Multimedia Tools and Applications (2024) 83:7317–7334
https://doi.org/10.1007/s11042-023-15541-4

Learning graph-based representations for scene flow estimation

Mingliang Zhai[1] · Hao Gao[1] · Ye Liu[1] · Jianhui Nie[1] · Kang Ni[2]

Received: 19 March 2022 / Revised: 11 July 2022 / Accepted: 19 April 2023 /
Published online: 7 June 2023

Abstract
Scene flow estimation is a fundamental task of autonomous driving. Compared with optical flow, scene flow can provide sufficient 3D motion information of the dynamic scene. With the increasing popularity of 3D LiDAR sensors and deep learning technology, 3D LiDAR-based scene flow estimation methods have achieved outstanding results on public benchmarks. Current methods usually adopt Multiple Layer Perceptron (MLP) or traditional convolution-like operation for feature extraction. However, the characteristics of point clouds are not exploited adequately in these methods, and thus some key semantic and geometric structures are not well captured. To address this issue, we propose to introduce graph convolution to exploit the structural features adaptively. In particular, multiple graph-based feature generators and a graph-based flow refinement module are deployed to encode geometric relations among points. Furthermore, residual connections are used in the graph-based feature generator to enhance feature representation and deep supervision of the graph-based network. In addition, to focus on short-term dependencies, we introduce a single gate-based recurrent unit to refine scene flow predictions iteratively. The proposed network is trained on the FlyingThings3D dataset and evaluated on the FlyingThings3D, KITTI, and Argoverse datasets. Comprehensive experiments show that all proposed components contribute to the performance of scene flow estimation, and our method can achieve potential performance compared to the recent approaches.

Keywords Deep learning · Scene flow estimation · Graph convolutional networks · 3D point cloud · Scene understanding

1 Introduction

Scene flow represents the 3D motion vector of every point in 3D space. It has been widely applied in many real-world artificial intelligence tasks, such as autonomous driving [1], robot [29], and action recognition [43]. Due to the increasing popularity of 3D LiDAR data

✉ Mingliang Zhai
zhaimingliang@njupt.edu.cn

Extended author information available on the last page of the article.

and deep learning technique [33, 36], scene flow estimation from point cloud sequence has shown impressive accuracy on public scene flow estimation benchmarks.

Based on the pioneering optical flow model FlowNet [5], FlowNet3D [21] and HPLFlowNet [6] adopt U-Net architecture for learning 3D motion fields in an end-to-end fashion. In contrast to [6, 21], PointPWC-Net [50] designs a spatial pyramid network for 3D motion estimation, which can adaptively refine the flow fields in a coarse-to-fine strategy. However, these approaches commonly rely on multi-scale feature extraction, which may neglect local shapes and semantic information. To address this issue, FLOT [31] introduces an optimal transport module into the scene flow estimation network, which can calculate transport cost between two points by constructing the pairwise similarity among deep features. Although FLOT achieves promising performance on public benchmarks due to the accurate all-pairs matching results produced by the optimal transport module, it only considers local correlations and ignores long-range correlations. To solve this problem, PV-RAFT [47] designs a point-voxel correlation module for feature matching, which can capture both local and global correlations from two adjacent point clouds. Moreover, PV-RAFT adopts Gated Recurrent Unit (GRU) as an iterative refinement module for updating flow fields. Because of constructing the point-voxel correlation and adopting the RAFT framework [39], PV-RAFT obtains competitive results on public benchmarks. Nevertheless, it still has some challenging drawbacks. First, PV-RAFT relies on SetConv (PointNet-like operation [32]) to extract local features, which may lose the structural information of the point cloud. Second, the PointNet-like backbone for feature and context extraction is not so powerful, which leads to the unstable performance of feature learning and representation. Third, the performance of PV-RAFT is limited by redundant gate units in GRU-based iterator. Since the inputs of PV-RAFT are only two adjacent point clouds, it is more important to focus on short-term dependencies.

Recently, graph neural networks have shown promising performance on several 3D visual tasks, such as 3D object detection [3, 37], point cloud generation [16], point cloud classification and segmentation [17, 19, 45]. These works have proven the effectiveness of learning graph-based representation for point clouds processing. In particular, Wang et al. [45] apply graph convolutional networks to point clouds and propose dynamic graph convolutional networks for 3D semantic segmentation, effectively capturing the relationship of points and local geometric structure. Based on [45], Lin et al. [19] propose deformable 3D graph convolutional networks for point cloud classification and semantic segmentation, which can learn 3D deformable kernels with graph convolutional operation for extracting geometric features from point cloud across different scales. Compared to PointNet-like or traditional convolution-based backbones, graph convolutional networks can extract locally structural and geometric information from point clouds. Moreover, graph convolutional networks can build complex hierarchies of features by dynamically constructing neighborhood graphs from similarity among the high-dimensional feature representations of the points properly. However, few works have been proposed to investigate the effect of the graph-based representation for scene flow estimation.

Motivated by the above analysis, this paper proposes a novel graph-based deep neural network for scene flow estimation. The objective of our research work can be concluded as two aspects. On the one hand, we aim to introduce graph convolution to explore the geometric structures of point clouds. On the other hand, we aim to introduce an efficient single gated-based flow refinement module to focus on short-term dependencies between two adjacent point clouds. In comparison with previous work, our main contributions can be summarized as follows:

- To exploit graph-based representation and investigate the effect of graph convolutional networks for scene flow estimation, we propose a graph-based feature extractor to learn graph-based features, which can capture geometric structures by modeling the relative dynamic positions of points. Furthermore, we introduce a graph-based flow refinement module to refine scene flow fields in 3D space.
- We construct a skip connection in the graph-based feature extractor, which can enhance the graph-based network's feature representation and deep supervision. Moreover, instead of using a GRU-based iterator, we introduce a single gate-based flow iterator to focus on short-term dependencies between two adjacent point clouds.
- Comprehensive experiments on public scene flow benchmarks demonstrate that the proposed method performs substantially better than most existing approaches and achieves potential performance on FlyingThings3D, KITTI, and Argoverse datasets compared to the recent non-learning method [18] and deep learning-based methods [6, 21, 31, 38, 40, 46, 47, 50].

The rest of this paper is organized as follows: Section 2 reviews the related work of scene flow estimation and graph neural networks. Section 3 mainly describes the proposed method, including the network architecture and the training strategy. Section 4 reports the experimental results on public scene flow datasets. Section 5 concludes this paper.

