Bi-PointFlowNet: Bidirectional Learning for Point Cloud Based Scene Flow Estimation

Wencan Cheng[1][0000−0002−7996−0236] and Jong Hwan Ko[2][0000−0003−4434−4318]

[1] Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16419, South Korea
[2] College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea
{cwc1260,jhko}@skku.edu

Abstract. Scene flow estimation, which extracts point-wise motion between scenes, is becoming a crucial task in many computer vision tasks. However, all of the existing estimation methods utilize only the unidirectional features, restricting the accuracy and generality. This paper presents a novel scene flow estimation architecture using bidirectional flow embedding layers. The proposed bidirectional layer learns features along both forward and backward directions, enhancing the estimation performance. In addition, hierarchical feature extraction and warping improve the performance and reduce computational overhead. Experimental results show that the proposed architecture achieved a new state-of-the-art record by outperforming other approaches with large margin in both FlyingThings3D and KITTI benchmarks. Codes are available at https://github.com/cwc1260/BiFlow.

Keywords: Scene flow estimation, Point cloud, Bidirectional learning

1 Introduction

A scene flow estimation task is to capture the point-wise motion from two consecutive frames. As it provides the fundamental low-level information of dynamic scenes, it has become an essential step in various high-level computer vision tasks including object detection and motion segmentation. Therefore, accurate scene flow estimation is crucial for perceiving dynamic environment in real-world applications such as autonomous driving and robot navigation [42, 16].

Early scene flow estimation approaches employed RGB images as an input. However, due to the increasing applications of LiDAR sensors that can capture dynamic scenes in the form of three-dimensional (3D) point clouds, scene flow estimation using point cloud has been actively studied. FlowNet3D [21] proposed the first point cloud based estimation model using the hierarchical architecture of PointNet++ [32]. Based on this scheme, several studies [37, 9] proposed the multi-scale correlation propagation structure for more accurate estimation. Recently, PointPWC [46] significantly improved the estimation performance using

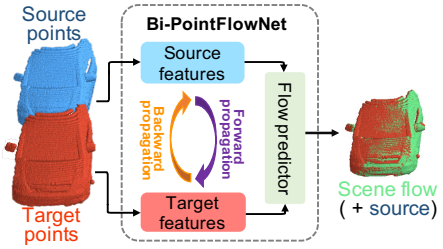

Fig. 1. Illustration of the bidirectional learning for scene flow estimation. The features extracted from each input frame are propagated bidirectionally for generating augmented feature representations that benefit scene flow estimation. The estimated scene flows are warped with the source frame for a clear comparison with the target frame.

regression of multi-scale flows in a coarse-to-fine manner. Another study [30] proposed integration of an optimal transport-solving module in a neural network for estimating scene flow.

All of these existing methods utilized only the unidirectional feature propagation (i.e., propagating source point features to target points) for calculating flow correlations. Meanwhile, various models for natural language processing (NLP) tasks [33, 8, 6, 18] showed that the bidirectionally-learned features can significantly improve the performance due to their strong contextual information. Since scene flow estimation is also a temporal sequence processing task, bidirectional learning can boost the estimation performance. Bidirectional configuration has already proved its effectiveness on optical flow estimation, which is similar representation as scene flow estimation [11, 43, 14, 12, 20]. However, to the best of our knowledge, there is no prior work that utilized bidirectional learning for the estimation of scene flow in the 3D space.

Based on this motivation, we propose Bi-PointFlowNet, a novel bidirectional architecture for point cloud based scene flow estimation. As shown in Fig. 1, the bidirectional correlations can be learned by forward-propagation from the source features and backward-propagation from the target features. Therefore, each frame contains knowledge from the other, allowing the features to produce stronger correlations. In addition, the proposed Bi-PointFlowNet adopts the coarse-to-fine method for multi-scale bidirectional correlation extraction.

We evaluated the proposed model on two challenging benchmarks, the FlyingThings3D [23] and KITTI [26] datasets, under both occluded and non-occluded conditions. On the FlyingThings3D dataset, Bi-PointFlowNet outperforms all existing methods with more than 44% and 32% of estimation error reduction on the non-occluded cases and the occluded cases, respectively. To evaluate generalization performance, we trained the models on the synthetic (FlyingThings3D) dataset and evaluated on the real-world LiDAR scan (KITTI Scene Flow 2015) dataset without fine-tuning. Compared to the existing approaches, the results show that Bi-PointFlowNet achieves improved generality with 44% and 21% lower error on the the non-occluded and occluded cases, respectively. Our Bi-

PointFlowNet also showed better time efficiency while maintaining high accuracy.

The key contributions of this paper are summarized as follows:

- We are the first to apply the bidirectional learning architecture used for a 3D scene flow estimation task based on point cloud. The model can extract bidirectional correlations that significantly improve flow estimation performance.
- We propose a decomposed form of the bidirectional layer that optimizes the computation count for accelerated bidirectional correlations extraction.
- The proposed model achieves the state-of-the-art performance and generality on the synthetic FlyingThings3D and real-world KITTI benchmarks in both occluded and non-occluded conditions.

