (CD) between the predicted point cloud frame $\widehat{PC}_t \in \mathbb{R}^{N\times 3}$ at time t and the Ground Truth $PC_t \in \mathbb{R}^{N\times 3}$, as follows:

$$L_{intp} = \frac{1}{N}\sum_{\hat{x}^i \in \widehat{PC}_t} \min_{x^j \in PC_t} \left\|\hat{x}^i - x^j\right\|_2 + \frac{1}{N}\sum_{x^j \in PC_t} \min_{\hat{x}^i \in \widehat{PC}_t} \left\|\hat{x}^i - x^j\right\|_2, \tag{6}$$

In addition to this loss, we find that the following two regularizations can significantly improve the performance. First is a **half-way cycle consistency** that forces both of the forward estimate $\widehat{PC}_{0,t}$ and the backward estimate $\widehat{PC}_{1,t}$ to be close to the Ground Truth. Note we do not directly enforce a cycle loss, *i.e.* reconstructing frame 0 from frame 1 using reverse motion, since it might encourage the network to learn an identity mapping. Mathematically, the half-way cycle loss encourages cycle consistency as well. This half-way cycle loss is given by the sum of two CD losses:

$$L_{CD_1} = CD(\widehat{PC}_{0,t}, PC_t), \tag{7}$$

$$L_{CD_2} = CD(\widehat{PC}_{1,t}, PC_t), \tag{8}$$

The second regularization is the **pyramid reconstruction loss**. Thanks to the hierarchical architecture, FastPCI offers predicted frames of different sizes. We thus propose the following multiscale loss L_{ms} following the pyramid structure as shown next:

$$L_{ms} = \sum \alpha_l (CD(\widehat{PC}_t^l, PC_t^l)), \tag{9}$$

where PC_t^l is obtained through furthest point sampling from the Ground Truth. α_l denotes the loss weight for pyramid level l. Throughout the experiments, we use $\alpha_0 = 0.05$, $\alpha_1 = 0.1$, $\alpha_2 = 0.2$. We empirically find that as long as α is set to $(0.025 - 0.25)$, the final performance is similar.

Overall, FastPCI is optimized by the sum of all losses and regularization:

$$L = L_{intp} + L_{CD_1} + L_{CD_2} + L_{ms}. \tag{10}$$

4 Experiments

4.1 Experimental Setup

Datasets. The proposed FastPCI is assessed on three large outdoor LiDAR datasets, namely KITTI odometry [5], Argoverse 2 sensor [3], and Nuscenes [2]. The KITTI odometry and Argoverse 2 sensor data are point cloud frames collected at a frame rate of 10Hz, whereas the Nuscenes data are collected at 20Hz. To align datasets, the Nuscenes data are first downsampled to 10Hz. We follow NeuralPCI [40] to split the training and testing sets. In detail, KITTI odometry dataset contains 11 annotated LiDAR point cloud sequences in total, where the first 7 sequences are used as training and the left is used as testing. Argoverse 2 sensor dataset is composed of 1,000 scenarios with 150 LiDAR sweeps per scenario on average, while Nuscenes dataset also consists of 1,000 driving scenes with about 400 LiDAR frames for each scene. For these two datasets, we use the first 700 scenes to train, and 850-1000 scenes to test. For all datasets, the input spatial resolution is 8192 points, and the temporal resolution is 2Hz. For evaluation, we estimate 3 intermediate frames per two frames.

Table 1: Quantitative comparison with the state-of-the-art on KITTI odometry, Argoverse 2 sensor, and Nuscenes. *Frame-1*, *Frame-2* and *Frame-3* refer to the three uniformly intermediate frames to be interpolated between the two input frames. Average denotes the average result on these three frames. **Bold** represents the best performance across different methods.

