PT-FlowNet: Scene Flow Estimation on Point Clouds With Point Transformer

Jingyun Fu, Zhiyu Xiang, Chengyu Qiao, and Tingming Bai

***Abstract*—As a low-level task of 3D perception, scene flow is a fundamental representation of dynamic scenes and provides non-rigid motion descriptions for the objects in the 3D environment, which can strongly support many upper-level applications. Inspired by the revolutionary success of deep learning, many attention-based neural networks have recently been proposed to estimate scene flow from consecutive point clouds. However, extracting effective features and estimating accurate point motions for irregular and occluded point clouds remains a challenging task. In this letter, we propose PT-FlowNet, the first end-to-end scene flow estimation network embedding the point transformer (PT) into all functional stages of the task. In particular, we design novel PT-based modules for point feature extraction, iterative flow update, and flow refinement stage to encourage effective point-level feature aggregation. Experimental results on FlyingThings3D and KITTI datasets show that our PT-FlowNet achieves state-of-the-art performance. Trained on synthetic data only, our PT-FlowNet can generalize to real-world scans and outperforms the existing methods by at least 36.2% for the EPE3D metric on the KITTI dataset.**

***Index Terms*—Computer vision for automation, vision-based navigation.**

I. Introduction

SCENE flow refers to the 3D motion field of the points in the world space. Scene flow estimation provides fundamental low-level motion information for 3D dynamic scene understanding and can be used as a building block for various real-world applications, such as dynamic SLAM [1], 3D object detection [2], motion tracking [3], and robotic manipulation [4]. They employ scene flow to filter out influences from dynamic objects [1] or keep accurate tracking for them [3], fuse features among successive point clouds [2], and acquire accurate motion of the tools for manipulation [4], etc.

Manuscript received 28 September 2022; accepted 15 February 2023. Date of publication 8 March 2023; date of current version 20 March 2023. This letter was recommended for publication by Associate Editor P. V. Koerich Borges and Editor C. Cadena Lerma upon evaluation of the reviewers' comments. This work was supported in part by the The Key Research & Development Plan of Zhejiang Province under Grant 2021C01196 and in part by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grant U1709214. *(Corresponding author: Zhiyu Xiang.)*

Jingyun Fu, Chengyu Qiao, and Tingming Bai are with the College of Information Science and Eletronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: fujingyun@zju.edu.cn; 3140104437@zju.edu.cn; incredibai@zju.edu.cn).

Zhiyu Xiang is with the Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking, Zhejiang University, Hangzhou 310027, China (e-mail: xiangzy@zju.edu.cn).

Code and models can be accessed at https://github.com/FuJingyun/PT-FlowNet.

Digital Object Identifier 10.1109/LRA.2023.3254431

Many previous scene flow methods focus on employing stereo or RGB-D images as input. However, with the rapid development and the increasing application of Lidar, accurate spatial information can be efficiently collected by the sensor and direct scene flow estimation on raw point cloud has become a new research trend. With advances in deep neural networks on point clouds [5], [6], [7], [8], a series of learning-based methods [9], [10], [11], [12], [13], [14] have been proposed to generate scene flow from two consecutive point clouds. To tackle the disordered and irregular structure of the point clouds and enable deep learning on point sets, many existing point-based methods resort to K-nearest neighbor (KNN) algorithms [15], [16], [17], [18], [19], [20], [21]. However, as the position of the KNN center varies in space, invalid points from noise, background, or other objects can be included in the KNN neighborhood, which may lead to incorrect point-wise correlation and confuse the resulting point features.

Recently, attention mechanisms are showing strong potential in point cloud segmentation and classification [22], [23], [24], [25]. In particular, the attention-based methods assign adapted weights to each point in a local neighborhood based on the point-wise feature similarity. Consequently, the irrelevant points with weights close to zero can be distinguished as outliers and the more crucial points with greater weights will play more important roles in the process of feature aggregation. Some recent works have made preliminary attempts to apply the attention mechanism to scene flow estimation [26], [27], [28] and the latest work SCTN [29] first introduces transformer into its point-based model. However, these existing works limit the application of attention mechanism to some specific steps in scene flow estimation, such as point cloud abstraction or filtering candidate points for flow embedding. Besides, the transformer is only utilized for global attention on the whole point cloud in SCTN [29] and its powerful ability in aggregating information from local neighborhoods is ignored.

Based on the above observations, we believe that the potential of the transformer in scene flow estimation task has not been fully exploited and we further develop a novel full-stage PT-based network, namely PT-FlowNet. Point transformer [22] is adopted for every step of the scene flow estimation pipeline, including capturing local point relation in a multi-scale fashion for point-wise feature extraction, iterative flow generation, and handling occlusions as well as boosting the flow smoothness and consistency in the flow refinement stage. We conduct extensive experiments on both synthetic FlyingThings3D and real-world KITTI datasets and the effectiveness of each PT-based module

is also validated through ablation studies. Our key contributions are summarized as follows:

- We propose a novel PT-based network for end-to-end scene flow estimation on point clouds. To the best of our knowledge, this is the first work that embeds the point transformer onto the full stage of the scene flow estimation pipeline.
- We propose a PT-based KNN branch within the iterative update module. It can aggregate the correlated features more effectively than the common KNN equipped with max-pooling.
- The proposed PT-FlowNet achieves state-of-the-art performance on the synthetic FlyingThings3D dataset and surpasses previous approaches by remarkable margins on the real-world KITTI dataset.