2 Related Work

2.1 Scene Flow Estimation

The 3D scene flow, first introduced by [39], represents a dense 3D motion vector field for each point on every surface in the scene. Early dense scene flow estimation approaches [10, 38, 29, 44, 3, 25, 23, 40] used stereo RGB images as an input. With the rapid development of 3D sensors and the emergence of point cloud based networks [31, 45, 32], a line of studies proposed estimating the scene flow using the raw 3D point clouds. FlowNet3D [21] was the first study that estimated the scene flow from two raw point cloud frames through a deep neural network. However, the performance of FlowNet3D was restricted by its single flow correlation. To address this drawback, Gu et al. proposed HPLFlownet [9] that captures multi-scale correlations using a bilateral convolutional layer [13, 34]. PointPWC-Net [46] further improved the performance and efficiency by hierarchically regressing scene flow in a coarse-to-fine manner. There are several other approaches that leveraged the all-to-all correlation, including FLOT [30] that learns the all-to-all correlation by solving an optimal transport problem, and FlowStep3D [16] that iteratively aligns point clouds based on the iterative closest point (ICP) algorithm [5, 2]. However, learning an all-to-all correlation matrix is computationally inefficient when the input point clouds contain a large number of points.

Our Bi-PointFlowNet is inspired by these point cloud based methods. It also adopts the coarse-to-fine architecture to capture the multi-level correlation and to reduce computational overhead. However, the proposed method is different from the existing models as it utilizes bidirectional learning, which collects the contextual information from both source and target features for more accurate estimation.

2.2 Bidirectional Models

The bidirectional models aim to extract features based on both the current and future states. They are able to capture strong contextual information with future

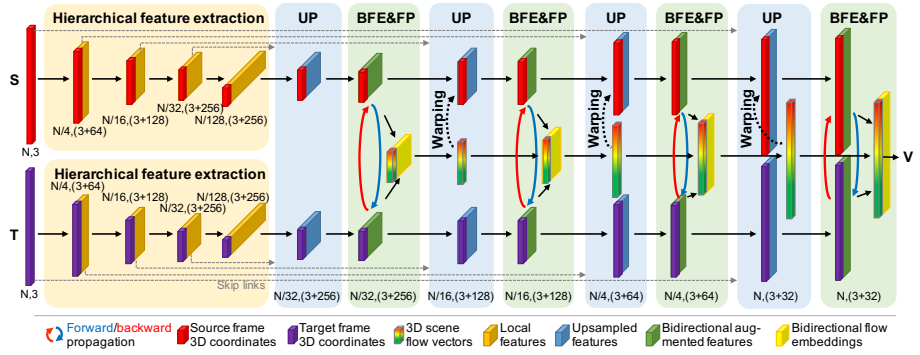

Fig. 2. Architecture of Bi-PointFlowNet for scene flow estimation. (**UP** stands for an upsampling layer. **BFE&FP** stand for a bidirectional flow embedding layer and a flow prediction layer. They are visualized in the same block for a clear representation.) First, we feed two consecutive input point frames into the shared hierarchical feature extraction module for multilevel feature extraction. Then the upsampling layers propagate features from high levels to low levels and the warping operations are directly applied upon the upsampled points. After each upsampling layer, a bidirectional flow embedding layer is adopted for bidirectional feature (forward feature and backward features) propagation and flow embedding generation. The flow embeddings are immediately fed into the flow prediction layer for scene flow regression according to the current level. The figure is best viewed in color.

knowledge, which is helpful for many time-sequence processing tasks such as natural language processing (NLP). The bidirectional model was first proposed in a bidirectional RNN (BRNN) [33] that learns sequence representations forward and backward through two separate networks. Subsequently, a more powerful structure called bidirectional long short-term memory (BiLSTM) [8] was proposed and successfully applied in frame-wise phoneme classification. Based on these fundamental studies, various approaches [1, 49, 28, 24] have been actively explored. In recent years, the bidirectional encoder representation transformer (BERT) [6] and its variants [22, 17] have achieved overwhelming performance in various applications including language understanding [48, 18, 47].

Recently, a series of studies showed that 2D optical flow estimation can also benefit from bidirectional learning, because optical flow estimation is a type of time series-based task as well. MirrorFlow [11] reused a symmetric optical flow algorithm bidirectionally to extract forward and backward optical flows, which are then constrained by the bidirectional motion and occlusion consistency. Similarly, Wang et al. [43] also proposed an approach that generates bidirectional optical flows but by reusing a neural network. In addition, Janai et al. [14] proposed a method that extracts bidirectional optical flows in a coarse-to-fine manner based on a pyramid structure. Based on the bidirectional models, Hur et al. [12] implemented an architecture that iteratively refines the optical flow estimation by using the previous output.

However, bidirectional learning has not been yet explored in 3D scene flow estimation. To the best of our knowledge, we propose the first bidirectional model for scene flow estimation based on 3D point cloud. Different from 2D optical flow estimation methods, our proposed model does not reuse an unidirectional flow estimator nor explicitly generate both forward and backward flows. Instead, we only implicitly encode bidirectional features like BRNN and fuse them for only forward flow estimation. Consequently, the model can eliminate redundant computations.

3 Problem Statement

Scene flow estimation using a point cloud is to estimate a 3D point-wise motion field in a dynamic scene. The inputs are two consecutive point cloud frames, the source frame $S = \{p_i = (x_i, f_i)\}_{i=1}^{N}$ and target frame $T = \{q_j = (y_j, g_j)\}_{j=1}^{M}$, where each point consists of a 3D coordinate $x_i, y_j \in \mathbb{R}^3$ and its corresponding feature $f_i, g_j \in \mathbb{R}^c$. The outputs are 3D motion field vectors $V = \{v_i \in \mathbb{R}^3\}_{i=1}^{N}$ that represent the point-wise non-rigid transformation from the source frame toward the target frame. Our goal is to estimate the best non-rigid transformation V that represents the best alignment from the source frame towards the target frame. Note that N and M denote the number of points in the source and target frame, respectively. However, N and M are not required to be equal because of sparsity and occlusion in a point cloud. Therefore, learning the hard correspondence between the two frames is not feasible. Instead, we directly learn the flow vector for each point in the source frame, as in the most of the recent methods [21, 46, 30, 9, 16, 19].

