SCTN: Sparse Convolution-Transformer Network for Scene Flow Estimation

Bing Li, Cheng Zheng, Silvio Giancola, Bernard Ghanem

King Abdullah University of Science and Technology
{bing.li, cheng.zheng, silvio.giancola, Bernard.Ghanem}@kaust.edu.sa

Abstract

We propose a novel scene flow estimation approach to capture and infer 3D motions from point clouds. Estimating 3D motions for point clouds is challenging, since a point cloud is unordered and its density is significantly non-uniform. Such unstructured data poses difficulties in matching corresponding points between point clouds, leading to inaccurate flow estimation. We propose a novel architecture named Sparse Convolution-Transformer Network (SCTN) that equips the sparse convolution with the transformer. Specifically, by leveraging the sparse convolution, SCTN transfers irregular point cloud into locally consistent flow features for estimating continuous and consistent motions within an object/local object part. We further propose to explicitly learn point relations using a point transformer module, different from exiting methods. We show that the learned relation-based contextual information is rich and helpful for matching corresponding points, benefitting scene flow estimation. In addition, a novel loss function is proposed to adaptively encourage flow consistency according to feature similarity. Extensive experiments demonstrate that our proposed approach achieves a new state of the art in scene flow estimation. Our approach achieves an error of 0.038 and 0.037 (EPE3D) on FlyingThings3D and KITTI Scene Flow respectively, which significantly outperforms previous methods by large margins.

Introduction

Understanding 3D dynamic scenes is critical to many real-world applications such as autonomous driving and robotics. Scene flow is the 3D motion of points in a dynamic scene, which provides low-level information for scene understanding (Vedula et al. 1999; Liu, Qi, and Guibas 2019; Geiger et al. 2013). The estimation of the scene flow can be a building block for more complex applications and tasks such as 3D object detection (Shi et al. 2020), segmentation (Thomas et al. 2019) and tracking (Qi et al. 2020). However, many previous scene flow methods estimate the 3D motion from stereo or RGB-D images. With the increasing popularity of point cloud data, it is desirable to estimate 3D motions directly from 3D point clouds.

Recent methods e.g.,(Wu et al. 2020; Liu, Qi, and Guibas 2019; Wei et al. 2021; Puy, Boulch, and Marlet 2020; Wang et al. 2021; Li et al. 2021) propose deep neural networks to learn scene flow from point clouds in an end-to-end way, which achieves promising estimation performance. However, estimating scene flow from point clouds is still challenging. In particular, existing methods (Liu, Qi, and Guibas 2019; Puy, Boulch, and Marlet 2020) extract feature for each point by aggregating information from its local neighborhood. However, such extracted features are not discriminative enough to matching corresponding points between point clouds (see Figure 1), leading to inaccurate flow estimation, since the feature extraction of these methods ignore two facts. First, the density of points is significantly non-uniform within a point cloud. It is non-trivial to learn point features that are simultaneously favorable for both points from dense regions and those from sparse regions. Second, due to Lidar and object motions, the density of points within an object often varies at the temporal dimension, leading that the geometry patterns of corresponding local regions are inconsistent between consecutive point clouds. As a result, extracted features, that are aggregated from only local regions with-
Our contributions are fourfold:

1. Instead of designing convolution kernels, our VIFE module leverages simple operators – voxelization and interpolation for feature extraction, showing such smoothing operator is effective to extract local consistent while discriminative features for scene flow estimation.
2. Our PTFE module shows that explicitly modeling point relations can provide rich contextual information and is helpful for matching corresponding points, benefiting scene flow estimation. We are the first to introduce transformer for scene flow estimation.
3. We propose a new consistency loss equipping stop-gradient-based architecture that helps the model trained on synthetic dataset well adapt to real data, by controlling spatial consistency of estimated flows.
4. We propose a novel network that outperforms the state-of-the-art methods with remarkable margins on both FlyingThings3D and KITTI Scene Flow benchmarks.

Related Work

**Optical Flow.** Optical flow estimation is defined as the task of predicting the pixels motions between consecutive 2D video frames. Optical flow is a fundamental tool for 2D scene understanding, that have been extensively studied in the literature. Traditional methods (Horn and Schunck 1981; Black and Anandan 1993; Zach, Pock, and Bischof 2007; Weinaebergel et al. 2013; Brox, Bregler, and Malik 2009; Ranftl, Bredies, and Pock 2014) address the problem of estimating optical flow as an energy minimization problem, that does not require any training data. Dosovitskiy et al. (Dosovitskiy et al. 2015) proposed a first attempt for an end-to-end model to solve optical flow based on convolution neural network (CNN). Inspired by this work, many CNN-based studies have explored data-driven approaches for optical flow (Dosovitskiy et al. 2015; Mayer et al. 2016; Ilg et al. 2017; Hui, Tang, and Loy 2018; Hui and Loy 2020; Sun et al. 2018; Teed and Deng 2020).