Dataset	Methods	Frame-1		Frame-2		Frame-3		Average	
		CD	EMD	CD	EMD	CD	EMD	CD↓	EMD↓
KITTI odometry	PointPWC-Net [32]	0.64	71.14	0.80	91.91	0.91	60.35	0.78	74.46
	NSFP [11]	0.58	70.53	0.68	84.76	1.95	115.42	1.07	90.24
	PointINet [14]	0.72	55.25	0.82	77.87	0.83	73.74	0.79	69.19
	NeuralPCI [40]	0.64	52.61	0.85	62.73	0.64	52.15	0.71	55.83
	FastPCI(our)	**0.54**	**51.23**	**0.61**	**59.84**	**0.58**	**50.27**	**0.58**	**53.78**
Argoverse 2 sensor	PointPWC-Net [32]	0.90	56.44	1.07	79.86	1.26	65.32	1.07	67.20
	NSFP [11]	0.72	62.30	0.85	73.89	2.14	98.99	1.24	78.40
	PointINet [14]	0.83	57.89	1.25	67.73	1.06	62.97	1.05	62.86
	NeuralPCI [40]	0.68	55.03	0.88	65.93	**0.69**	55.30	0.75	58.75
	FastPCI(our)	**0.68**	**54.99**	**0.77**	**64.29**	0.74	**54.59**	**0.73**	**57.95**
Nuscenes	PointPWC-Net [32]	1.52	172.31	1.39	224.64	2.16	181.39	1.69	192.78
	NSFP [11]	1.10	173.03	1.33	212.32	4.37	319.56	2.27	234.97
	PointINet [14]	1.48	183.03	1.67	202.10	1.50	186.51	1.55	190.54
	NeuralPCI [40]	**1.00**	163.10	1.37	**205.24**	1.06	168.98	1.15	179.11
	FastPCI(our)	1.02	**162.78**	**1.28**	205.75	**1.03**	**152.39**	**1.11**	**173.64**

Implementation Details. We use PyTorch [20] to implement FastPCI. We train FastPCI in all three training sets using the same parameters on a NVIDIA GeForce RTX 3090 GPU. FastPCI is optimized by Adam [8] with a batch size of 4, a weight decay of 10^{-4}, and an initial learning rate of 10^{-3} that is reduced by half every 80 epochs. We train KITTI odometry, Argoverse 2 sensor, and Nuscenes datasets by 100, 200, and 200 epochs, respectively.

Evaluation Metrics. We adopt CD and Earth Mover's Distance (EMD) as quantitative evaluation metrics following [14, 40]. EMD measures the minimum average distance required to move data from a point cloud to another point cloud. Given two point clouds $\widehat{PC}_t \in \mathbb{R}^{N\times 3}$ and $PC_t \in \mathbb{R}^{N\times 3}$, EMD can be calculated as follows, where $\phi : \widehat{PC}_t \to PC_t$ is the bijection set:

$$L_{EMD} = \min_{\phi:\widehat{PC}_t \to PC_t} \frac{1}{N} \sum_{\hat{x}\in\widehat{PC}_t} \|\hat{x} - \phi(\hat{x})\|_2 \tag{11}$$

4.2 Evaluation of Point Cloud Interpolation

To demonstrate the performance of FastPCI, we compare our method to previous SOTA methods, namely PointINet [14] and NeuralPCI [40], as well as the pioneering work [11, 32]. Due to the different preprocessing and data split across methods, it is not fair to compare with original results directly. Therefore, in our work, we use the same data splitting and preprocessing as in the most recent work and the state-of-the-art NeuralPCI. We report the original results of