II. Related Work

A. Scene Flow Estimation From Point Clouds

FlowNet3D [9] has pioneered scene flow estimation on consequent point cloud frames through the deep-learning neural network. HPLFlow-Net [10] maps point clouds into permutohedral lattices and extracts multi-scale correlations. PointPWC-Net [11] proposes novel self-supervised losses and a coarse-to-fine architecture. FLOT [12] formulates scene flow estimation as an optimal transport problem and PV-RAFT [16] leverages point-voxel correlation fields. Inspired by the successful iterative flow update technique employed in RAFT [30], FlowStep3D [15] and PV-RAFT [16] adopt the Gated Recurrent Unit (GRU) [31] to promote prediction accuracy. RMS-FlowNet [19] relies on Random-Sampling instead of the commonly used Farthest-Point-Sampling [6] for efficient large-scale point cloud processing, and RCP [20] proposes an effective two-stage recurrent network to avoid the complicated cost volume design in irregular point clouds. The recent Bi-PointFlowNet [21] proposes novel bidirectional layers for flow embedding.

Recently, the application of attention mechanism for scene flow estimation is gaining increasing attention. FESTA [26] proposes a trainable aggregate pooling to generate more stable down-sampled points than Farthest-Point-Sampling. HALFlow [27] introduces a novel attentive embedding module to focus on task-related regions and reduce information loss for patch-to-patch cost volume construction. Res3DSF [28] proposes an attention-based context-aware set convolution layer to produce local point features and a residual flow learning structure is employed to handle long-distance movement. SCTN [29] first introduces transformer for scene flow estimation, but the transformer is only built on a global scale. In this work, we propose a full-stage PT-based network for accurate scene flow estimation by deeply exploring the potential of the point transformer in aggregating information from point clouds.

B. 3D Point-Based Deep Learning

Since point-based scene flow estimation is a sub-problem of 3D point cloud processing, 3D point-based deep learning is also a very close field. PointNet [5] first puts forward innovative permutation-invariant operators to allow direct learning on point sets. PointNet++ [6] utilizes query ball grouping and further hierarchically organizes PointNet to enhance the local perception ability of the network. Some successive point-based approaches [8], [32], [33] also consume point clouds without converting them into 3D grids. Some other works [34], [35] voxelize point clouds to enable standard 3D convolution and the sparse convolution is employed afterward for more efficient learning [7], [36], [37]. Many recent works [38], [39], [40], [41] transform the point cloud into a graph and then apply graph-based operations.

Recently, the transformer from natural language processing has been extended to point cloud processing and is drawing increasing research interest. [23] presents a transformer-based framework with offset-attention and [22] develops an efficient PT layer to apply self-attention to the local region in 3D point clouds. [24] further introduces a lightweight local self-attention module to boost computational efficiency. Different from the existing models, our PT-FlowNet applies the transformer to each stage of the scene flow estimation pipeline, rather than being limited to the point cloud feature extraction process.

III. Method

Considering two consecutive frames of point clouds, i.e., $PC_1 = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ and $PC_2 = \{\mathbf{q}_j \in \mathbb{R}^3\}_{j=1}^{M}$, where $(\mathbf{p}_i, \mathbf{q}_j)$ are the 3D Cartesian coordinates for points in each set and (N, M) denote the set size, the task for the network is estimating the scene flow $SF = \{\mathbf{sf}_j \in \mathbb{R}^3\}_{i=1}^{M}$ for every point in PC_2 by finding the correlation between point clouds.

The pipeline of our proposed PT-FlowNet is shown in Fig. 1. It consists of three key modules: (1) PT-based feature extraction, (2) iterative flow update, and (3) PT-based flow refinement. In the following subsections, we first describe the common structure of the core PT layer that is embedded in each module, then introduce the design of each module one by one.

A. PT Layer

PT layer is integrated into each stage of scene flow estimation as the core functional component in our PT-FlowNet network. We follow [22] to construct the PT layer based on vector self-attention [42] and subtraction relation for point-wise feature aggregation in the 3D space. Different from typical scalar attention that uses dot-product results as shared weights between feature channels, vector attention adopts weight vectors to assign adapted attention weights to different channels and is proved to be more powerful in point cloud processing.

As depicted in Fig. 2, let φ, ψ, and α denote linear projections for point-wise feature transformation, δ denote the position encoding, the calculation in a PT layer can be represented as

$$\mathbf{y}_i = \sum_{\mathbf{x}_j \in \mathcal{X}(i)} \rho\left(\gamma(\varphi(\mathbf{f}_i) - \psi(\mathbf{f}_j) + \delta)\right) \odot \left(\alpha(\mathbf{f}_j) + \delta\right), \quad (1)$$

where $(\mathbf{f}_i, \mathbf{f}_j)$ are the input feature of point $(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{y}_i$ is the output feature for $\mathbf{x}_i$. $\mathcal{X}(i)$ is the set of the k-nearest neighbors of point $\mathbf{x}_i$. γ denotes an MLP for attention vector generation and ρ