2 Related work

In this section, we first review the references on scene flow estimation from binocular images. Then, we describe the references on scene flow estimation from monocular images. Finally, we mainly review the literature on scene flow estimation from point clouds.

2.1 Scene flow from binocular images

Binocular stereo matching is an effective technique to recover three-dimensional information, which can calculate the disparity in the binocular image of the same object at different positions. The core idea of the scene flow estimation method based on binocular image data is to estimate the 3D motion vector by combining the optical flow and disparity information.

Traditional methods usually use multiple prior assumptions to model an energy function for optimization. Huguet et al. [9] propose the binocular scene flow estimation framework, which models an energy function by constructing the spatial and temporal correlation in binocular image sequences. Based on [9], a series of methods [25, 27, 35, 42] are proposed to overcome challenging problems, such as large displacement estimation and motion edge preservation. However, optimizing an energy function is usually time-consuming and unsuitable for real-world applications.

To address these problems, some recent works adopt deep learning technology for scene flow estimation. Ilg et al. [12] propose a supervised multi-branch learning framework for scene flow estimation. However, the supervised learning framework relies on a large number of labeled synthetic images as training data, resulting in unstable performance in real scenes. To solve this problem, many works [14, 15, 20, 23] adopt the unsupervised framework to learn scene flow from unlabeled data. Compared with traditional methods, deep learning-based methods can effectively use deep neural networks to learn the scene flow from a large amount of training data, which improves the efficiency and accuracy of scene flow estimation. However, these methods cannot estimate scene flow in 3D space directly.

2.2 Scene flow from monocular images

With the popularity of depth sensor Kinect, many works attempt to estimate scene flow from monocular image sequences. Traditional methods [7, 8, 34] often formulate optical flow estimation as an energy minimization problem. However, due to the complex optimization process, these methods are unsuitable for real-world applications.

Recent deep learning-based methods usually build a multi-task learning framework for scene flow estimation. Zhou et al. [53] propose a multiple branch architecture for joint learning of depth and camera motion, which can recover the rigid motion from a monocular image sequence. However, real-world scenes often include both rigid and non-rigid scenes. To solve this problem, Yin et al. [52] propose a cascade architecture composed of two stages to adaptively estimate the rigid and non-rigid motion in the dynamic scene. Inspired by [52], many methods [10, 11, 22, 51, 54] are proposed to improve the performance of scene flow estimation by designing a more robust network architecture and loss function. Compared with traditional methods, deep learning-based methods can perform efficiently on graphic computing units. However, these methods rely on 2D representation, which cannot handle the point cloud data.

2.3 Scene flow from point clouds

With the wide availability of commodity LiDAR sensors, scene flow estimation from point cloud has become an active field of research. Traditional methods [4, 41] usually formulate the problem of scene flow estimation for LiDAR sensors as an energy minimization problem. However, optimizing a complex objective function is time-consuming in practical application.

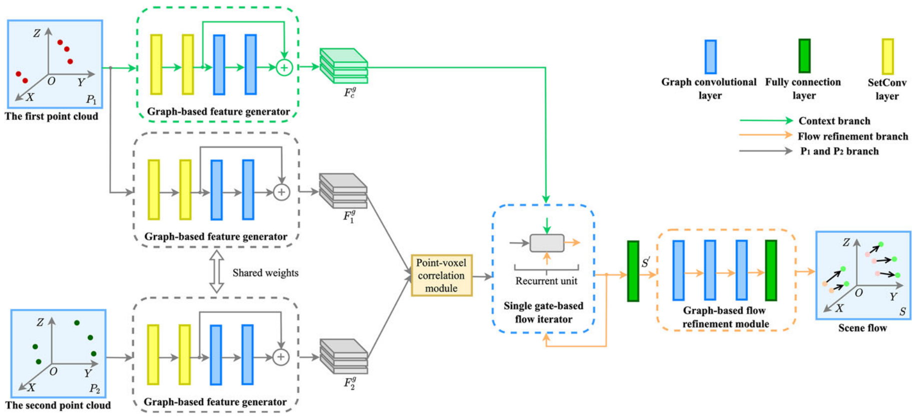
Recently, many researchers have attempted to estimate scene flow from point clouds. Based on FlowNet [5], FlowNet3D [21] introduces an end-to-end supervised network for scene flow estimation from point clouds, which adopts the PointNet-like architecture [32] as a backbone to learn the hierarchical feature. FlowNet3D++ [44] designs point-to-plane and cosine distance losses for constructing geometric constraints, improving the convergence speed and stability of training. HASF [46] introduces hierarchical attention learning architecture into scene flow estimation, which aggregates the feature of adjacent points using a dual attention mechanism. SPLATNet [38] presents a Sparse Lattice Networks (SLN) for point cloud processing. Motivated by [38], HPLFlowNet [6] adopts multiple Bilateral Convolutional Layers (BCL) [13] as a feature extractor, which can process general unstructured point cloud efficiently. PointPWC-Net [50] adopts multiple point convolutional layers [49] as a feature extractor and designs a spatial pyramid architecture for scene flow refinement in a coarse-to-fine strategy. However, the feature extractors of [6, 21, 44, 46, 50] rely on point cloud down-sampling to reduce the size of inputs, which may lose some important details in a scene. To address this issue, instead of using a down-sampling operation, FLOT [31] keeps the size of features as input size and introduces an optimal transport module into a network, which can find all-pairs correspondences between two adjacent point clouds. NSFP [18] uses a sample MLP to infer scene flow priors, which can optimize neural networks at execution time. EgoFlow [40] presents a cascade architecture for decoupling non-rigid flow and ego-motion flow in dynamic 3D scenes. To enlarge the search range of feature matching, PV-RAFT [47] uses a point-voxel correlation module to extract long-range correlations. However, exiting methods ignore to exploit the effectiveness of graph-based representations and lack enough geometric and structural information for constructing rich graph-based information.