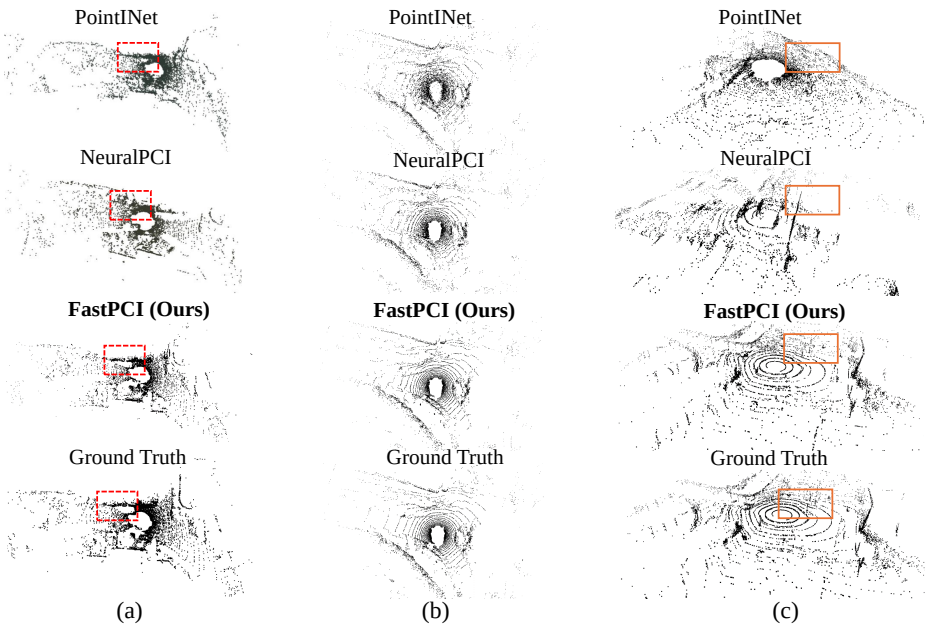


Fig. 5: Qualitative comparisons with the state-of-the-art on KITTI odometry, Argoverse 2 sensor, and Nuscenes dataset. Columns (a)-(c) represent the results on three datasets, respectively. Each row represents a different method. Our FastPCI (3^{rd} row) yields the best qualitative results compared to the state-of-the-art PointINet (1^{st} row) and NeuralPCI (2^{nd} row).

NeuralPCI and retrain other methods using the official implementations and the NeuralPCI's setups in all datasets.

Results on KITTI Odometry Dataset. rows 3-7 of Table 1 show the results on the KITTI odometry dataset, where FastPCI achieves the best results on all frames and in terms of all metrics. Specially, our approach reduces the EMD error by 2.89 and the CD error by 0.21 in the frame-2 compared to the state-of-the-art NeuralPCI [40], ultimately reducing 2.05 and 0.13 in the overall EMD and CD metrics results, respectively. Figure 5 (a) shows the visual comparisons among different methods on the KITTI odometry dataset. Highlighted in the red closeup, our FastPCI yields the best visual quality, producing the sharpest edge for vehicles with much less noise.

Results on Argoverse 2 Sensor Dataset. The quantitative comparison results are shown in rows 8-12 of Table 1, where FastPCI again achieves the best performance on nearly all frames across metrics. Figure 5 (b) shows the qualitative experimental results of interpolated frames on the Argoverse 2 sensor dataset. Overall, the results of our method are closer to ground truth. Refer to the road and the boundary for differences.

Table 2: Average inference time Comparison. Time is measured per frame with 8192 points on a NVIDIA RTX 3090 GPU.

Methods	**PointINet [14]**	**NeuralPCI [40]**	**FastPCI(our)**
Runtime	1458ms	60822ms	**109ms**

Results on Nuscences Dataset. Rows 13-17 in Table 1 display the quantitative results for Nuscenes. FastPCI has achieved the top performance on the overall index. Our method is slightly worse than the NeuralPCI [40] method on the CD metric in frame-1 and the EMD metric in frame-2, but exceeds the NeuralPCI method by 5.47 and 0.04 on the overall EMD and CD metric. In the qualitative experiment, we present the frame interpolation visualization results of two advanced PCI methods and the proposed FastPCI on the Nuscenes dataset in Figure 5 (c). From the contents of the yellow box, it is not difficult to find that compared with other methods, our method can insert clearer human outline lines.