4 Bi-PointFlowNet

The proposed Bi-PointFlowNet estimates scene flow using a hierarchical architecture with bidirectional flow embedding extraction. The network accepts two consecutive point cloud frames S and T as an input. The output of the network is the estimated scene flow vectors V. As shown in Fig. 2, Bi-PointFlowNet consists of four components. First, a hierarchical feature extractor extracts multi-level local features in both input frames. Second, novel bidirectional flow embedding layers are applied at different upsampled levels for multi-level bidirectional correlation extraction. Third, upsampling and warping layers propagate features from higher levels to lower levels. Finally, the flow predictor aggregates bidirectional correlations and propagated features to obtain the flow estimation for each level.

4.1 Hierarchical Feature Extraction

To extract informative features from point clouds more efficiently and effectively, we adopt the hierarchical feature extraction scheme commonly used in point cloud processing [32, 45]. Feature extractions proceed in L levels for generating

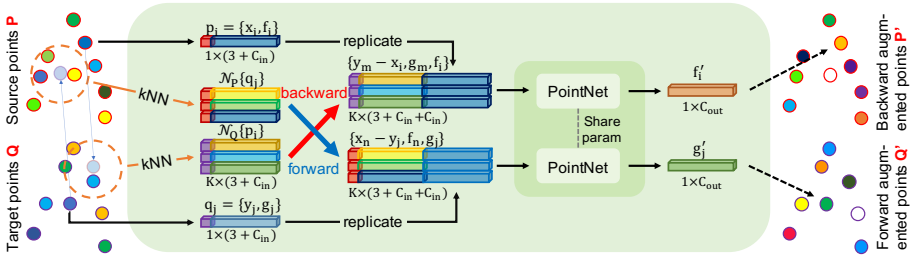

Fig. 3. Bidirectional feature propagation in the novel Bidirectional Flow Embedding layer. Each point first groups the nearest neighbors from the other point cloud forming a local region. (Forward grouping: a source point groups points from the target points. Backward grouping: a target point groups points from the source points.) Each point in the local region is then concatenated with its own local features propagated from previous feature upsampling. Finally, a PointNet layer with shared parameters accepts the local regions as input and updates the bidirectional augmented features for each point.

hierarchical features from dense to sparse. At each level l, dense input points and their corresponding features are first subsampled through the *furthest point sampling*, which forms a sparse point set. Then, the *k-nearest neighbor* is used to locally group dense points around every subsampled sparse point. Finally, a *Pointconv* [45] layer aggregates the features and coordinates from the grouped local points, and generates the local feature for each subsampled point.

4.2 Bidirectional Flow Embedding

Unlike conventional correlation extraction that uses only unidirectional features between two consecutive frames, we propose a novel bidirectional flow embedding (BFE) layer that provides rich contextual information. The BFE layer first generates bidirectional augmented feature representations through a bidirectional feature propagation (BFP) module, as shown in Fig. 3. Then, a conventional *flow embedding* (FE) layer is followed to extract correlation embeddings for flow regression.

Let the inputs to the BFP module be P and Q, where $P \subset S$ and $Q \subset T$ are the subsampled points. For each point $p_i \in P$ in the target frame, the BFP module first collects the nearest points from the source frame that forms the group $\mathcal{N}_Q\{p_i\}$. Likewise, the BFP module collects points from the target frame for each point $q_j \in Q$ in the source frame that forms group $\mathcal{N}_P\{q_j\}$. Subsequently, the points p_i, q_j and their groups $\mathcal{N}_Q\{p_i\}$, $\mathcal{N}_P\{q_j\}$ are simultaneously processed by a shared PointNet [31, 32] layer to generate the bidirectionally-augmented point representations. Thus, the bidirectional-augmented point features, which are backward augmented feature f_i' for p_i and forward augmented feature g_j' for q_j, are respectively represented as:

$$f_i^{'} = \underset{(y_m, g_m) \in \mathcal{N}_Q\{p_i\}}{MAX} (MLP([y_m - x_i, g_m, f_i])), \tag{1}$$

$$g_j^{'} = \underset{(x_n, f_n) \in \mathcal{N}_P\{q_j\}}{MAX} (MLP([x_n - y_j, f_n, g_j])), \tag{2}$$

where MLP and MAX denote the shared MLP and maxpooling layer of the learned PointNet, respectively, and '$[\cdot, \cdot]$' denotes the channel concatenation operator.

Since the output estimation are only forward directed, a normal unidirectional *flow embedding* (FE) correlation layer captures correlations from the source bidirectionally-augmented points to the target bidirectionally-augmented points after the BFP. We name the correlation as the bidirectional flow embedding, because they are extracted from the bidirectional features. Note that, the generated augmented points are also fed to the subsequent upsampling layer for hierarchical feature propagation, which will be elaborated in Sec. 4.4.