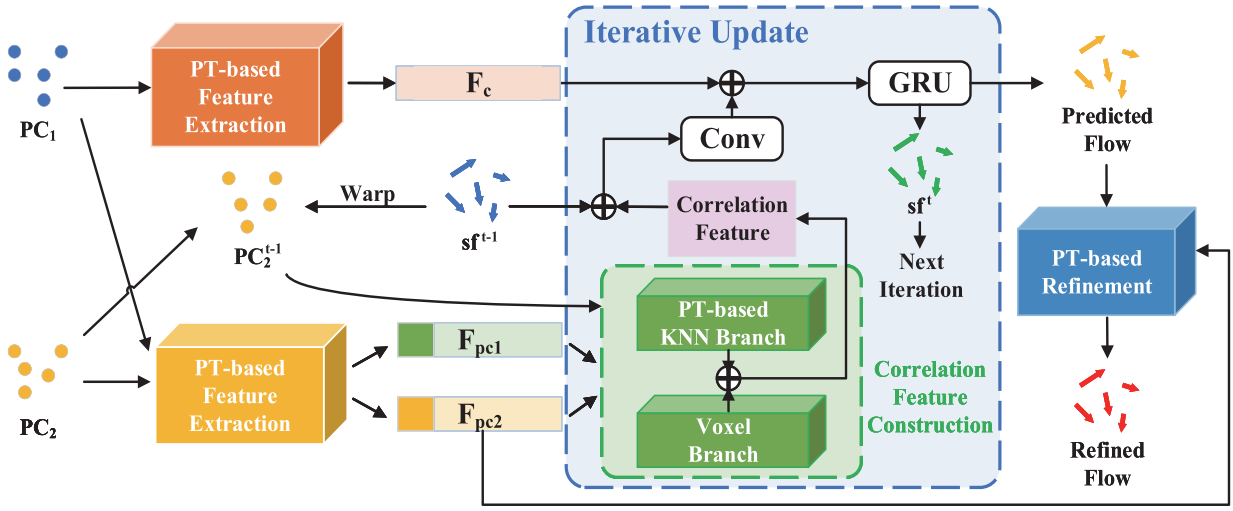

Fig. 1. The pipeline of our proposed PT-FlowNet. PC_1 and PC_2 share a PT-based feature extraction module; meanwhile, PC_1 is encoded by another PT-based module to generate contextual feature $\mathbf{F}_c$. During the tth iteration in the iterative flow update stage, the point $\mathbf{q}_j$ in PC_2 is warped to $\mathbf{q}_j^{(t-1)}$ in PC_2^{t-1} by the last flow estimation $\mathbf{sf}^{t-1}$. The point-wise feature $(\mathbf{F}_{pc1}, \mathbf{F}_{pc2})$ and the point coordinates of PC_2^{t-1} are the inputs for the correlation feature construction. The obtained correlated feature combined with $\mathbf{sf}^{t-1}$ is fed into the GRU together with $\mathbf{F}_c$. The hidden state of GRU is then updated to obtain the residual flow and the new flow prediction $\mathbf{sf}^t$ is produced for the next iteration. After T iterations, the last flow predicted from the iterative update module is fed into a PT-based refinement module to generate the final refined scene flow estimation. $\oplus$ represents concatenation.

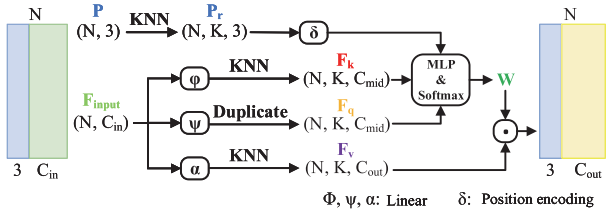

Fig. 2. The structure of the PT layer. C_{in}, C_{mid}, and C_{out} are the dimensions of the input point-wise feature, the outputs of the linear projections, and the output feature, respectively. We set the number of nearest neighbors $K = 32$ and $C_{\text{mid}} = C_{\text{out}}/2 = 32$. The input point-wise feature $\mathbf{F}_{\text{input}}$ is mapped by several linear projections to produce the key, query, and value attention vectors $\mathbf{F}_k, \mathbf{F}_q, \mathbf{F}_v$. $\mathbf{P}$ and $\mathbf{P}_r$ in the upper branch refer to the 3D coordinates and relative 3D spatial position of the points respectively, except for the refinement module (described in Section III-D).

is *softmax* for weight normalization. $\odot$ represents the Hadamard product between vectors.

B. PT-Based Feature Extraction

The primary function of the feature extraction module is to generate 128-dimensional point-wise features for the input point cloud, which are represented by $\mathbf{F}_c$, $\mathbf{F}_{pc1}$, and $\mathbf{F}_{pc2}$ in Fig. 1. As shown in Fig. 3, the input of the feature extraction module is the raw 3D coordinates of points, which are also adopted as the initialized point-wise features.

Following the design in [22], the backbone of the PT-based feature extraction module is built based on residual PT blocks, transition down and up units with U-shaped cross-layer connections, and MLPs. A transition down unit first performs Farthest-Point-Sampling [6] on the input point cloud to get a subset of points. After that, the K nearest neighbors of each subset point are selected from the original input points and their features will go through a linear transformation, followed by batch normalization and ReLU. Then these processed features are gathered by applying max-pooling to each feature dimension of the K nearest neighbors to generate aggregated point-wise feature for each subset point. Guided by the skip link, a corresponding transition up unit leads an inverse process that uses trilinear interpolation to map features from the downsampled point subset onto its superset. Specifically, the interpolated features are summarized with the features from the transition down stage.

We modulate the number of transition down and up units based on the input size of the point sets. Moreover, we add a block with a PT layer and MLPs at the end of the feature extraction module to adjust the dimension of the output features and enhance the representation ability of the network. With the transition down and transition up operations, the network can capture the structural information of objects at different scales.

C. Iterative Flow Update

We adopt the GRU-based [31] structure which enables iterative residual flow generation to obtain better flow estimation results. As a simplified variant of LSTM with fewer control gates, GRU can capture the long-term dependents of the context and produce better outputs than traditional RNNs, which are essential for the iterative scene flow update module. In addition, we suggest a novel PT-based KNN branch within correlation feature construction to reduce information loss and further strengthen the feature aggregation of the network. The procedure of iterative flow update is summarized in Fig. 1 and our improved PT-based KNN branch is illustrated in Fig. 4. During the tth iteration, the point $\mathbf{q}_j$ in PC_2 is warped to $\mathbf{q}_j^{(t-1)}$ in PC_2^{t-1} by the temporary flow estimation $\mathbf{sf}^{t-1}$ before generating the new estimation $\mathbf{sf}^t$.