Runtime Comparison. Fast point cloud frame interpolation can facilitate downstream applications in autonomous driving scenarios (such as target detection, target tracking, etc.). Therefore, we give the average inference time of a point cloud frame interpolated at any time in Table 2. Compared with the current state-of-the-art point cloud frame interpolation methods, FastPCI shows the fastest inference times, being over 10 times faster than PointINet and 600 times faster than NeuralPCI, due to its Pyramid Convolutional-Transformer architecture.

4.3 Ablation Study

We contribute in both architecture design and loss functions as follows: (1) Structure-aware motion Estimation; (2) Dual-Direction Estimation; (3) Hybrid Convolution-Transformer architecture; (4) Motion Compensation block; (5) RefineNet; (6) Cycle consistency loss; (7) Multiscale loss. We ablation each contribution through the following ablation study.

4.4 Ablate Architecture Designs

Structure-Aware Motion Estimation aims to keep the structure of the front and back frames consistent, which is crucial for the quality of the frame interpolation. To verify the effectiveness of our compound motion-structure learning, we remove all the structural branches and only keep the motion feature extraction in our network. As shown in Table 3, without a structure-aware design, a significant increase in the distances of CD and EMD can be observed. This experiment shows the importance of the Motion-Structure joint learning.

Dual-Direction Estimation is designed to encourage the interaction of information between forward and backward frames. To show its effectiveness, we

Table 3: Ablate architecture designs on all three datasets. We show that our structure-aware motion estimation, the hybrid Convolution-Transformer architecture, the Dual-Direction Cross-Attention, the Motion Compensation block, and the RefineNet lead to improved performance in all datasets.

Dataset	Component	Metrics CD↓	EMD↓
KITTI odometry	w/o structure branch	0.79	67.60
	Transformer → convolution	0.61	56.38
	w/o Dual Direction	0.60	56.34
	w/o Motion Compensation	0.60	55.29
	w/o RefineNet	0.60	54.53
	Baseline	**0.58**	**53.78**
Argoverse 2 sensor	w/o structure branch	0.85	62.23
	Transformer → convolution	0.77	58.16
	w/o Dual Direction	0.75	58.31
	w/o Motion Compensation	0.75	58.14
	w/o RefineNet	0.74	58.17
	Baseline	**0.73**	**57.95**
Nuscenes	w/o structure branch	1.64	193.95
	Transformer → convolution	1.43	189.95
	w/o Dual Direction	1.23	181.58
	w/o Motion Compensation	1.20	179.35
	w/o RefineNet	1.12	175.66
	Baseline	**1.11**	**173.64**

Table 4: Ablate loss functions. The proposed dual-direction reconstruction losses L_{CD_1}, L_{CD_2}, and the pyramid loss L_{ms} improve the performance on all benchmarks. The dual-direction reconstruction losses are more crucial.

Dataset	Loss Item L_{CD_1}	L_{CD_2}	L_{ms}	Metrics CD↓	EMD↓
KITTI odometry	√	√		0.73	60.99
	√		√	0.59	55.54
		√	√	0.74	58.47
				0.78	67.59
	√	√	√	**0.58**	**53.78**
Argoverse 2 sensor	√	√		0.80	59.35
	√		√	0.75	59.92
		√	√	0.83	60.57
				0.92	61.90
	√	√	√	**0.73**	**57.95**
Nuscenes	√	√		1.37	184.28
	√		√	1.23	179.80
		√	√	1.56	185.15
				1.57	189.63
	√	√	√	**1.11**	**173.64**

replace all reverse input features $F^l_{1,0}$ in Figure 4 with forward features $F^l_{0,1}$. In other words, we use standard self-attention on forward features to substitute the original Dual-Direction Cross Attention. Shown in Table 3, without Dual Direction leads to a clearly worse performance, despite the existing cycle loss.