2.4 Graph neural networks

Recently, graph neural networks have achieved potential performance on many tasks [3, 16, 17, 19, 37, 45, 48]. Wu et al. [48] propose a semi-supervised multi-view graph convolutional network for webpage classification, which can adaptively extract graph structures via graph convolution. DGCNN [45] introduces a graph-based convolutional layer named EdgeConv for point cloud segmentation, which can construct graphs dynamically and extract graph-based structural features. Chen et al. [3] propose a hierarchical graph network for 3D object detection, effectively capturing and aggregating local shape information, multi-level semantics, and global scene information. HSGAN [16] designs a hierarchical graph learning network for the point cloud generation, which captures the graph-based structural information in a hierarchical strategy. To overcome shift and scale changes, 3DGCN [19] designs a deformable graph convolutional network for 3D point cloud processing, which can extract structural features using deformable kernel size. Due to the exploitation of graph-based information, the work [19] achieves promising results on several public 3D classification and segmentation benchmarks. However, there are few works to investigate the effectiveness of graph-based representations on scene flow estimation.

3 Proposed method

In this section, we first describe the network architecture of our method. Then, we give the details of the supervised learning strategy. Figure 1 shows the architecture of our proposed network. Given two adjacent point clouds $P_1 \in \mathbb{R}^{3\times N}$ and $P_2 \in \mathbb{R}^{3\times N}$, the proposed network can estimate scene flow $S \in \mathbb{R}^{3\times N}$ in an end-to-end fashion.



The first point cloud
P_1
Graph-based feature generator
F_c^g
Graph-based feature generator
F_1^g
Shared weights
Graph-based feature generator
F_2^g
The second point cloud
P_2
Point-voxel correlation module
Recurrent unit
Single gate-based flow iterator
S'
Graph-based flow refinement module
Scene flow
S
Graph convolutional layer
Fully connection layer
SetConv layer
Context branch
Flow refinement branch
P1 and P2 branch

Fig. 1 The architecture of our proposed network. The entire architecture contains four main components: graph-based feature generator, point-voxel correlation module, single gate-based flow iterator, and graph-based flow refinement module

3.1 Network architecture

3.1.1 Graph Convolutional Layer (GCL)

Graph convolution is a kind of general convolution. Compared to convolution used for image processing (formatted in a 2D grid), graph convolution is more suitable for point cloud data (formatted in 3D space). Motivated by graph convolutional networks [19], we use the GCL to construct a graph-based feature generator, which can adaptively exploit the structural information of the point cloud.

Given a point cloud (N points) $P = \{p_i | i = 1, ..., N\}$, the receptive field of p_i can be defined as

$$O_i^Q = \{p_i, p_j | \forall p_j \in \kappa(p_i, Q)\}, \tag{1}$$

where p_j denotes the neighboring point of p_i in the receptive field O_i^Q, and $\kappa(p_i, Q)$ denotes Q nearest neighbors of pi.

In the receptive field O_i^Q, the directional vector can be defined as

$$v_{j,i} = p_j - p_i, \tag{2}$$

where $v_{j,i}$ denotes the directional vector between p_i and p_j.

The 3D kernel of GCL can be defined as

$$E^L = \{e', e_1, e_2, ..., e_L\}, \tag{3}$$

where $e' = (0, 0, 0)$ denotes the center of the kernel, L denotes the number of supports in E^L, and e_1 to e_L denote the associated supports in E^L.

In the 3D kernel E^L, due to $e' = (0, 0, 0)$, the directional vector can be defined as $e_l = e_l - e'$, $l = 1, 2, ..., L$. For e_l, we define the weight vector $w(e_l)$ to achieve the convolution operation on feature $f(p_i)$.

Based on the above definition, the receptive field O_i^Q and the 3D kernel E^L can be seen as standard graphs. Graph node is represented by point feature $f(p_i)$ or weight vector $w(e_l)$. Graph edge is represented by the directional vector $v_{j,i}$ or e_l. Thus, the graph convolution can be defined as

$$GConv(O_i^Q, E^L) = \langle f(p_i), w(e') \rangle + \sum_{l=1}^{L} \max_{j \in (1,Q)} \{sim(p_j, e_l)\}, \tag{4}$$

where $\langle \cdot \rangle$ denotes the inner-product operation, and sim denotes the similarity function. The sim function can be defined as

$$sim(p_j, e_l) = \langle f(p_j), w(e_l) \rangle + \frac{\langle v_{j,i}, e_l \rangle}{\| v_{j,i} \|_2 \| e_l \|_2}, \tag{5}$$

where $\| \cdot \|_2$ denotes the L2 norm.

3.1.2 Graph-based feature generator

To exploit structural information from point clouds, we use GCL to build P_1, P_2, and context branches for graph-based feature extraction. The P_1 and P_2 are used to extract features for finding point-voxel correspondences. The context branch is used to extract contextual information. In Fig. 1, two SetConv layers [21] gradually transform the point cloud into a latent feature space, and two GCLs are used to extract graph-based features. The channel dimensions are set to 32, 64, 128, and 256, respectively. The weights of the P_1 and P_2 branches are shared mutually. However, the weights of the context branch are not shared with the P_1 and P_2 branches. Moreover, we conduct a residual connection in graph-based feature generator to enhance the network's feature representation and deep supervision. Given P_1 and P_2, the graph-based feature generator in P_1, P_2, and context branches can output graph-based features F_1^g, F_2^g, and F_c^g, respectively.

3.1.3 Point-voxel correlation module

Finding correspondences between two adjacent point clouds is an important operation to estimate the scene flow. In this paper, we adopt the point-voxel correlation module proposed in [47] to build all-pairs correlation fields. Given F_1^g and F_2^g, we can obtain the correlation feature J by matrix dot product operation. Y_z denotes the translated point cloud, and z denotes the z^{th} iteration of flow iterator. Thus, the point-voxel correlation module can construct point correlation feature $J_p^{Y_z,P_2}$ via point branch and voxel correlation feature $J_v^{Y_z,P_2}$ via voxel branch. Furthermore, the point correlation feature and voxel correlation feature are fused by adding. Thus, we can obtain the point-voxel correlation feature J_z.

3.1.4 Single gate-based flow iterator

The recent work [47] adopts GRU for iterative flow estimation. Unlike [47], we use a single gate-based flow iterator to focus on short-term dependencies. The input of flow iterator includes correlation features J_z, current flow fields S_z (direction vectors between Y_z and P_1), hidden state from the previous iteration h_{z-1} and context feature F_c^g. Given hidden state from the previous iteration h_{z-1} and current iteration feature x_z, the current hidden state can be defined as

$$h_z = tanh(Conv([h_{z-1}, x_z], W_h)) \tag{6}$$

where $Conv()$ denotes 1-d convolution, W_h denotes learnable weight parameters, $tanh()$ denotes tanh activation function, [., .] is a concatenation, and x_z is a concatenation of J_z, S_z, and F_c^g. The current hidden state h_z is then fed into a fully connection layer to obtain coarse scene flow S'.