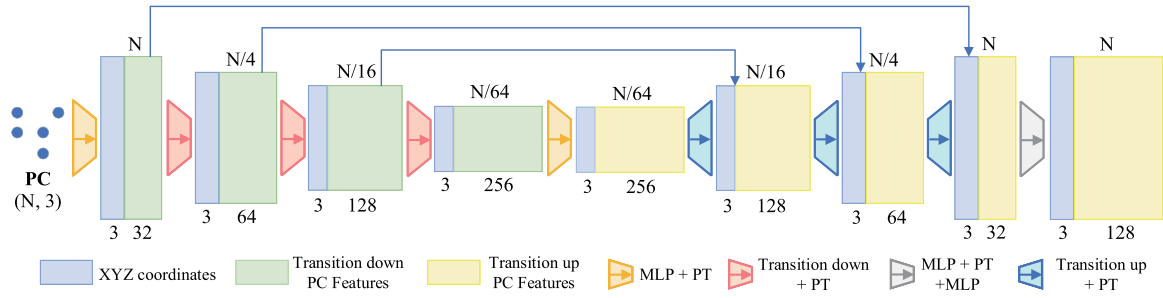

Fig. 3. PT-based point-wise feature extraction module.

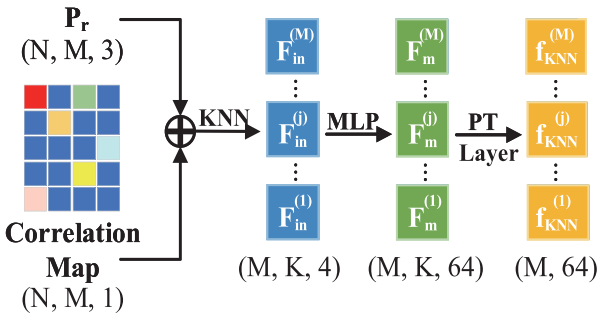

Fig. 4. PT-based KNN branch. $\mathbf{P}_r$ here is the relative 3D spatial position between the points in PC_1 and PC_2^{t-1}. $\bigoplus$ represents concatenation.

The point-wise features $(\mathbf{F}_{pc1}, \mathbf{F}_{pc2})$ generated from the feature extraction module are used for calculating a correlation map based on the matrix dot product:

$$\mathbf{C}_{corr} = \mathbf{F}_{pc1} \cdot \mathbf{F}_{pc2}, \tag{2}$$

thus point pairs with similar features have high correlation values. Based on this correlation map, two branches, i.e., voxel and KNN branch, are employed for correlation feature construction.

Voxel Branch: The voxel branch first builds a voxel neighbor cube centered around $\mathbf{q}_j^{(t-1)}$, which is a cube with $L = a \times a \times a$ sub-cubes. The side length of each sub-cube is r and $\mathcal{N}_r^{(l)}$ is the set of the PC_1 points that locate in the lth sub-cube. The correlation values of the points inside each sub-cube are averaged to produce the sub-cube feature:

$$\mathbf{f}_{sub}^{(l)} = \frac{1}{n_l} \sum_{\mathbf{p}_n \in \mathcal{N}_r^{(l)}} \mathbf{C}_{corr}(\mathbf{p}_n), \tag{3}$$

where n_l is the set size of $\mathcal{N}_r^{(l)}$. $\mathbf{C}_{corr}(\mathbf{p}_n)$ denotes the correlation value between $\mathbf{q}_j$ and $\mathbf{p}_n$, which is computed by Eq. (2). Then the output feature of $\mathbf{q}_j^{(t-1)}$ from the voxel branch is defined as

$$\mathbf{f}_{voxel}^{(j)} = \mathrm{MLP}_{voxel}\left(\mathbf{f}_{sub}^{(1)} \oplus \mathbf{f}_{sub}^{(2)} \oplus \ldots \oplus \mathbf{f}_{sub}^{(L)}\right), \tag{4}$$

where $\oplus$ represents concatenation.

PT-based KNN Branch: Let $\mathcal{N}(j)$ denote the PC_1 points that belong to the k-nearest neighbor of $\mathbf{q}_j^{(t-1)}$, the primary feature for each point $\mathbf{p}_m \in \mathcal{N}(j)$ is defined as

$$\mathbf{f}_m^{(j)} = \mathrm{MLP}_{KNN}(\mathbf{C}_{corr}(\mathbf{p}_m) \oplus (\mathbf{p}_m - \mathbf{q}_j^{(t-1)})), \tag{5}$$

where $\mathbf{f}_m^{(j)} \in \mathbb{R}^{1\times 64}$ is a KNN component of $\mathbf{F}_m^{(j)} \in \mathbb{R}^{K\times 64}$ in Fig. 4. $\mathbf{C}_{corr}(\mathbf{p}_m)$ denotes the correlation value between $\mathbf{q}_j$ and $\mathbf{p}_m$, which is computed by Eq. (2). The output feature of $\mathbf{q}_j^{(t-1)}$ from the original KNN branch [16] gathers the K point features by applying the max-pooling operation to K dimension of $\mathbf{F}_m^{(j)}$:

$$\mathbf{f}_{KNN}^{(j)} = \underset{K}{\mathrm{maxpool}}(\mathbf{F}_m^{(j)}), \tag{6}$$

The KNN branch concentrates on correlations within the small local regions, while the voxel branch has a large receptive field and can handle point pairs with large displacements, which are common in the initial stage of the iterative update. Therefore, the KNN branch largely determines the accuracy of the final prediction in the sense that the voxel branch cannot provide detailed correlations. However, the simple max-pooling operation in Eq. (6) inevitably causes large information loss and hence leads to performance degradation.