4.3 Decomposed Form of Bidirectional Flow Embedding

The aforementioned BFE layer directly follows the standard procedure (i.e. grouping $\rightarrow$ concatenation $\rightarrow$ MLP $\rightarrow$ max-pooling) to fuse the local information, as presented in [32]. However, this procedure requires a large number of operations as it should be executed for each point of the input point cloud. Let the inputs to the BFE module be $P = \{(x_i, f_i) \in \mathbb{R}^{3+C}\}_{i=1}^{N'}$ and $Q = \{(y_j, g_j) \in \mathbb{R}^{3+C}\}_{j=1}^{M'}$, and the number of grouping points to be K. For convenient analysis, we assume a one-layer MLP whose weights are $W \in \mathbb{R}^{(3+C+C) \times C'}$. Then, $(3 + C + C) \times C'$ MLP computations are required for $(N' + M') \times K$ times. Therefore, the total operation count of BFE is $(N' + M') \times K \times (3 + C + C) \times C'$. However, as the total number of input points is $(N' + M')$, every K neighbor points are grouped into $(N' + M')$ groups and then calculated by the MLP. Thus, at least $(N' + M') \times (K - 1)$ MLP operations are repeatedly calculated.

To optimize this redundancy, we propose a decomposed form of BFE. First, we decompose the MLP weights W into three sub-weights: the weights for local position encoding $W_p \in \mathbb{R}^{3 \times C'}$, the weights for the bidirectional propagated feature $W_b \in \mathbb{R}^{C \times C'}$ and the weights for the replicated feature $W_r \in \mathbb{R}^{C \times C'}$. W_b and W_r are performed at both P and Q before grouping, thus forming transformed features $W_b f_i$, $W_b g_j$, $W_r f_i$ and $W_r g_j$. These transformed features and their corresponding coordinates are then supplied for grouping. Afterwards, only W_p is used for transformation of the grouped local coordinates. Finally, we simply add the transformed local coordinates with the transformed features together and apply the activation function. Therefore, Equation 1 and 2 can be transformed as:

$$f_i^{'} = \underset{(y_m, W_b g_m) \in \mathcal{N}_Q\{p_i = (x_i, W_r f_i)\}}{MAX} \sigma(W_p(y_m - x_i) + W_b g_m + W_r f_i), \tag{3}$$

$$g_j^{'} = \underset{(x_n, W_b f_n) \in \mathcal{N}_P\{q_j=(y_j, W_r g_j)\}}{MAX} \sigma(W_p(x_n - y_j) + W_b f_n + W_r g_j), \tag{4}$$

where σ represents the activation function. Thus, computing W_b, W_r at P, Q only requires $(N'+M') \times (C+C) \times C'$ operations, while local coordinate transformation requires $(N'+M') \times (K \times 3) \times C'$ operations. As a result, the total computations count of decomposed BFE is reduced to $(N'+M') \times (K \times 3 + C + C) \times C'$.

4.4 Upsampling and Warping

The upsampling (UP) layer can propagate features (including flows, local features, and bidirectional augmented points) from sparse levels to dense levels. To reduce the computational cost, we adopt the 3D interpolation using the inverse-distance weighted function based on the k nearest neighbors. Let $\{(x_j^l, f_j^l)\}_{j=1}^{N^l}$ denotes the coordinates and features from a high level, and $\{x_i^{l-1}\}_{i=1}^{N^{l-1}}$ denotes the coordinates from a low level through a super link, where N^{l-1} and N^l are the number of points and $N^{l-1} > N^l$. The interpolated feature of a dense point $\{x_i^l\}$ is defined as:

$$f_i^{l-1} = \frac{\sum_{j=1}^{k} w(x_j^l, x_i^{l-1}) f_j^l}{\sum_{j=1}^{k} w(x_j^l, x_i^{l-1})}, \tag{5}$$

where $w(x_j^l, x_i^{l-1}) = 1/||x_j^l - x_i^{l-1}||_2$, and $k = 3$ by default.

The upsampled scene flows are immediately accumulated to the source frame in order to obtain a frame closer to the corresponding target frame through a warping layer. This process can be simply denoted as $x_w^l = x^l + v^l$ for each source point x^l from the l-th level, where v^l denotes an upsampled flow vector. Through warping, the warped points gradually become close to the target frame. Thus, the subsequent BFE layer can easily group more valuable points with high semantic similarity that can promote more accurate flow estimation. In addition, accurate flow estimation of the current level also enhances warping at the next level.

4.5 Scene Flow Prediction

We implement a scene flow predictor in order to regress the scene flow vector. For each level, the inputs are the upsampled flows and features from the upsampling layer, and bidirectional flow embeddings from the BFE layer. First, the predictor uses a *Pointconv* to produce smooth features by locally fusing these features and flows around each warped source point. Subsequently, a MLP transforms the smooth high-dimensional features into 3-dimensional scene flow vectors for all points. Since the predictor only focuses on a small region around each warped source point, the outputs from the last MLP layer are point-wise flow residuals, as in [41, 27]. Afterwards, the residuals are further accumulated with the upsampled flows forming the output flow estimation for the current level.