3.1.5 Graph-based flow refinement module

To make the coarse scene flow S' smoother in the 3D space, we introduce a graph-based flow refinement module to refine scene flow fields. As shown in Fig. 1, the graph-based refinement module is composed of three GCLs and a fully connected layer. The channel numbers of GCLs and fully connected layer are set to 256, 128, 64, and 3. The input and output of the graph-based refinement module include the coarse flow S' and the refined flow S, respectively.

3.2 The supervised learning strategy

To optimize the network, we follow the supervised learning strategy as in the [47] framework. The total loss is divided into iteration flow supervised loss and refinement flow supervised loss. The iteration flow supervised loss function $\mathcal{L}_1$ measures the difference between the coarse flow S' and ground truth S_{gt} at the end of the single gate-based flow iterator. $\mathcal{L}_1$ can be defined as

$$\mathcal{L}_1 = \sum_{z=1}^{Z} \eta(Z - z - 1)\|(S'_z - S_{gt})\|_1, \tag{7}$$

where S'_z denotes the estimated flow at z^{th} iteration of the single gate-based flow iterator, Z denotes the total number of flow iteration, $\|\cdot\|_1$ denotes the L1 norm, and η denotes a hyper-parameter that controls the weight of each iteration. In our experiments, η was set to 0.8.

The refinement flow supervised loss function $\mathcal{L}_2$ measures the difference between the refined flow S and ground truth S_{gt} at the end of the entire network. $\mathcal{L}_2$ can be defined as

$$\mathcal{L}_2 = \|S - S_{gt}\|_1, \tag{8}$$

where $\|\cdot\|_1$ denotes the L1 norm.

The final loss function can be defined as

$$\mathcal{L} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2, \tag{9}$$

where λ_1 and λ_2 denote respective loss weights. In our experiments, λ_1 and λ_2 were set to 0.8 and 1, respectively.

4 Experiments

In this section, we first introduce the implementation details, datasets, and metrics. Then, we report the experimental results on FlyingThings3D [24], KITTI [25, 26], and Argoverse [2] datasets. Moreover, we conducted a series of ablation studies to evaluate the effectiveness of our proposed components. Finally, we report the comparison of running time.

4.1 Implementation details

Experiments were conducted on a single NVIDIA RTX 3090 GPU with PyTorch [28]. We used Adam optimizer to train our network. The initial learning rate was set to 0.001, and the batch size was set to 2. The entire training process can be divided into two stages. In the first stage, we trained the network (without the graph-based flow refinement module) on the FlyingThings3D dataset [24] with 30 epochs. In the second stage, we froze the pre-trained weights and trained the graph-based flow refinement module on the FlyingThings3D dataset with 20 epochs. Following [47], we randomly sampled the input point cloud to 8192 points. The updating numbers of the flow iterator were set to 8 and 32 in the training and testing process.

4.2 Datasets

Table 1 summarizes various datasets for scene flow training and testing. Due to the difficulty of acquiring the ground truth of scene flow, we used the FlyingThings3D dataset [24] for training and testing. It contains a large number of synthetic scenarios. To verify that the model can work effectively in real-world scenes, we used KITTI [25, 26] and Argoverse [2] datasets for evaluating our model. In contrast to FlyingThings3D, both KITTI and Argoverse are real-world and self-driving datasets. In particular, Argoverse is a large-scale dataset for self-driving with challenging scenes.

Table 1 An overview of datasets for scene flow training and testing

Dataset	Training samples	Test samples samples	Real-world or synthetic	Scenario
FlyingThings3D (converted by [6])	4910	3824	Synthetic	Flying things
KITTI (converted by [6])	142	142	Real-world	self-driving
Argoverse (converted by [30])	2691	212	Real-world	self-driving

FlyingThings3D [24] FlyingThings3D is a synthetic dataset for optical flow, disparity, and scene flow training and evaluation. It contains 19640 image pairs of samples for training and 3824 image pairs for testing totally. Following [6, 31, 40, 47, 50], we used one-quarter of the training set (4910 samples) for training and used total samples in the test set (3824 samples) for evaluation. Note that we kept aside 2000 examples from the original training set as a validation set.

KITTI [25, 26] KITTI is a real-world dataset collected from a driving platform. It can be used for several 2D and 3D visual tasks, such as optical flow estimation, scene flow estimation, object detection, etc. Following [6, 31, 40, 47, 50], we used the KITTT scene flow 2015 [26] dataset (142 samples in the training set) to test our model. Note that we removed the ground by height (< 0.3m).

Argoverse [2] Argoverse is a real-world autonomous driving dataset for 3D mapping, stereo estimation, 3D tracking, and motion forecasting. Since the ground truth of scene flow was not provided in the original Argoverse dataset, Pontes et al. [30] created a new dataset for scene flow estimation, named Argoverse scene flow. It contains 2691 samples for training and 212 samples for testing. Following [18], we only used 212 test samples of the Argoverse scene flow dataset to evaluate our method.

4.3 Metrics

To make fair comparisons with [6, 18, 21, 31, 38, 40, 46, 47, 50], we adopted EPE (m), Outliers (%), Acc Strict (%), Acc Relax (%), and θ_ε (rad) as metrics.