Based on the above analysis, we replace the max-pooling operation in Eq. (6) with a PT layer:

$$\mathbf{f}_{KNN}^{(j)} = \sum_{\mathbf{p}_m \in \mathcal{N}(j)} \rho\left(\gamma(\varphi(\mathbf{f}_m^{(j)}) - \psi(\mathbf{f}_m^{(j)}) + \delta)\right) \odot \left(\alpha(\mathbf{f}_m^{(j)}) + \delta\right). \tag{7}$$

Different from the original KNN branch, our PT-based KNN branch utilizes the transformer to adaptively aggregate the local features. Compared with the KNN branch equipped with max-pooling, the PT-based KNN branch has less information loss and is more robust to the structural distortions in the iterative process.

Following [16], the correlation feature from the voxel and KNN branch are then concatenated with $\mathbf{sf}^{t-1}$ and $\mathbf{F}_c$, which is taken as input x_t for the GRU cell [31] to produce the hidden state h_t:

$$z_t = \sigma(\mathrm{Conv}_{1\,d}([h_{t-1}, x_t], W_z)) \tag{8}$$

$$r_t = \sigma(\mathrm{Conv}_{1\,d}([h_{t-1}, x_t], W_r)) \tag{9}$$

$$\hat{h}_t = \tanh(\mathrm{Conv}_{1\,d}([r_t \odot h_{t-1}, x_t], W_h)) \tag{10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{11}$$

where W_z, W_r, and W_h are the weight matrices that can be learned during training. Finally, the obtained hidden state h_t is

fed into convolutional layers to generate the predicted scene flow $\mathbf{sf}^t$.

D. PT-Based Flow Refinement

The PT-based flow refinement module adopts the same backbone structure as the point-wise feature extraction module and is trained independently. The difference is that the flow refinement module takes the final predicted flow from the iterative update stage and the point-wise feature $\mathbf{F}_{pc2}$ as input, as shown in Fig. 1. These two inputs also replace $\mathbf{P}$ and $\mathbf{F}_{input}$ in the PT layer shown in Fig. 2 respectively. The consequent $\mathbf{P}_r$ in Fig. 2 becomes the difference between point-wise flow vectors. The main function of the refinement module is to promote flow smoothness in the 3D space by transformer-based flow aggregation. In the existence of occlusion, the points in the two scenes may not match each other completely. In this case, the PT-based flow refinement can produce reasonable flow estimation for the occluded points from their neighbors. Besides, the points with similar features and closer positions in space are more likely to produce similar scene flow predictions through PT-based operations. This encourages flow consistency of the points collected from the same object and improves the network performance thereby.

E. Loss Function

Our model is trained under full supervision on annotated synthetic FlyingThings3D dataset [43]. Following the hierarchical design in [15] and [16], we accumulate the loss from every iteration as the supervised loss for the feature extraction and the iterative update module. We also use the l_1-norm loss for the flow refinement stage.

The loss function for the sub-network with the first two modules is

$$\mathcal{L}_{iter} = \sum_{t=1}^{T} w^{(t)} \|(\mathbf{sf}_{est}^{(t)} - \mathbf{sf}_{gt})\|_1 \tag{12}$$

with

$$w^{(t)} = \gamma * (T - t - 1), \tag{13}$$

where $\mathbf{sf}_{est}^{(t)}$ is the flow estimated from the tth iteration in the iterative flow update stage, and $\mathbf{sf}_{gt}$ denotes the ground truth. T is the maximum number of iterations and $w^{(t)}$ is the weight for tth iteration. We adopt the hyper-parameter setting in [16] with $\gamma = 0.8$.

The loss for the refinement module is

$$\mathcal{L}_{ref} = \|(\mathbf{sf}_{ref} - \mathbf{sf}_{gt})\|_1, \tag{14}$$

where $\mathbf{sf}_{ref}$ is the final refined flow prediction.

IV. Experiments

A. Datasets

As the first benchmark for scene flow estimation, FlyingThings3D (FT3D) [43] is a large-scale synthetic dataset composed of 19,640 pairs of labeled training samples and 3,824 samples in the testing set. Each sample contains stereo and RGB-D images of the moving synthetic objects selected from ShapeNet [44]. With the camera parameters and disparities provided in FT3D, the previous methods [10], [11], [12], [15] generate 3D point clouds and ground truth flows by lifting the 2D data to 3D points. KITTI Scene Flow 2015 [45] is another benchmark with 142 available real-world samples. For a fair comparison with other methods, we follow the above-mentioned data pre-processing and conduct experiments on both FT3D and KITTI datasets established by Gu et al. [10]. Note that the points with a depth of more than 35 m or a height of less than 0.3 m are also removed. Following [16], we select 2000 samples from the FT3D training set for validation and use the remains for training. Our model is only trained on synthetic FT3D dataset and is applied to real-world KITTI scans without fine-tuning.

B. Evaluation Metrics

We adopt the same evaluation metrics used in recent works [10], [11], [12], [15], [18] for performance comparison.

- *EPE3D (m):* $\|\mathbf{sf}_{est} - \mathbf{sf}_{gt}\|_2$ 3D end-point-error averaged over non-occluded points.
- *ACC3D Strict:* A strict version of 3D accuracy, the percentage of points whose EPE3D < 0.05 m or relative error $< 5\%$.
- *ACC3D Relax:* A relaxed version of 3D accuracy, the percentage of points whose EPE3D < 0.1 m or relative error $< 10\%$.
- *Outliers3D:* The percentage of points whose EPE3D $>$ 0.3 m or relative error $> 10\%$.
- *EPE2D (px):* 2D end-point-error measured by projecting points back to the 2D image plane, which is a common metric for optical flow evaluation.
- *ACC2D:* The percentage of points whose EPE2D $<$ 3px or relative error $< 5\%$.