Hybrid Convolution-Transformer utilizes convolution to extract local features and Transformer for global reasoning ability. We also ablate replacing the Transformer with regular convolutions. Due to the lack of Transformer, we do not perform cross attention anymore, and only use pure convolutions (two-layer mini-PointNet) to learn both motion and structure. As shown in Table 3 w/o Transformer, significantly worse results are reported. These experiments point out the importance of Transformer in our design.

Motion Compensation block refines the estimated motion from motion and structure features. Here, we remove this block in Table 3, and see a slightly worse performance. This module marginally improves the performance, which is not the key component in FastPCI.

RefineNet is used to refine the interpolated point cloud. Similar to the Motion Compensation block, we remove the RefineNet and use the wrapped forward and backward estimates to fuse and get final results directly. Table 3 w/o RefineNet

shows a marginal performance drop, which indicates that the use of RefineNet can enhance results but the crucial part is Motion Compensation.

4.5 Ablate Loss Functions

Half-way Cycle Consistency aims to ensure the accuracy of forward and backward estimated motion. From Table 4, one can observe a significant drop when removing L_{CD_1}, showing the importance of supervising the coarse estimation of frame interpolation. The addition of backward warp loss L_{CD_2} further improves cycle consistency, and thus improves the performance.
Multiscale loss provides hierarchical supervision for the interpolated frames estimated from the pyramid features. We empirically find that the multiscale loss contributes nontrivially to the performance of FastPCI. Along with the pyramid architecture, the hierarchical design in the network and the loss function are the most important designs in FastPCI and significantly boost performance. See **Appendix** for the ablation study in the pyramid architecture for details.

5 Limitation

While FastPCI has made great strides in point cloud frame interpolation, there are still some limitations. The effectiveness of this method under different environmental conditions and its robustness to outliers and occlusion need to be further studied. Furthermore, the adaptability of FastPCI to real-world and real-time deployment remains to be tested and explored.

6 Conclusion

This paper introduces FastPCI, a pioneering Pyramid Convolution-Transformer hybrid architecture designed for rapid and accurate point cloud frame interpolation. The introduction of Transformer and the hierarchical architecture bring long-range information extraction capability to our system, while maintaining efficiency. We also observe the importance of structure consistency and cycle consistency to the task of point cloud frame interpolation. To respect both of them, our FastPCI proposes a unique Dual-Direction Motion-Structure estimation, to mix information between forward and backward estimates, as well as between motion and structure features. We also present two losses closely related to our architecture design. The first is the half-way cycle consistency loss that encourages the network to learn a cycle-consistent motion estimation. The second is a multilevel reconstruction loss that allows middle-stage supervision using lower-resolution ground truth, thanks to our hierarchical architecture design. Overall, FastPCI establishes a new benchmark in point cloud frame interpolation, surpassing the state-of-the-art methods PointINet and NeuralPCI with notable margins while being much more efficient.

Acknowledgements

The authors would like to thank the valuable feedback from the anonymous reviewers. This work is supported in part by the NSFC under Grant Nos. 62276144.