4.6 Loss Function

The training process adopts the multi-level supervision manner used in previous studies for optical flow estimation [7, 35] and scene flow estimation [42, 46]. At each level, the estimated flows are supervised by the L2 loss. Let $\{v_i^l\}_{i=1}^{N^l}$ denote the scene flow vectors estimated from the l-th level and $\{\hat{v}_i^l\}_{i=1}^{N^l}$ denote the ground truth scene flow vectors of the l-th level. The training loss is defined as:

$$\mathcal{L} = \sum_{l=0}^{L-1} \alpha^l \sum_{i=1}^{N^l} \|\hat{v}_i^l - v_i^l\|_2, \tag{6}$$

where α^l is the weight of the loss function at level l. The weights are set to be $\alpha^0 = 0.16$, $\alpha^1 = 0.08$, $\alpha^2 = 0.04$, $\alpha^3 = 0.02$ by default.

5 Experiments

5.1 Experimental Settings

We conducted experiments on an NVIDIA TITAN RTX GPU with PyTorch. As shown in Fig. 2, we implemented a hierarchical model with $L = 4$ levels. We used $N = M = 8,192$ points as inputs. The numbers of subsampled points of each level are defined as $N^1 = 2,048$, $N^2 = 512$, $N^3 = 256$, and $N^4 = 64$. As in the previous methods, we first trained and evaluated networks on the synthetic FlyingThings3D [23] dataset (Sec. 5.3). Then, to validate the generalization ability, the trained model was directly evaluated on the real-world KITTI Scene Flow 2015 [26] dataset without any fine-tuning (Sec. 5.4).

5.2 Evaluation Metrics

For a fair comparison, we adopted the same evaluation metrics as used in recent works [9, 46, 30, 16, 19].

- **EPE3D$_{full}$ (m)**: the main evaluation metric measuring end-point-error $\|\hat{v}_i^l - v_i^l\|_2$ averaged over **all** point.
- **EPE3D (m)**: the main evaluation metric measuring end-point-error $\|\hat{v}_i^l - v_i^l\|_2$ averaged each **non-occluded** point.
- **ACC3DS**: the percentage of points whose EPE3D < 0.05m or relative error $< 5\%$.
- **ACC3DR**: the percentage of points whose EPE3D < 0.1m or relative error $< 10\%$.
- **Outliers3D**: the percentage of points whose EPE3D > 0.3m 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 $< 5\%$.

Dataset	Method	EPE3D (m) ↓	ACC3D S ↑	ACC3D R ↑	Outliers 3D ↓	EPE2D (px) ↓	ACC 2D ↑
$FT3D_s$	FlowNet3D [21]	0.113	0.412	0.771	0.602	5.974	0.569
	HPLFlowNet [9]	0.080	0.614	0.855	0.429	4.672	0.676
	PointPWC [46]	0.059	0.738	0.928	0.342	3.239	0.799
	FLOT [30]	0.052	0.732	0.927	0.357	-	-
	HCRF-Flow [19]	0.048	0.835	0.950	0.261	2.565	0.870
	FlowStep3D [16]	0.045	0.816	0.961	0.216	-	-
	Ours	**0.028**	**0.918**	**0.978**	**0.143**	**1.582**	**0.929**
$KITTI_s$	FlowNet3D [21]	0.177	0.374	0.668	0.527	7.214	0.509
	HPLFlowNet [9]	0.117	0.478	0.778	0.410	4.805	0.593
	PointPWC [46]	0.069	0.728	0.888	0.265	1.902	0.866
	FLOT [30]	0.056	0.755	0.908	0.242	-	-
	HCRF-Flow [19]	0.053	0.863	0.944	0.179	2.070	0.865
	FlowStep3D [16]	0.054	0.805	0.925	0.149	-	-
	Ours	**0.030**	**0.920**	**0.960**	**0.141**	**1.056**	**0.949**

Table 1. Comparison of the proposed method with previous state-of-the-art methods on the non-occluded $FT3D_s$ and $KITTI_s$ datasets. All methods are trained only on the $FT3D_s$ dataset.

Dataset	Method	$EPE3D_{full}$ (m) ↓	EPE3D (m) ↓	ACC3D S ↑	ACC3D R ↑	Outliers 3D ↓
$FT3D_o$	FlowNet3D [21]	0.211	0.157	0.228	0.582	0.804
	HPLFlowNet [9]	0.201	0.168	0.262	0.574	0.812
	FLOT [30]	0.250	0.153	0.396	0.660	0.662
	PointPWC [46]	0.195	0.155	0.416	0.699	0.638
	OGSFNet [27]	0.163	0.121	0.551	0.776	0.518
	RAFT-3D (16 iters) [36]	-	**0.064**	**0.837**	0.892	-
	Ours	**0.102**	0.073	0.791	**0.896**	**0.274**
$KITTI_o$	FlowNet3D [21]	0.183	-	0.098	0.394	0.799
	HPLFlowNet [9]	0.343	-	0.103	0.386	0.814
	FLOT [30]	0.130	-	0.278	0.667	0.529
	PointPWC [46]	0.118	-	0.403	0.757	0.496
	OGSFNet [27]	0.075	-	0.706	0.869	0.327
	Ours	**0.065**	-	**0.769**	**0.906**	**0.264**

Table 2. Comparison of the proposed method with previous state-of-the-art methods on the occluded $FT3D_o$ and $KITTI_o$ datasets. All methods are trained only on the $FT3D_o$ dataset.

5.3 Training and Evaluation on FlyingThings3D

FlyingThing3D [23] is a synthetic dataset composed of 19,640 pairs of frames for training and 3,824 pairs of frames for testing. Each frame consists of stereo and RGB-D images rendered from scenes with multiple moving objects sampled from

ShapeNet [4] dataset. We trained and evaluated our proposed model based on two versions of datasets prepared by different pre-processing methodologies. The first version is FT3D_s, which removes the occluded points after transforming the image data into points, as suggested in [9, 46, 30, 16]. The second version, FT3D_o introduced by [21, 30, 27], remains the occluded points. The input points of $N = 8,192$ are randomly sampled from each frame with non-correspondence.