C. Implementation Details

We implement PT-FlowNet in PyTorch [46]. For a fair comparison with the previous works, we randomly sample $N = M = 8,192$ input points from each point cloud as the input of our network. Following the correlation module settings in [16], $k = 32$ nearest neighbors are selected for the point branch; 3-level cubes with cube resolution $a = 3$ and side length $r = 0.25, 0.5, 1$ are built for the voxel branch. The total amounts of iterations are $T = 8$ and 32 for training and evaluation, respectively. The whole network other than the refinement module is first trained for 50 epochs and the weights are fixed afterward. Then the refinement module is integrated and trained for another 10 epochs. We use the Adam optimizer [47] with an initial learning rate of 0.001.

D. Comparison to State-of-The-Art (SOTA)

We report the performance of our PT-FlowNet compared to the state-of-the-art methods on both FT3D and KITTI datasets described in Section IV-A. The evaluation results are shown in

TABLE I
THE QUANTITATIVE RESULTS ON THE FT3D AND KITTI DATASETS

Dataset	Method	EPE3D (m) ↓	ACC3D Strict ↑	ACC3D Relax ↑	Outliers 3D ↓	EPE2D (px) ↓	ACC 2D ↑
FT3D	ICP [48]	0.4062	0.1614	0.3038	0.8796	23.2280	0.2913
	FlowNet3D [9]	0.1136	0.4125	0.7706	0.6016	5.9740	0.5692
	HPLFlowNet [10]	0.0804	0.6144	0.8555	0.4287	4.6723	0.6764
	PointPWC [11]	0.0588	0.7379	0.9276	0.3424	3.2390	0.7994
	FLOT [12]	0.0520	0.7320	0.9270	0.3570	-	-
	HALFlow [27]*	0.0492	0.7850	0.9468	0.3083	2.7555	0.8111
	HCRF-Flow [18]	0.0488	0.8337	0.9507	0.2614	2.5652	0.8704
	PV-RAFT [16]	0.0461	0.8169	0.9574	0.2924	-	-
	FlowStep3D [15]	0.0455	0.8162	0.9614	0.2165	-	-
	RMS-FlowNet [19]	0.0560	0.7920	0.9550	0.3240	-	-
	RCP [20]	0.0403	0.8567	0.9635	0.1976	-	-
	SCTN [29]	0.0380	0.8470	0.9680	0.2680	-	-
	Res3DSF [28]*	0.0310	0.9139	0.9768	**0.1551**	1.7504	0.9113
	Ours	**0.0304**	**0.9142**	**0.9814**	0.1735	**1.6150**	**0.9312**
KITTI	ICP [48]	0.5181	0.0669	0.1667	0.8712	27.6752	0.1056
	FlowNet3D [9]	0.1767	0.3738	0.6677	0.5271	7.2141	0.5093
	HPLFlowNet [10]	0.1169	0.4783	0.7776	0.4103	4.8055	0.5938
	PointPWC [11]	0.0694	0.7281	0.8884	0.2648	3.0062	0.7673
	FLOT [12]	0.0560	0.7550	0.9080	0.2420	-	-
	HALFlow [27]*	0.0622	0.7649	0.9026	0.2492	2.5140	0.8128
	HCRF-Flow [18]	0.0531	0.8631	0.9444	0.1797	2.0700	0.8656
	PV-RAFT [16]	0.0560	0.8226	0.9372	0.2163	-	-
	FlowStep3D [15]	0.0546	0.8051	0.9254	0.1492	-	-
	RMS-FlowNet [19]	0.0530	0.8180	0.9380	0.2030	-	-
	RCP [20]	0.0481	0.8491	0.9448	0.1228	-	-
	SCTN [29]	0.0370	0.8730	0.9590	0.1790	-	-
	Res3DSF [28]*	0.0351	0.8932	0.9620	0.1654	1.2879	0.9442
	Ours	**0.0224**	**0.9551**	**0.9838**	**0.1186**	**0.9893**	**0.9667**

* HALFlow and Res3DSF take $N = M = 8,192$ points as input but only output scene flow for $2,048$ sampled points; other methods produce scene flow for all $8,192$ input points.
All methods are trained on FT3d in a supervised manner.