**Scene Flow from Stereo and RGB-D Videos.** Estimating scene flow from stereo videos has been studied for years (Chen et al. 2020; Vogel, Schindler, and Roth 2013; Wedel et al. 2008; Ilg et al. 2018; Jiang et al. 2019; Teed and Deng 2020a). Many works estimate scene flow by jointly estimating stereo matching and optical flow from consecutive stereo frames (N.Mayer et al. 2016b). Similar to optical flow, traditional methods formulate scene flow estimation as an energy minimization problem (Huguet and Devernay 2007; Wedel et al. 2008). Recent works estimate scene flow from stereo videos using neural networks (Chen et al. 2020). For example, networks for disparity estimation and optical flow are combined in (Ilg et al. 2018; Ma et al. 2019). Similarly, other works (Quiroga et al. 2014; Sun, Sudderth, and Pfister 2015) explore scene flow estimation from RGB-D video.

**Scene Flow on Point Clouds.** Inspired by 
FlowNet (Dosovitskiy et al. 2015), FlowNet3D (Liu, Qi, and Guibas 2019) propose an end-to-end network to estimate 3D scene flow from raw point clouds. Different from traditional methods (Dewan et al. 2016; Ushani et al. 2017), FlowNet3D (Liu, Qi, and Guibas 2019) is based on PointNet++ (Qi et al. 2017b), and propose a flow embedding layer to aggregate the information from consecutive point clouds and extract scene flow with convolutional layers. FlowNet3D++ (Wang et al. 2020) improves the accuracy of FlowNet3D by incorporating geometric constraints. HPLFlowNet (Gu et al. 2019) projects point clouds into permutohedral lattices, and then estimates scene flow using Bilateral Convolutional Layers. Inspired by the successful optical flow method PWC-Net (Sun et al. 2018), PointPWC (Wu et al. 2020) estimates scene flow in a coarse-to-fine fashion, introducing cost volume, upsampling, and warping.
modules for the point cloud processing. The most related recent work to our approach is FLOT (Puy, Boulch, and Marlet 2020). FLOT addresses the scene flow as a matching problem between corresponding points in the consecutive clouds and solve it using optimal transport. Our method differs from FLOT (Puy, Boulch, and Marlet 2020) in two aspects. First, our method explicitly explores more suitable feature representation that facilitate the scene flow estimation. Second, our method is trained to enforce the consistency of predicted flows for local-region points from the same object, which is ignored in FLOT (Puy, Boulch, and Marlet 2020). Recently, Gojcic et al. (Gojcic et al. 2021) explore weakly supervised learning for scene flow estimation using labels of ego motions as well as ground-truth foreground and background masks. Other works (Wu et al. 2020; Kittenplon, Eldar, and Raviv 2021; Mittal, Okorn, and Held 2020) study unsupervised/self-supervised learning for scene flow estimation on point clouds, proposing regularization losses that enforces local spatial smoothness of predicted flows. These losses are directly constraining points in a local region to have similar flows, but are not feature-aware.

3D Deep Learning. Many works have introduced deep representation for point cloud classification and segmentation (Qi et al. 2017b; Thomas et al. 2019; Zhang, Hua, and Yeung 2019; Liu et al. 2020; Wu, Qi, and Fuxin 2019; Li et al. 2018; Lei, Akhtar, and Mian 2020; Hu et al. 2022; Zhao et al. 2021). Qi et al. (Qi et al. 2017a) propose PointNet that learns point feature only from point positions. PointNet++ (Qi et al. 2017b) extends PointNet by aggregating information from local regions. Motivated by PointNet++, many works (Zhang, Hua, and Yeung 2019; Thomas et al. 2019; Lei, Akhtar, and Mian 2020; Li et al. 2018) design various local aggregation functions for point cloud classification and segmentation. Different from these point-based convolutions, (Wu et al. 2015; Wang et al. 2017) transform point cloud into voxels, such that typical 3D convolution can be applied. However, such voxel-based convolution suffers from expensive computational and memory cost as well as information loss during voxelization. Liu et al. (Liu et al. 2019) combine PointNet (Qi et al. 2017a) with voxel-based convolution to reduce the memory consumption. For the sake of efficient learning, researchers have explored sparse convolution for point cloud segmentation (Xie et al. 2020; Choy, Gwak, and Savarese 2019) which shows impressive performance. Tang et al. (Tang et al. 2020) propose to combine PointNet with sparse convolution for large-scale point cloud segmentation. Differently, we not only leverage voxelization and interpolation for feature extraction, but also explicitly model point relations to provide complementary information.