For training, we used the Adam optimizer [15] with beta1 = 0.9, beta2 = 0.999, and starting learning rate $\alpha = 0.0001$. The learning rate is reduced by half every 80 epochs. We trained the model for 560 epochs.

Results. We report the performance of the proposed model compared to other state-of-the-art approaches [21, 9, 46, 30, 16, 27]. On the non-occluded FlyingThings3D dataset, the proposed Bi-PointFlowNet achieved a new state-of-the-art record on all evaluation metrics based on point cloud, as shown in Table 1. It outperformed all recent state-of-the-art methods with more than 44% reduction of estimation error. When compared to the similar coarse-to-fine PointPWC [46], our model achieved an error reduction of 52%. On the other hand, Table 2 also shows remarkable performance of our work when handling occluded data. Our Bi-PointFlowNet improved the state-of-the-art performance by 32%. In addition, we also compared our method with RGB-D image-based RAFT-3D [36]. Table 2 shows that our method achieved comparable performance than Raft-3D with 16 iterations. Although our method did not achieve better EPE3D and ACC3DS, it outperformed Raft-3D for the ACC3DR metric. Despite a minor increase in errors, we report that our model requires significant less computation (13.3GFLOPs) and parameter size (7.9M) than RAFT-3D (329GFLOPs, 45M), making it more applicable to time-sensitive low-power applications. According to [36], we expect that RAFT-3D having similar computation to ours with less iteration will yield much worse accuracy than ours.

5.4 Generalization on KITTI

In order to evaluate the generalization ability of Bi-PointFlowNet to real-world data, we followed the same evaluation strategy as in the recent studies [21, 9, 46, 30, 16, 27]. We directly tested the trained model on the real-world KITTI [26] dataset without fine-tuning. The KITTI dataset contains 200 training and 200 testing sets. However, due to unprovided disparities in the testing set and in parts of the training set, we used 142 scenes (non-occluded) and 150 scenes (occluded) from the training set with available raw point clouds. For fair comparison of our method with the previous approaches [21, 9, 46, 30, 16, 27], we followed the common step that removes ground points by height < 0.3 m. According to the preparation of the FlyingThings3D dataset, both the non-occluded KITTI_s and occluded KITTI_o datasets are created.

Results. The generalization results on KITTI_s and KITTI_o are listed in Table 1 and 2, respectively. Our method significantly outperforms other methods on all metrics by a large margin. Table 1 represents that the model outperforms the previous state-of-the-art method by 44% on the main EPE3D metric. Compared with previous coarse-to-fine network, PointPWC-Net [46], our method achieves

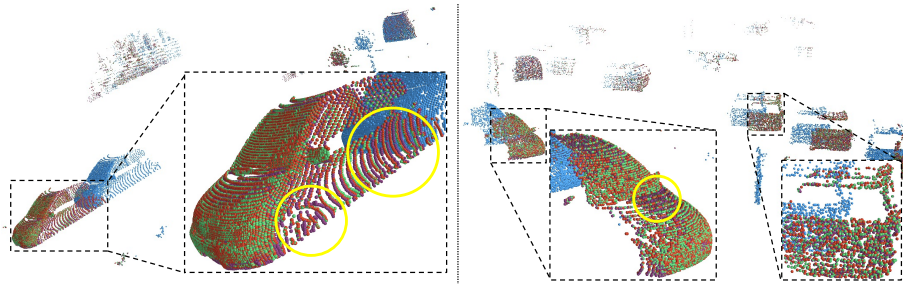

Fig. 4. Qualitative results of Bi-PointFlowNet on the non-occluded KITTI$_s$ dataset. Points are colored to indicate points as from source frame, target frame, unidirectional PointPWCNet estimated points (source frame + scene flow) or as bidirectional Bi-PointFlowNet estimated points (source frame + scene flow).

56% error reduction. Meanwhile, Table 2 shows that our model outperforms the previous state-of-the-art method by 21% of error reduction. In addition, we present the qualitative results on the non-occluded cases of the KITTI$_s$ dataset in Fig. 4. The results show that our Bi-PointFlowNet reduced the estimation error for all points compared to the unidirectional coarse-to-fine PointPWC-Net. Furthermore, Bi-PointFlowNet is able to keep more accurate surface and contour details than PointPWC-Net (marked in the yellow circles in Fig. 4).

5.5 Ablation Study

Ablation of the bidirectional flow embedding layer. As described in Sec. 4.2, the key component of the proposed bidirectional flow embedding layer is the bidirectional feature propagation module, which is followed by the conventional unidirectional flow embedding layer. To evaluate the contribution of BFP, we implemented an ablation model that removes the BFP module resulting in a unidirectional network. We compare the performance of this ablation model with our proposed full model in Table 3. The results show that the proposed BFP module significantly improved the performance on all metrics with a large margin. Especially, the EPE3D error of the generality test on the KITTI$_s$ dataset was reduced by 43%, which shows important implications in the real-world applications. In addition, the ablation model without BFP and original PointPWC-Net are both coarse-to-fine architectures. However, due to the introduction of the residual in the flow predictor, the ablation model still outperformed PointPWC-Net, according to Table 1 and 3.