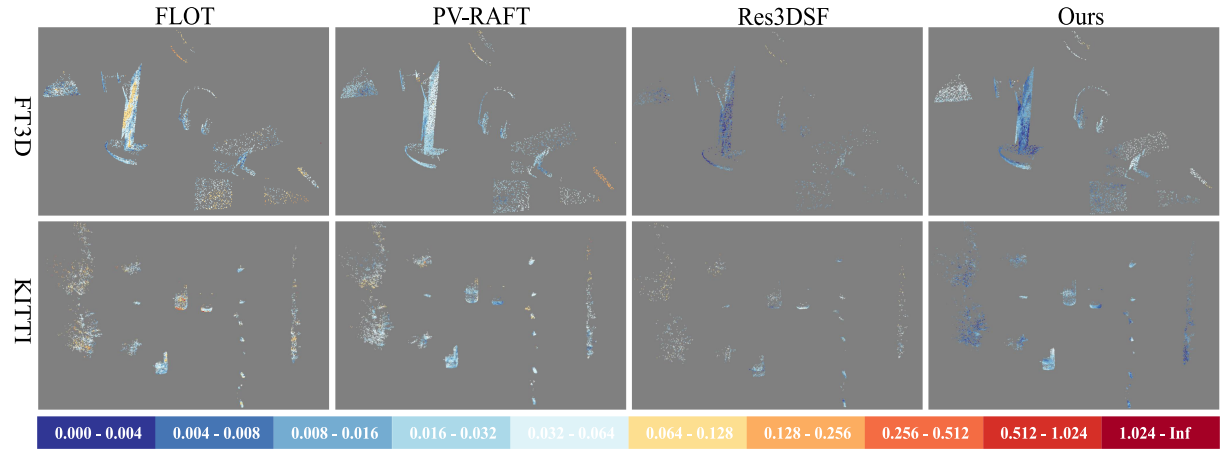

Fig. 5. The qualitative comparison of different methods on FT3D (top) and KITTI (bottom) datasets. From left to right: The EPE3D (m) error map of FLOT [12], PV-RAFT [16], Res3DSF [28], and our proposed method. Res3DSF [28] estimates scene flow for one quarter of the original input points, so its point clouds are more sparse in the visualization results. Compared with other methods, our PT-FlowNet shows lower errors especially for real-world scenes in the KITTI dataset.

Table I and the qualitative comparison results are presented in Fig. 5.

Evaluation on FT3D: According to Table I, our method outperforms recent state-of-the-art approaches. Although Res3DSF [28] shows comparable results on FT3D, it generates more sparse scene flow for only a quarter of the points down-sampled from the input point cloud, as shown in Fig. 5. When compared to SCTN [29] which also includes point transformer for capturing point relation information in the feature extraction stage, we report a relative improvement of 20.0% on the EPE3D metric. The evaluation results confirm the effectiveness of our method to predict scene flow accurately.

Generalization on KITTI: The generalization ability of our method is evaluated by directly applying the model trained on the synthetic FT3D datasets to real-world point cloud frames in KITTI without fine-tuning. The results listed in Table I

TABLE II
THE ABLATION STUDY RESULTS ON THE FT3D AND KITTI DATASETS

Dataset	Feature Extraction	KNN Branch	Refinement Module	EPE3D (m) ↓	ACC3D Strict ↑	ACC3D Relax ↑	Outliers 3D ↓	EPE2D (px) ↓	ACC 2D ↑
FT3D	Pointnet++	with PT	-	0.0508	0.7461	0.9442	0.3484	2.6466	0.8413
	PT	w/o PT	-	0.0362	0.8714	0.9760	0.2454	1.8709	0.9050
	PT	with PT	-	0.0332	0.8944	0.9783	0.1968	1.7756	0.9191
	PT	with PT	conv	0.0312	0.9065	0.9799	0.1769	1.6671	0.9282
	PT	with PT	PT†	0.0308	0.9132	**0.9815**	0.1776	1.6559	0.9298
	PT	with PT	PT	**0.0304**	**0.9142**	0.9814	**0.1735**	**1.6150**	**0.9312**
KITTI	Pointnet++	with PT	-	0.0392	0.8693	0.9604	0.1888	2.0143	0.8865
	PT	w/o PT	-	0.0338	0.9095	0.9746	0.1588	1.4579	0.9366
	PT	with PT	-	0.0261	0.9499	0.9796	0.1349	1.3083	0.9499
	PT	with PT	conv	0.0239	0.9518	0.9780	0.1304	1.1730	0.9584
	PT	with PT	PT†	0.0230	0.9544	0.9806	0.1228	1.1278	0.9630
	PT	with PT	PT	**0.0224**	**0.9551**	**0.9838**	**0.1186**	**0.9893**	**0.9667**

† We also report the scores from the network where the refinement module is trained without fixing the parameters of the preceding sub-network.

demonstrate that our method achieves the leading position on KITTI and is the first to reduce the EPE3D metric below 3 cm, with a surprising 36.2% improvement over the state-of-the-art method Res3DSF [28]. In addition, our proposed method surpasses the transformer-based SCTN [29] with an error reduction of 39.5%. We believe that the PT-based operations in each processing stage promote efficient information sharing between points and allow the network to better adapt to the different motion patterns of the objects.

E. Ablation Study

We verify the effectiveness of each module by simply removing the PT-based components or replacing them with other components used in previous works. The detailed results are shown in Table II.

Feature Extraction: We compare our PT-based feature extraction module with the PointNet++ network used in [12] for point-wise feature abstraction. According to the experimental results in Table II, using PT-based point cloud abstraction brings significant gain to all evaluation metrics. In particular, EPE3D is reduced by 34.6% and 33.4% on FT3D and KITTI, respectively. The quantitative results strongly demonstrate that our PT-based network generates point-wise features with higher quality and is a more powerful embedding method for point sets.

KNN Branch: We validate the significance of the PT layer in the KNN branch by rolling back to the original max-pooling operation in [16] and then evaluating the network performance after this alteration. According to the EPE3D metric shown in Table II, the performance of the network on the FT3D dataset has been enhanced by 8.3% and the prediction accuracy on the KITTI dataset has been improved by 22.8% after the PT-based KNN branch is applied.

Flow Refinement Module: Both the convolutional layers adopted in [16] and our PT-based refinement module are separately applied to our framework for comparison. The experimental results show that the methods with the refinement module perform better than those without it, and our PT-based refinement module exhibits superior performance over the convolutional refinement method. Furthermore, we notice that slightly better results are obtained when the parameters of the sub-network are fixed rather than unfixed during the flow refinement step. The intuitive explanation is that the iterative flow update within the sub-network is the main step to generate scene flow, and the PT-based flow refinement module is a relatively independent component that improves the flow smoothness in the space based on the existing scene flow estimation results.