- EPE (m): Mean of Euclidean distances between the predicted and ground truth pair of 3D vectors over all points. EPE can be defined as

$$EPE = \frac{1}{N} \sum_{y, y^* \in \mathbb{N}} \|y - y^*\|_2. \tag{10}$$

- Outliers (%): Percentage of points whose EPE >0.3m or relative error $>10\%$. Assuming that the set of points with EPE >0.3m is H_1 and the set of points satisfying the following condition is H_2, the condition can be defined as

$$\frac{\|y - y^*\|_2}{\|y^*\|_2} > 10\%. \tag{11}$$

Thus, the error point set H can be defined as $H_1 \cup H_2$. Assuming that D represents the number of elements in set H, Outliers can be defined as

$$Outliers = \frac{D}{N} \times 100\%. \tag{12}$$

- Acc Strict (%): Percentage of points whose EPE <0.05m or relative error <5%. Assuming that the set of points with EPE <0.05m is A'_1 and the set of points satisfying the following condition is A'_2, the condition can be defined as

$$\frac{\|y - y^*\|_2}{\|y^*\|_2} < 5\%. \tag{13}$$

Thus, the accuracy point set A' can be defined as $A'_1 \cup A'_2$. Assuming that R' represents the number of elements in set A', Acc Strict can be defined as

$$Acc\ Strict = \frac{R'}{N} \times 100\%. \tag{14}$$

- Acc Relax (%): Percentage of points whose EPE <0.1m or relative error <10%. Assuming that the set of points with EPE <0.1m is A''_1 and the set of points satisfying the following condition is A''_2, the condition can be defined as

$$\frac{\|y - y^*\|_2}{\|y^*\|_2} < 10\%. \tag{15}$$

Thus, the accuracy point set A'' can be defined as $A''_1 \cup A''_2$. Assuming that R'' represents the number of elements in set A'', Acc Relax can be defined as

$$Acc\ Relax = \frac{R''}{N} \times 100\%. \tag{16}$$

- θ_ε (rad): Mean angle error between the estimated flow and ground truth. θ_ε can be defined as

$$\theta_\varepsilon = \frac{1}{N}\sum_{y, y^* \in \mathbb{N}} arccos\left[\frac{1.0 + y_{\tau_1} y^*_{\tau_1} + y_{\tau_2} y^*_{\tau_2} + y_{\tau_3} y^*_{\tau_3}}{\sqrt{1.0 + y^2_{\tau_1} + y^2_{\tau_2} + y^2_{\tau_3}}\sqrt{1.0 + (y^*_{\tau_1})^2 + (y^*_{\tau_2})^2 + (y^*_{\tau_3})^2}}\right]. \tag{17}$$

In (10), (11), (13), (15), and (17), y denotes the estimated scene flow, and y^* denotes the ground truth of scene flow. In (10), (11), (13), and (15) $\| \cdot \|_2$ denotes the L2 norm. In (10), (12), (14), (16), and (17), N denotes the total point number of the point cloud. In (10) and (17), $\mathbb{N}$ denotes the set of points. In (17), y_{τ_1}, y_{τ_2}, and y_{τ_3} denote the motion component in the XYZ directions of y, respectively. Moreover, $y^*_{\tau_1}$, $y^*_{\tau_2}$, and $y^*_{\tau_3}$ denote the motion component in the XYZ directions of y^*, respectively.

4.4 Results

Scene flow comparisons conducted on both FlyingThings3D and KITTI datasets were summarized in Tables 2 and 3. Note that the best results on each dataset and metric are bold in Tables 2 and 3. We compared our method with the recent deep learning-based methods, including SPLATFlowNet [38], FlowNet3D [21], HPLFlowNet [6], EgoFlow [40], PointPWC-Net [50], FLOT [31], HASF [46], and PV-RAFT [47]. As shown in Table 2, the EPE and Outliers values produced by our method outperform [6, 21, 31, 38, 40, 46, 47, 50] on both FlyingThings3D and KITTI datasets, demonstrating the effectiveness of our method. In particular, on the KITTI dataset, the best EPE value of PV-RAFT [47] is 0.560, but the EPE value of our method is 0.484. As shown in Table 3, we can find that our model's Acc Strict and Acc Relax values outperform [6, 21, 31, 38, 40, 46, 50] by a large margin.

Table 2 Performance comparison on FlyingThings3D and KITTI datasets (Metrics: EPE and Outliers)

Dataset	Method	EPE (m) ↓	Outliers (%) ↓
FlyingThings3D	SPLATFlowNet [38]	0.1205	61.87
	FlowNet3D [21]	0.1136	60.16
	HPLFlowNet [6]	0.0804	42.87
	EgoFlow [40]	0.0688	40.36
	PointPWC-Net [50]	0.0588	34.24
	FLOT [31]	0.052	35.7
	HASF [46]	0.0492	30.83
	PV-RAFT [47]	0.0461	29.24
	Ours	**0.0442**	**26.37**
KITTI	SPLATFlowNet [38]	0.1988	65.75
	FlowNet3D [21]	0.1767	52.71
	HPLFlowNet [6]	0.1169	41.03
	EgoFlow [40]	0.1034	39.39
	PointPWC-Net [50]	0.0694	26.48
	HASF [46]	0.0622	24.92
	FLOT [31]	0.056	24.2
	PV-RAFT [47]	0.0560	21.63
	Ours	**0.0484**	**18.67**

Table 3 Performance comparison on FlyingThings3D and KITTI datasets (Metrics: Acc Strict and Acc Relax)

Dataset	Method	Acc Strict (%) ↑	Acc Relax (%) ↑
FlyingThings3D	SPLATFlowNet [38]	41.97	71.80
	FlowNet3D [21]	41.25	77.06
	HPLFlowNet [6]	61.44	85.55
	EgoFlow [40]	67.03	87.92
	PointPWC-Net [50]	73.79	92.76
	FLOT [31]	73.2	92.7
	HASF [46]	78.50	94.68
	PV-RAFT [47]	81.69	95.74
	Ours	**83.80**	**95.98**
KITTI	SPLATFlowNet [38]	21.74	53.91
	FlowNet3D [21]	37.88	66.77
	HPLFlowNet [6]	47.83	77.76
	EgoFlow [40]	48.84	82.24
	PointPWC-Net [50]	72.81	88.84
	FLOT [31]	75.5	90.8
	HASF [46]	76.49	90.26
	PV-RAFT [47]	82.26	93.72
	Ours	**86.59**	**95.15**

Table 4 Performance comparison on Argoverse dataset

Method	EPE (m) ↓	Acc Strict (%) ↑	Acc Relax (%) ↑	θ_ε (rad) ↓
(Non-learning) [18]	0.159±0.01	38.43±0.48	63.08±0.59	0.374±0.01
FlowNet3D [21]	0.455	1.34	6.12	0.736
PointPWC-Net [50]	0.405	8.25	25.47	0.674
PV-RAFT [47]	0.367	15.21	30.74	0.522
Ours	**0.351**	**19.58**	**33.87**	**0.483**