Ablation of the decomposed form for BFE. We performed two comparative experiments to evaluate the effectiveness and efficiency of the proposed decomposed form of BFE. The one is Bi-PointFlowNet with original BFE (Sec. 4.2) and the other one is the model with the decomposed BFE (Sec. 4.3). Table 4 shows that the model using the decomposed form significantly reduces the total

Dataset	BFP	EPE3D (m) ↓	ACC3D S ↑	ACC3D R ↑	Outliers 3D ↓	EPE2D (px) ↓	ACC 2D ↑
$FT3D_s$	×	0.042	0.836	0.962	0.263	2.270	0.882
	√	**0.028**	**0.918**	**0.978**	**0.143**	**1.582**	**0.929**
$KITTI_s$	×	0.053	0.858	0.930	0.194	1.894	0.880
	√	**0.030**	**0.920**	**0.960**	**0.141**	**1.056**	**0.949**

Table 3. Ablation of the bidirectional flow embedding layer. BFP indicates whether the BFP module is used. All methods are trained only on the FlyingThings3D dataset.

Dataset	Decomp.	EPE3D (m) ↓	ACC3D S ↑	ACC3D R ↑	Outliers 3D ↓	EPE2D (px) ↓	ACC 2D ↑	GFLOPs	Runtime (ms)
$FT3D_s$	×	0.029	0.917	0.977	**0.142**	1.633	0.928	23.8	61.2
	√	**0.028**	**0.918**	**0.978**	0.143	**1.582**	**0.929**	13.3	40.5
$KITTI_s$	×	0.030	**0.925**	**0.965**	**0.133**	1.079	**0.951**	23.8	61.2
	√	**0.030**	0.920	0.960	0.141	**1.056**	0.949	13.3	40.5

Table 4. Ablation of the decomposed form of the bidirectional flow embedding layer. Decomp. indicates whether using decomposed form of BFE. GFLOPs indicates the total operation count. All methods are trained only on the $FT3D_s$ dataset.

Model	FT3D EPE3D (m)	KITTI EPE3D (m)	Param size (M)
PointPWC	0.059	0.069	7.72M
PointPWC + BFP	**0.051**	**0.059**	7.98M
FlowNet3D	0.157	0.173	1.23M
Deeper FlowNet3D	0.160	0.197	1.33M
FlowNet3D + BFP	**0.138**	**0.118**	1.33M

Table 5. Comparison of the bidirectional feature propagation on PointPWC and FlowNet3D. Although the selected baselines showed strong performance, our proposed BFP still reduced the errors by large margins.

operation count by 44% and accelerates the inference by 33% while maintaining the accuracy, compared with the original model.

Ablation of our contribution to FlowNet3D and PointPWC. We validate the contribution of the proposed bidirectional learning method by applying the BFP module into other state-of-the-art methods, FlowNet3D [21] and PointPWC [46]. We built two models by directly inserting BFPs before their flow correlation modules. Since adding BFP requires additional parameters, we also implemented a deeper FlowNet3D network with an equivalent amount of parameters as the model with BFP. Please note that, the experiments related to FlowNet3D were evaluated on the occluded datasets while the PointPWC-based

Method	PointPWC [46]	FLOT [30]	FlowStep3D [16]	Ours
Runtime (ms)	51.3	289.6	820.8	40.5

Table 6. Runtime comparison. The results are evaluated on a single TITAN RTX GPU.

experiments were tested on the non-occluded datasets. Table 5 indicates that the proposed BFP achieves the excellent efficiency and effectiveness. With 0.2M (only 3% of total) additional parameters to the PointPWC, the performance is improved with a 13% error reduction. Moreover, the combination of FlowNet3D and BFP significantly reduce the generalization error by 31%. Furthermore, the ablation of the deeper FlowNet3D reveals the improved performance is owing to the bidirectional strategy rather than an effect of increased number of parameters.

5.6 Runtime

We compare the running time of our proposed methods to that of other state-of-the-art approaches in Table 6. We measured the runtimes of all methods on a single NVIDIA TITAN RTX GPU. The model ran in 40.5 ms, which is faster than the coarse-to-fine PointPWC [46] due to the use of the BFE decomposition. Moreover, compared with other recent advanced approaches [30, 16], our methods outperformed by a large margin in terms of running time while achieving superior accuracy and generality.

6 Conclusion

We presented Bi-PointFlowNet for accurate and fast scene flow estimation. Our proposed network leverages a novel bidirectional flow embedding module that worked with hierarchical feature extraction and propagation to accurately estimate flow. For further accelerating inference, the proposed method applied the decomposed form of the bidirectional flow embedding layer that removes the redundant computations. Experimental results on two challenging datasets showed our network significantly outperformed previous state-of-the-art methods under both non-occluded and occluded conditions. The proposed models also demonstrated excellent time efficiency, allowing the models to be further applied to resource-limited devices, such as wearable devices, drones, IoT edge devices, etc.

Acknowledgement. This work was partly supported by the National Research Foundation (NRF) grants (2022R1F1A1074142, 2022R1A4A3032913) and Institute of Information and Communication Technology Planning & Evaluation (IITP) grants (IITP-2019-0-00421, IITP-2020-0-00821, IITP-2021-0-02052, IITP-2021-0-02068), funded by the MSIT (Ministry of Science and ICT) of Korea. Wencan Cheng was partly supported by the China Scholarship Council (CSC).