TABLE III
THE RUNNING TIME COMPARISON BETWEEN THE OFFICIAL IMPLEMENTATION OF PV-RAFT [16] AND OUR PT-FLOWNET

Method	$T = 8$	$T = 32$
PV-RAFT [16]	292.8 ms	740.0 ms
PT-FlowNet (ours)	354.7 ms	898.5 ms

F. Runtime

As shown in Table III, we measure the running time of our PT-FlowNet and the most related work PV-RAFT [16], which also involves iterative flow estimation and flow refinement. Evaluated on a single NVIDIA A40 GPU with $N = M = 8,192$ input points, our PT-based method consumes relatively higher running time than PV-RAFT.

V. CONCLUSION

In this letter, we propose PT-FlowNet - the first full-stage PT-based deep learning architecture for accurate scene flow estimation in an end-to-end manner. We design PT-based modules for each flow prediction stage to enable effective point-level feature aggregation and correlation. The embedded point transformer helps aggregate point features with different scales in each step and produce accurate prediction results in irregular real data with occlusion. Experimental results on both synthetic and real-world datasets demonstrate that our method outperforms previous works and reaches state-of-art performance. The success of our method also suggests the promising potential of the PT-based attention mechanism in point cloud processing. One of the future directions could be improving the real-time performance of the method to make it more applicable to robotic tasks.

REFERENCES

[1] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual slam and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 1290–1297.

[2] Y. Zhang, Y. Ye, Z. Xiang, and J. Gu, "SDP-Net: Scene flow based real-time object detection and prediction from sequential 3D point clouds," in *Proc. 15th Asian Conf. Comput. Vis.*, 2020, pp. 140–157.

[3] A. Behl, D. Paschalidou, S. Donné, and A. Geiger, "PointFlowNet: Learning representations for rigid motion estimation from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7962–7971.

[4] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held, "ToolFlowNet: Robotic manipulation with tools via predicting tool flow from point clouds," in *Proc. Conf. Robot Learn.*, 2023, pp. 1038–1049.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[6] 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. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[7] H. Su et al., "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2530–2539.

[8] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9613–9622.

[9] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.

[10] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, "HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3249–3258.

[11] W. Wu, Z. Y. Wang, Z. Li, W. Liu, and L. Fuxin, "PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 88–107.

[12] G. Puy, A. Boulch, and R. Marlet, "FLOT: Scene flow on point clouds guided by optimal transport," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 527–544.

[13] I. Tishchenko, S. Lombardi, M. R. Oswald, and M. Pollefeys, "Self-supervised learning of non-rigid residual flow and ego-motion," in *Proc. 3DV*, 2020, pp. 150–159.

[14] G. Wang, C. Jiang, Z. Shen, Y. Miao, and H. Wang, "SFGAN: Unsupervised generative adversarial learning of 3D scene flow from the 3D scene self," *Adv. Intell. Syst.*, vol. 4, no. 4, 2022, Art. no. 2100197.

[15] Y. Kittenplon, Y. C. Eldar, and D. Raviv, "FlowStep3D: Model unrolling for self-supervised scene flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4112–4121.

[16] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "PV-RAFT: Point-voxel correlation fields for scene flow estimation of point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6950–6959.

[17] B. Ouyang and D. Raviv, "Occlusion guided scene flow estimation on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2799–2808.

[18] R. Li, G. Lin, T. He, F. Liu, and C. Shen, "HCRF-Flow: Scene flow from point clouds with continuous high-order CRFs and position-aware flow embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 364–373.

[19] R. Battrawy, R. Schuster, M. N. Mahani, and D. Stricker, "RMS-FlowNet: Efficient and robust multi-scale scene flow estimation for large-scale point clouds," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 883–889.

[20] X. Gu, C. Tang, W. Yuan, Z. Dai, S. Zhu, and P. Tan, "RCP: Recurrent closest point for point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8206–8216.

[21] W. Cheng and J. H. Ko, "Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2022, pp. 108–124.

[22] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 16259–16268.

[23] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.

[24] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16949–16958.

[25] Y. Xia et al., "SOE-Net: A self-attention and orientation encoding network for point cloud based place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11343–11352.

[26] H. Wang, J. Pang, M. A. Lodhi, Y. Tian, and D. Tian, "FESTA: Flow estimation via spatial-temporal attention for scene point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14168–14177.

[27] G. Wang, X. Wu, Z. Liu, and H. Wang, "Hierarchical attention learning of scene flow in 3D point clouds," *IEEE Trans. Image Process.*, vol. 30, pp. 5168–5181, 2021.

[28] G. Wang, Y. Hu, X. Wu, and H. Wang, "Residual 3-D scene flow learning with context-aware feature extraction," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–9, 2022.

[29] B. Li, C. Zheng, S. Giancola, and B. Ghanem, "SCTN: Sparse convolution-transformer network for scene flow estimation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 2, 2022, pp. 1254–1262.

[30] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 402–419.

[31] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. 8th Workshop Syntax, Semantics Struct. Stat. Transl.*, 2014, pp. 103–111.

[32] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6410–6419.

[33] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5588–5597.

[34] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.

[35] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, "VV-Net: Voxel VAE net with group convolutions for point cloud segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8499–8507.

[36] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-Temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3070–3079.

[37] R. Cheng, R. Razani, Y. Ren, and L. Bingbing, "S3Net: 3D LiDAR sparse semantic segmentation network," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 14040–14046.

[38] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5560–5568.

[39] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9267–9276.

[40] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1708–1716.

[41] H. Zhou, Y. Feng, M. Fang, M. Wei, J. Qin, and T. Lu, "Adaptive graph convolution for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4945–4954.

[42] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10073–10082.

[43] N. Mayer et al., "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4040–4048.

[44] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," *arXiv:1512.03012*, 2015.

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[46] A. Paszkeet al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[48] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," *Proc. SPIE*, vol. 1611, 1992, pp. 586–606.