Methodology

Problem Definition. Given two consecutive point clouds \( \mathcal{P}^t \) and \( \mathcal{P}^{t+1} \), scene flow estimation is to predict the 3D motion flow of each point from \( \mathcal{P}^t \) to \( \mathcal{P}^{t+1} \). Let \( p_i^t \) be the 3D coordinates of \( i \)-th point in \( \mathcal{P}^t = \{ p_i^t \}_{i=1}^n \). Like previous work (Puy, Boulch, and Marlet 2020), we predict scene flow based on the correlations of each point pair between \( \mathcal{P}^t \) and \( \mathcal{P}^{t+1} \). Given a pair of points \( p_i^t \in \mathcal{P}^t \) and \( p_j^{t+1} \in \mathcal{P}^{t+1} \), the correlation of the two points is computed as follows:

\[
C(p_i^t, p_j^{t+1}) = \frac{(F_i^t)^T \cdot F_j^{t+1}}{||F_i^t||_2 ||F_j^{t+1}||_2}
\]

(1)

where \( F_j^{t+1} \) is the feature of \( p_j^{t+1} \), and \( || \cdot ||_2 \) is the L2 norm.

Point feature \( F \) is the key to computing the correlation \( C(p_i^t, p_j^{t+1}) \) which further plays an important role in scene flow estimation. Hence, it is desirable to extract effective point features that enable point pairs in corresponding regions to achieve higher correlation values between \( \mathcal{P}^t \) and \( \mathcal{P}^{t+1} \). However, different from point cloud segmentation and classification that focus on static point cloud, scene flow estimation operates on dynamic ones, which poses new challenges for feature extraction in two aspects. For example, the input point clouds of scene flow are not only irregular and unordered, but also its density is spatially non-uniform and temporally varying, as discussed in previous sections.

Our goal is to extract locally consistent while discriminative features for points, so as to achieve accurate flow estimation. Different from exiting methods directly extracting
point feature from the neighborhood in original point cloud, we proposed two feature extraction modules for scene flow estimation. The Voxelization-Interpolation based Feature Extraction (VIFE) is proposed to address the issue of point cloud’s non-uniform density. With the features extracted by VIFE, Point Transformer based Feature Extraction (PTFE) further enhance feature discriminability by modeling point relations globally. Since the two kinds of features provide complementary information, we fuse these features, such that the fused features provide proper correspondences to predict accurate scene flows.

Overview. Figure 2 illustrates the overall framework of our approach, that takes two consecutive point clouds $\mathcal{P}^t$ and $\mathcal{P}^{t+1}$ as inputs and predict the scene flow from $\mathcal{P}^t$ to $\mathcal{P}^{t+1}$. First, the VIFE module extracts features for voxel points from voxelized point clouds, and projects back the voxel features into the original 3D points to obtain locally consistent point features. Second, our PTFE module adopt the point feature representation by modeling relation between points. Third, with features extracted by VIFE and PTFE module, we calculate the correlations of points between $\mathcal{P}^t$ and $\mathcal{P}^{t+1}$, where a sinkhorn algorithm (Puy, Bouchl, and Marlet 2020; Cuturi 2013; Chizat et al. 2018) is leveraged to predict the flows. We train our method with an extra regularizing loss to enforce spatial consistency of predicted flows.

Voxelization-Interpolation Based Feature Extraction

As mentioned in previous sections, the density of consecutive point clouds is spatially non-uniform and temporally varying, posing difficulties in feature extraction. To address the issue, we propose the Voxelization-Interpolation based Feature Extraction (VIFE) module. VIFE first voxelizes the consecutive point clouds into voxels. As illustrated in Figure 2, the spatial non-uniform distributions and temporally variations of points are reduced to some extent.

After that, VIFE conducts convolutions on voxel points rather than all points of the point cloud, and then interpolate features for each point. We argue that such simple operators i.e., voxelization and interpolation, ensure points in a local neighborhood to have smoother features, ideally leading to consistent flows in space.

Voxel feature extraction. We then leverage a U-Net (Ronneberger, Fischer, and Brox 2015) architecture network to extract feature from voxelized point clouds, where convolution can be many types of point cloud convolutions such as pointnet++ used in FLOT (Puy, Bouchl, and Marlet 2020). Here, we adopt sparse convolution e.g., Minkowski Engine (Choy, Gwak, and Savarese 2019) for efficiency. More details are available in our supplementary material.