The main reason is that our method is based on the recent work PV-RAFT [47]. Thus, our network is also based on the powerful RAFT architecture [39] and contains the point-voxel correlation module. Moreover, compared to PV-RAFT [47], our model achieves competitive results on FlyingThings3D and KITTI datasets. In particular, on the KITTI dataset, the best Acc Strict value of [47] is 82.26%, but the Acc Strict value of our method is 86.59%. The first reason is that we introduce graph convolution to exploit the structural features. The second reason is our beneficial single gate-based flow iterator. Table 4 reports the performance comparison on Argoverse dataset. We compared our proposed approach with the non-learning method NSFP [18] and deep learning-based methods [21, 47, 50]. Note that the best results among deep learning-based methods on each metric are bold in Table 4. Following [18], we used the experimental protocol as in [21] and trained PV-RAFT [47] and our network on FlyingThings3D (converted by [21]). We can find that our method can achieve better performance than other deep learning-based methods [21, 47, 50]. However, the non-learning method NSFP [18] performs better than our method and other deep learning-based methods [21, 47, 50]. The main reason is that our method and other deep learning-based methods [21, 47, 50] are trained on the synthetic dataset. Thus, these deep learning methods encounter a large domain gap between training and test sets. However, since NSFP [18] optimizes parameters at runtime, the running time of NSFP [18] is larger than deep learning-based methods. We show the comparison of running time in Section 4.6. Furthermore, we give some visual examples compared to PV-RAFT [47]. Figure 2 shows some promising visualized examples on the KITTI dataset. As shown in Fig. 2, we can see that our model can produce more accurate flow fields of moving vehicles than PV-RAFT [47] in dynamic scenes.

4.5 Ablation study

To evaluate the effectiveness of our proposed components, we explore the variants of our architecture with experiments conducted on FlyingThings3D and KITTI datasets. Tables 5 and 6 reported the results of the ablation study on both FlyingThings3D and KITTI datasets, respectively. In Tables 5 and 6, "GCL", "RC", and "SGFI" denote the graph convolutional layer, residual connection, and single gate-based flow iterator. In Table 5, comparing the first and second rows, we find that using the GCL can improve the performance of scene flow estimation on the FlyingThings3D dataset. Comparing the second and third rows, we find that the improvement is achieved by adopting the RC component. Comparing the third and fourth rows, our model (with SGFI) generates lower EPE and Outlier values than without using SGFI. As shown in Table 6, we can find that using GCL, RC, and SGFI can improve the performance of scene flow estimation on the KITTI dataset. In particular, the value of

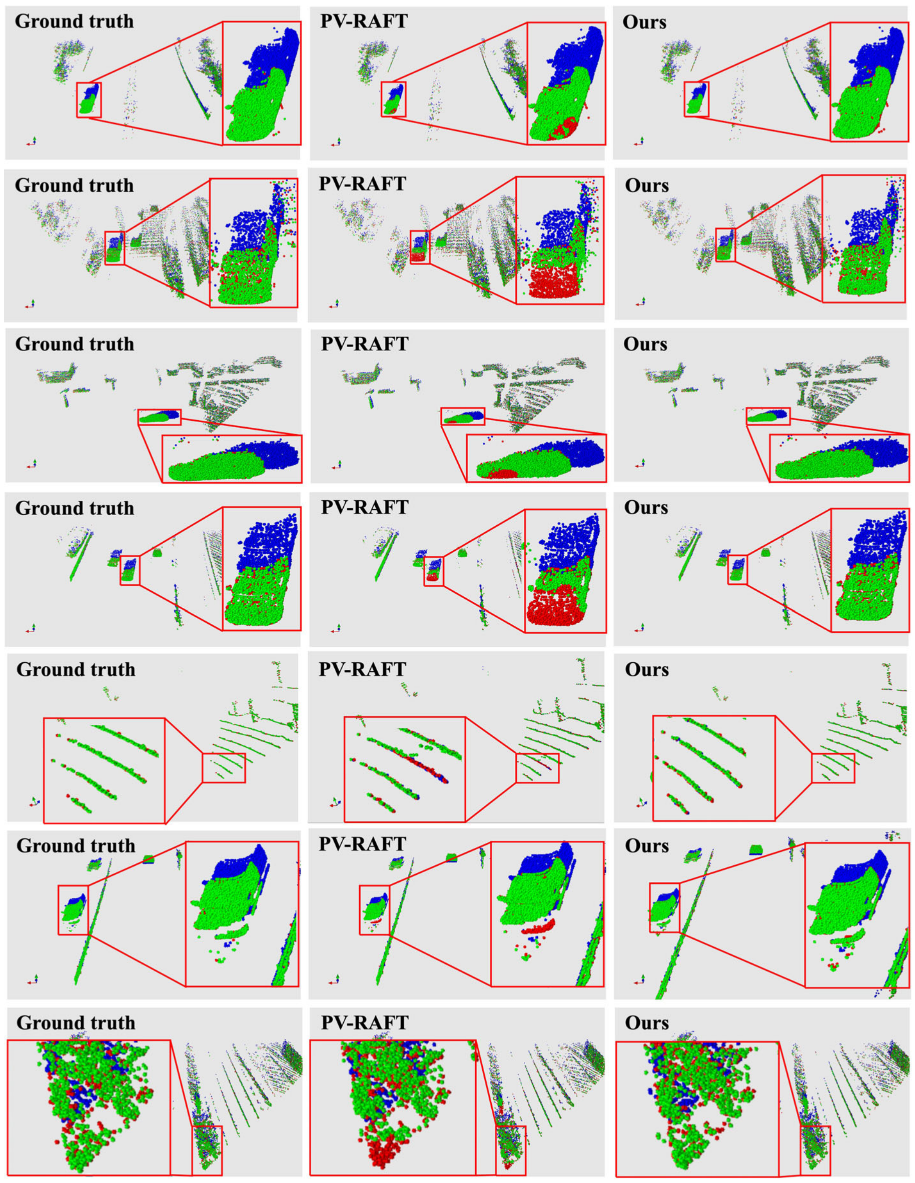


Fig. 2 Scene flow visual comparison between PV-RAFT [47] and our results on the KITTI dataset. Blue points and red points indicate the first point cloud and the second point cloud, respectively. Green points indicate the translated point cloud. In the first column "Ground truth", the first point cloud is translated by the ground truth scene flow

Table 5 Ablation study on FlyingThings3D dataset

GCL	RC	SGFI	FlyingThings3D			
			EPE (m) ↓	Outliers (%) ↓	Acc Strict (%) ↑	Acc Relax (%) ↑
			0.0461	29.24	81.69	95.74
✓			0.0450	27.92	82.13	95.91
✓	✓		0.0447	26.98	83.27	95.96
✓	✓	✓	0.0442	26.37	83.80	95.98

Acc Strict can be improved from 82.26% to 84.78% by using GCL. Based on the above observations, we can conclude that all proposed components benefit scene flow estimation.