References

1. Baldi, P., Brunak, S., Frasconi, P., Soda, G., Pollastri, G.: Exploiting the past and the future in protein secondary structure prediction. Bioinformatics **15**(11), 937–946 (1999)
2. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. International Society for Optics and Photonics (1992)
3. Čech, J., Sanchez-Riera, J., Horaud, R.: Scene flow estimation by growing correspondence seeds. In: CVPR 2011. pp. 3129–3136. IEEE (2011)
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
5. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and vision computing **10**(3), 145–155 (1992)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
7. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015)
8. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural networks **18**(5-6), 602–610 (2005)
9. Gu, X., Wang, Y., Wu, C., Lee, Y.J., Wang, P.: Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3254–3263 (2019)
10. Huguet, F., Devernay, F.: A variational method for scene flow estimation from stereo sequences. In: 2007 IEEE 11th International Conference on Computer Vision. pp. 1–7. IEEE (2007)
11. Hur, J., Roth, S.: Mirrorflow: Exploiting symmetries in joint optical flow and occlusion estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 312–321 (2017)
12. Hur, J., Roth, S.: Iterative residual refinement for joint optical flow and occlusion estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5754–5763 (2019)
13. Jampani, V., Kiefel, M., Gehler, P.V.: Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4452–4461 (2016)
14. Janai, J., Guney, F., Ranjan, A., Black, M., Geiger, A.: Unsupervised learning of multi-frame optical flow with occlusions. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 690–706 (2018)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Kittenplon, Y., Eldar, Y.C., Raviv, D.: Flowstep3d: Model unrolling for self-supervised scene flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4114–4123 (2021)

17. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
18. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics **36**(4), 1234–1240 (2020)
19. Li, R., Lin, G., He, T., Liu, F., Shen, C.: Hcrf-flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 364–373 (2021)
20. Liu, P., Lyu, M., King, I., Xu, J.: Selflow: Self-supervised learning of optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4571–4580 (2019)
21. Liu, X., Qi, C.R., Guibas, L.J.: Flownet3d: Learning scene flow in 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 529–537 (2019)
22. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
23. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4040–4048 (2016)
24. Melamud, O., Goldberger, J., Dagan, I.: context2vec: Learning generic context embedding with bidirectional lstm. In: Proceedings of the 20th SIGNLL conference on computational natural language learning. pp. 51–61 (2016)
25. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3061–3070 (2015)
26. Menze, M., Heipke, C., Geiger, A.: Object scene flow. ISPRS Journal of Photogrammetry and Remote Sensing **140**, 60–76 (2018)
27. Ouyang, B., Raviv, D.: Occlusion guided scene flow estimation on 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2805–2814 (2021)
28. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
29. Pons, J.P., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. International Journal of Computer Vision **72**(2), 179–193 (2007)
30. Puy, G., Boulch, A., Marlet, R.: Flot: Scene flow on point clouds guided by optimal transport. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16. pp. 527–544. Springer (2020)
31. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
32. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
33. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing **45**(11), 2673–2681 (1997)

34. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2530–2539 (2018)
35. Tam, G.K., Cheng, Z.Q., Lai, Y.K., Langbein, F.C., Liu, Y., Marshall, D., Martin, R.R., Sun, X.F., Rosin, P.L.: Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. IEEE transactions on visualization and computer graphics **19**(7), 1199–1217 (2012)
36. Teed, Z., Deng, J.: Raft-3d: Scene flow using rigid-motion embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8375–8384 (2021)
37. Tishchenko, I., Lombardi, S., Oswald, M.R., Pollefeys, M.: Self-supervised learning of non-rigid residual flow and ego-motion. In: 2020 International Conference on 3D Vision (3DV). pp. 150–159. IEEE (2020)
38. Valgaerts, L., Bruhn, A., Zimmer, H., Weickert, J., Stoll, C., Theobalt, C.: Joint estimation of motion, structure and geometry from stereo sequences. In: European Conference on Computer Vision. pp. 568–581. Springer (2010)
39. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. vol. 2, pp. 722–729. IEEE (1999)
40. Vogel, C., Schindler, K., Roth, S.: Piecewise rigid scene flow. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1377–1384 (2013)
41. Wang, G., Hu, Y., Wu, X., Wang, H.: Residual 3d scene flow learning with context-aware feature extraction. arXiv preprint arXiv:2109.04685 (2021)
42. Wang, G., Wu, X., Liu, Z., Wang, H.: Hierarchical attention learning of scene flow in 3d point clouds. IEEE Transactions on Image Processing **30**, 5168–5181 (2021)
43. Wang, Y., Yang, Y., Yang, Z., Zhao, L., Wang, P., Xu, W.: Occlusion aware unsupervised learning of optical flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4884–4893 (2018)
44. Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D.: Efficient dense scene flow from sparse or dense stereo data. In: European conference on computer vision. pp. 739–751. Springer (2008)
45. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9621–9630 (2019)
46. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In: European Conference on Computer Vision. pp. 88–107. Springer (2020)
47. Wu, X., Lv, S., Zang, L., Han, J., Hu, S.: Conditional bert contextual augmentation. In: International Conference on Computational Science. pp. 84–95. Springer (2019)
48. Yang, W., Zhang, H., Lin, J.: Simple applications of bert for ad hoc document retrieval. arXiv preprint arXiv:1903.10972 (2019)
49. Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B.: Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers). pp. 207–212 (2016)