References

1. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: CVPR (2019) 4
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. p. 11621–11631 (2020) 9
3. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: CVPR. p. 8748–8757 (2019) 9
4. Garrido, D., Rodrigues, R., Augusto Sousa, A., Jacob, J., Castro Silva, D.: Point cloud interaction and manipulation in virtual reality. In: 2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR). p. 15–20 (2021) 2
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR. p. 3354–3361 (2012) 9
6. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Real-time intermediate flow estimation for video frame interpolation. In: ECCV (2022) 4
7. Kalluri, T., Pathak, D., Chandraker, M., Tran, D.: Flavr: Flow-agnostic video representations for fast frame interpolation. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). p. 2070–2081 (2021) 4
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
9. Kong, L., Jiang, B., Luo, D., Chu, W., Huang, X., Tai, Y., Wang, C., Yang, J.: Ifrnet: Intermediate feature refine network for efficient frame interpolation. In: CVPR (2022) 4
10. Li, G., Müller, M., Qian, G., Delgadillo, I.C., Abualshour, A., Thabet, A.K., Ghanem, B.: Deepgcns: Making gcns go as deep as cnns. IEEE TPAMI **PP** (2021) 4
11. Li, X., Kaesemodel Pontes, J., Lucey, S.: Neural scene flow prior. NeurIPS **34** (2021) 2, 10
12. Liu, H., Liao, K., Lin, C., Zhao, Y., Guo, Y.: Pseudo-lidar point cloud interpolation based on 3d motion representation and spatial supervision. IEEE Transactions on Intelligent Transportation Systems **23**(7), 6379–6389 (2021) 2
13. Liu, H., Liao, K., Lin, C., Zhao, Y., Liu, M.: Plin: A network for pseudo-lidar point cloud interpolation. Sensors **20**(6), 1573 (2020) 2
14. Lu, F., Chen, G., Qu, S., Li, Z., Liu, Y., Knoll, A.: Pointinet: Point cloud frame interpolation network. In: AAAI. p. 2251–2259 (2021) 1, 2, 3, 5, 8, 10, 12
15. Lu, L., Wu, R., Lin, H., Lu, J., Jia, J.: Video frame interpolation with transformer. In: CVPR (2022) 4, 8
16. Luo, W., Yang, B., Urtasun, R.: Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In: CVPR. p. 3569–3577 (2018) 2

17. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421. Springer (2020) 2
18. Niklaus, S., Liu, F.: Context-aware synthesis for video frame interpolation. In: CVPR (2018) 4
19. Park, J., Ko, K., Lee, C., Kim, C.S.: Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In: ECCV (2020) 4
20. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS **32** (2019) 10
21. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: CVPR. pp. 10318–10327 (2021) 2
22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017) 4, 6
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017) 4
24. Qian, G., Hamdi, A., Zhang, X., Ghanem, B.: Pix4point: Image pretrained standard transformers for 3d point cloud understanding. 2024 International Conference on 3D Vision (3DV) pp. 1280–1290 (2024) 4
25. Qian, G., Hammoud, H., Li, G., Thabet, A., Ghanem, B.: Assanet: An anisotropic separable set abstraction for efficient point cloud representation learning. NeurIPS **34** (2021) 4
26. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B.: Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In: NeurIPS (2022) 4
27. Ronneberger, O., Fischer, P., Brox, T.: Unet: Convolutional networks for biomedical image segmentation. In: In International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI). p. 234–241 (2015) 8
28. Sim, H., Oh, J., Kim, M.: Xvfi: extreme video frame interpolation. In: ICCV (2021) 4
29. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV (2019) 4
30. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) (2019) 4
31. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR (2019) 4
32. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In: ECCV. p. 88–107. Springer (2020) 2, 10
33. Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H.: Point transformer v2: Grouped vector attention and partition-based pooling. Advances in Neural Information Processing Systems **35**, 33330–33342 (2022) 4
34. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. CoRR (2015) 8
35. Yin, T., Zhou, X., Krähenbühl, P.: Center-based 3d object detection and tracking. In: CVPR (2021) 2
36. Zeng, Y., Qian, Y., Zhang, Q., Hou, J., Yuan, Y., He, Y.: Idea-net: Dynamic 3d point cloud interpolation via deep embedding alignment. In: CVPR (2022) 2

37. Zhang, G., Zhu, Y., Wang, H., Chen, Y., Wu, G., Wang, L.: Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In: CVPR. p. 5682–5692 (2023) 4
38. Zhang, Z., Hu, L., Deng, X., Xia, S.: Sequential 3d human pose estimation using adaptive point cloud sampling strategy. In: IJCAI. p. 1330–1337 (2021) 2
39. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: ICCV. p. 16259–16268 (2021) 4, 8
40. Zheng, Z., Wu, D., Lu, R., Lu, F., Chen, G., Jiang, C.: Neuralpci: Spatio-temporal neural field for 3d point cloud multi-frame non-linear interpolation. In: CVPR (2023) 1, 2, 3, 8, 9, 10, 11, 12