4.6 Running time

We tested the running time on a single NVIDIA RTX 3090 GPU. Table 7 reported the running time comparison between the non-learning method NSFP [18] and the deep learning-based method PV-RAFT [47] and our method. Note that the running time represents the average time on all point clouds of the KITTI dataset. The point number was set to 8192. In Table 7, “Stage 1” and “Stage 2” denote the model trained in the first and second stages, respectively. As reported in Table 7, our method is faster than the non-learning method NSFP [18]. Compared to PV-RAFT [47], our method slightly increases the running time in the first and second stages.

5 Conclusion

In this paper, a graph-based feature extractor is proposed to encode point information from point neighborhoods better. Furthermore, multiple residual connections are used to effectively enhance the network’s graph-based feature representation and deep supervision. In addition, a single gate-based flow iterator is designed for updating flow fields iteratively in 3D space, which can boost the performance of scene flow estimation. Experimental results on widely used FlyingThings3D, KITTI, and Argoverse datasets demonstrate the effectiveness of our proposed approach.

Table 6 Ablation study on KITTI dataset

GCL	RC	SGFI	KITTI			
			EPE (m) ↓	Outliers (%) ↓	Acc Strict (%) ↑	Acc Relax (%) ↑
			0.0560	21.63	82.26	93.72
✓			0.0527	19.98	84.78	94.55
✓	✓		0.0501	19.22	85.63	94.97
✓	✓	✓	0.0484	18.67	86.59	95.15

Table 7 Comparison of running time

Method	Running time (in seconds)
NSFP (Non-learning) [18]	5.024
PV-RAFT (Stage 1) [47]	0.679
PV-RAFT (Stage 2) [47]	0.684
Ours (Stage 1)	0.709
Ours (Stage 2)	0.728

Acknowledgements This work is sponsored in part by the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20210594 and BK20210588), in part by the Natural Science Foundation for Colleges and Universities in Jiangsu Province (Grant No. 21KJB520016 and 21KJB520015), in part by the Natural Science Foundation of Nanjing University of Posts and Telecommunications (Grant No. NY221019 and NY221074), and in part by the National Natural Science Foundation of China (Grant No. 61931012, 61802204, and 62101280).

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Mingliang Zhai, Hao Gao, Ye Liu, Jianhui Nie and Kang Ni. The first draft of the manuscript was written by Mingliang Zhai and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability The datasets generated during and/or analysed during the current study are not publicly available due to an ongoing study but are available from the corresponding author on reasonable request.

Declarations

Disclosure of potential conflicts of interest The authors declare that there are no competing interests regarding the publication of this paper.

Research involving Human Participants and/or Animals The authors claim that this paper does not involve the research of human participants and/or animals.

Informed consent All authors contributed significantly to the research reported and have read and approved the submitted manuscript.

References

1. Behl A, Jafari OH, Mustikovela SK, Alhaija HA, Rother C, Geiger A (2017) Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In: IEEE International conference on computer vision (ICCV), pp 2593–2602
2. Chang M-F, Lambert J, Sangkloy P, Singh J, Bak S, Hartnett A, Wang D, Carr P, Lucey S, Ramanan D, Hays J (2019) Argoverse: 3d tracking and forecasting with rich maps. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 8740–8749
3. Chen J, Lei B, Song Q, Ying H, Chen DZ, Wu J (2020) A hierarchical graph network for 3d object detection on point clouds. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 389–398
4. Dewan A, Caselitz T, Tipaldi GD, Burgard W (2016) Rigid scene flow for 3d lidar scans. In: IEEE International conference on intelligent robots and systems (IROS), pp 1765–1770
5. Dosovitskiy A, Fischer P, Ilg E, Häusser P, Hazirbas C, Golkov V, Smagt Pvd, Cremers D, Brox T (2015) Flownet: Learning optical flow with convolutional networks. In: IEEE International conference on computer vision (ICCV), pp 2758–2766
6. Gu X, Wang Y, Wu C, Lee YJ, Wang P (2019) Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 3249–3258

7. Hadfield S, Bowden R (2011) Kinecting the dots: Particle based scene flow from depth sensors. In: International conference on computer vision, pp 2290–2295
8. Hornacek M, Fitzgibbon A, Rother C (2014) Sphereflow: 6 dof scene flow from rgb-d pairs. In: IEEE Conference on computer vision and pattern recognition, pp 3526–3533
9. Huguet F, Devernay F (2007) A variational method for scene flow estimation from stereo sequences. In: IEEE International conference on computer vision, pp 1–7
10. Hur J, Roth S (2020) Self-supervised monocular scene flow estimation. In: IEEE Conference on computer vision and pattern recognition (CVPR),pp 7394–7403
11. Hur J, Roth S (2021) Self-supervised multi-frame monocular scene flow. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 2683–2693
12. Ilg E, Saikia T, Keuper M, Brox T (2018) Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In: European conference on computer vision (ECCV), pp 626–643
13. Jampani V, Kiefel M, Gehler PV (2016) Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 4452–4461
14. Jiang H, Sun D, Jampani V, Lv Z, Learned-Miller E, Kautz J (2019) Sense: a shared encoder network for scene-flow estimation. In: IEEE International conference on computer vision (ICCV), pp 3194–3203
15. Lai H-Y, Tsai Y-H, Chiu W-C (2019) Bridging stereo matching and optical flow via spatiotemporal correspondence. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 1890–1899
16. Li Y, Baciu G (2021) Hsgan: Hierarchical graph learning for point cloud generation. IEEE Trans Image Process 30:4540–4554
17. Li G, Müller M, Thabet A, Ghanem B (2019) Deepgcns: Can gcns go as deep as cnns? In: IEEE International conference on computer vision (ICCV), pp 9266–9275
18. Li X, Pontes JK, Lucey S (2021) Neural scene flow prior. In: Advances in neural information processing systems (neurIPS)
19. Lin Z-H, Huang S-Y, Wang Y-CF (2020) Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 1797–1806
20. Liu P, King I, Lyu MR, Xu J (2020) Flow2stereo: Effective self-supervised learning of optical flow and stereo matching. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 6647–6656
21. Liu X, Qi CR, Guibas LJ (2019) Flownet3d: Learning scene flow in 3d point clouds. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 529–537
22. Luo C, Yang Z, Wang P, Wang Y, Xu W, Nevatia R, Yuille A (2020) Every pixel counts ++: Joint learning of geometry and motion with 3d holistic understanding. IEEE Trans Pattern Anal Mach Intell 42(10):2624–2641
23. Ma W-C, Wang S, Hu R, Xiong Y, Urtasun R (2019) Deep rigid instance scene flow. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 3609–3617
24. Mayer N, Ilg E, Häusser P, Fischer P, Cremers D, Dosovitskiy A, Brox T (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 4040–4048
25. Menze M, Geiger A (2015) Object scene flow for autonomous vehicles. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 3061–3070
26. Menze M, Heipke C, Geiger A (2015) Joint 3d estimation of vehicles and scene flow. ISPRS Annals of the Photogrammetry Remote Sensing and Spatial Information Sciences, pp 427–434
27. Pan L, Dai Y, Liu M, Porikli F, Pan Q (2020) Joint stereo video deblurring, scene flow estimation and moving object segmentation. IEEE Trans Image Process 29:1748–1761
28. Paszke A, Gross S, Massa F et al (2019) Pytorch: An imperative style , high-performance deep learning library. In: Advances in neural information processing systems (neurIPS)
29. Pillai S, Leonard JJ (2017) Towards visual ego-motion learning in robots. In: IEEE International conference on intelligent robots and systems (IROS), pp 5533–5540
30. Pontes JK, Hays J, Lucey S (2020) Scene flow from point clouds with or without learning. In: International conference on 3d vision (3DV), pp 261–270
31. Puy G, Boulch A, Marlet R (2020) Flot: Scene flow on point clouds guided by optimal transport. In: European conference on computer vision (ECCV), pp 527–544
32. Qi CR, Yi L, Su H, Guibas LJ (2017) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (neurIPS), pp 5099–5108

33. Qi CR, Zhou Y, Najibi M, Sun P, Vo K, Deng B, Anguelov D (2021) Offboard 3d object detection from point cloud sequences. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 6134–6144
34. Quiroga J, Brox T, Devernay F, Crowley J (2014) Dense semi-rigid scene flow estimation from rgbd images. In: European conference on computer vision (ECCV), pp 567–582
35. Schuster R, Wasenmuller O, Unger C, Kuschk G, Stricker D (2020) Sceneflowfields++: Multi-frame matching, visibility prediction, and robust interpolation for scene flow estimation. In: International journal of computer vision, vol 128, pp 527–546
36. Shen W, Wei Z, Huang S, Zhang B, Chen P, Zhao P, Zhang Q (2021) Verifiability and predictability: Interpreting utilities of network architectures for point cloud processing. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 10703–10712
37. Shi W, Rajkumar R (2020) Point-gnn: Graph neural network for 3d object detection in a point cloud. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 1708–1716
38. Su H, Jampani V, Sun D, Maji S, Kalogerakis E, Yang M-H, Kautz J (2018) Splatnet: Sparse lattice networks for point cloud processing. In: IEEE Conference on computer vision and pattern recognition, pp 2530–2539
39. Teed Z, Deng J (2020) Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision (ECCV), pp 402–419
40. Tishchenko I, Lombardi S, Oswald MR, Pollefeys M (2020) Self-supervised learning of non-rigid residual flow and ego-motion. In: International conference on 3d vision (3DV), pp 150–159
41. Ushani AK, Wolcott RW, Walls JM, Eustice RM (2017) A learning approach for real-time temporal scene flow estimation from lidar data. In: IEEE International conference on robotics and automation (ICRA), pp 5666–5673
42. Vogel C, Schindler K, Roth S (2013) Piecewise rigid scene flow. In: IEEE International conference on computer vision, pp 1377–1384
43. Wang P, Li W, Gao Z, Zhang Y, Tang C, Ogunbona P (2017) Scene flow to action map: a new representation for rgb-d based action recognition with convolutional neural networks. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 416–425
44. Wang Z, Li S, Howard-Jenkins H, Prisacariu VA, Chen M (2020) Flownet3d++: Geometric losses for deep scene flow estimation. In: IEEE Winter conference on applications of computer vision (WACV), pp 91–98
45. Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM (2019) Dynamic graph cnn for learning on point clouds. ACM Trans Graphics 38(5):1–12
46. Wang G, Wu X, Liu Z, Wang H (2021) Hierarchical attention learning of scene flow in 3d point clouds. IEEE Trans Image Process 30:5168–5181
47. Wei Y, Wang Z, Rao Y, Lu J, Zhou J (2021) Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 6954–6963
48. Wu F, Jing X-Y, Wei P, Lan C, Ji Y, Jiang G-P, Huang Q (2022) Semi-supervised multi-view graph convolutional networks with application to webpage classification. Inf Sci 591:142–154
49. Wu W, Qi Z, Fuxin L (2019) Pointconv: Deep convolutional networks on 3d point clouds. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 9613–9622
50. Wu W, Wang ZY, Li Z, Liu W, Fuxin L (2020) Pointpwc-net: Cost volume on point clouds for (self-)supervised scene flow estimation. In: European conference on computer vision (ECCV), pp 88–107
51. Yang G, Ramanan D (2020) Upgrading optical flow to 3d scene flow through optical expansion. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 1331–1340
52. Yin Z, Shi J (2018) Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: IEEE Conference on computer vision and pattern recognition, pp 1983–1992
53. Zhou T, Brown M, Snavely N, Lowe DG (2017) Unsupervised learning of depth and ego-motion from video. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 6612–6619
54. Zou Y, Luo Z, Huang J-B (2018) Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In: European conference on computer vision (ECCV), pp 38–55

Affiliations

Mingliang Zhai[1] · Hao Gao[1] · Ye Liu[1] · Jianhui Nie[1] · Kang Ni[2]

Hao Gao
tsgaohao@gmail.com

Ye Liu
yeliu@njupt.edu.cn

Jianhui Nie
njh19@njupt.edu.cn

Kang Ni
tznikang@163.com

[1] School of Automation, Nanjing University of Posts and Telecommunications, Nanjing, 210023, Jiangsu Province, China

[2] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, 210023, Jiangsu Province, China