Point feature interpolation. We project back the voxel features into point feature $\mathbf{F}_i^S$ for point $p_i$. In particular, we interpolate the point features from the $K$ closest voxels following equation (2), $\mathcal{N}^v(p_i)$ represents the set of $K$ nearest neighboring non-empty voxels for the point $p_i$, $\mathbf{v}_k \in \mathbb{R}^C$ represents the feature of $k$-th closest non-empty voxel and $d_{ik}$ the Euclidian distance between the point $p_i$ and the center of the $k$-th closest non-empty voxel.

$$\mathbf{F}_i^S = \frac{\sum_{k \in \mathcal{N}^v(p_i)} d_{ik}^{-1} \cdot \mathbf{v}_k}{\sum_{k \in \mathcal{N}^v(p_i)} d_{ik}^{-1}} \tag{2}$$

We observed that close points are encouraged to have similar features, which helps our method generate consistent flows for these points. This is favorable for local object parts or rigid objects with dense densities and consistent flows (e.g., LiDAR points on a car at close range).

Point Transformer Based Feature Extraction

Our VIFE module adopt aggressive downsampling to obtain a large receptive field and low computation cost. However, aggressive downsampling inevitably loses some important information (Tang et al. 2020). In such case, the features of points with large information loss are disadvantageous for estimating their scene flow. To address this issue, we explicitly exploit point relations as a complementary information on top of the point feature extracted with VIFE. Recent work (Zhu et al. 2021; Zhao, Jia, and Koltun 2020; Carion et al. 2020) employ transformer and self-attention to model internal relation in the features, achieving impressive performance in image tasks such as detection and recognition. Similar trend appeared in point cloud classification and segmentation (Guo et al. 2021; Zhao et al. 2021; Engel, Belagiannis, and Dietmayer 2020) showing the effectiveness of transformer in 3D. Inspired by these work, we resort to transformer for capturing point relation information as the point feature.

In an autonomous navigation scenario, point clouds represent complete scenes, with small object such as cars and trucks, but also large structure such as buildings and walls. The scene flows in such a large scene do not only depend on the aggregation from a small region, but rather a large one. As a results, we refrain in building a transformer for the local neighborhood as it would restrict the receptive field or require deeper model (i.e., increase the memory). Instead, our transformer module learns the relation of each point to all other points, such that the transformer can adaptively capture the rich contextual information from a complete scene.

Formally, given a point $p_i$, we consider every points in $\mathcal{P}$ as query and key elements. Our transformer module builds a point feature representation for $p_i$ by adaptively aggregating the features of all points based on self-attention:

$$\mathbf{F}_i^R = \sum_{j=1}^{n_P} \mathbf{A}_{i,j} \cdot g_v(\mathbf{F}_j^S, \mathbf{G}_j) \tag{3}$$

where $g_v$ is the a learnable function (e.g., linear function), $A_{i,j}$ is an attention defining a weight of $p_j$ to $p_i$, $\mathbf{G}_j$ is the positional encoding feature of $p_j$.

As pointed in literature (Zhao, Jia, and Koltun 2020; Carion et al. 2020), the positional encoding feature can provide important information for the transformer. The position encoding in recent transformer work (Zhu et al. 2021) encodes the relative point position to neighbors for point cloud classification or segmentation. Different from those tasks, the task of scene flow is to find correspondences between consecutive point clouds. Thus, we argue that an absolute position provides sufficient information to estimate
the scene flow. Therefore, given a point \( p \), our position encoding function encodes its absolute position \( p_j \):

\[
G_j = \phi(p_j)
\]

where \( \phi \) is a MLP layer. Using absolute positions reduce computational cost, compared with using relative positions.

We calculate an attention \( A_{i,j} \) as the similarity between the features of \( p_i \) and \( p_j \) in an embedding space. The similarity is estimated using features and position information:

\[
A_{i,j} \propto \exp\left( \frac{g_q(F_i^S, G_i^s) \cdot F_j^S + g_k(F_j^S, G_j^s)}{c_a} \right)
\]

where \( g_q(\cdot, \cdot) \) and \( g_k(\cdot, \cdot) \) are the learnable mapping functions to project feature into an embedding space, and \( c_a \) is the output dimension of \( g_q(\cdot, \cdot) \) or \( g_k(\cdot, \cdot) \). \( A_{i,j} \) is further normalized such that \( \sum_j A_{i,j} = 1 \). The architecture of our transformer module is illustrated in Figure 3.

**Flow Prediction**

Since the point feature from our VIFE and PTFE modules provide complementary information, we fuse the two kinds of features through skip connection for each point, i.e., \( F_i = F_i^S + F_i^R \). By feeding the fused point features into Eq. 1, we compute the correlations of all pairs \( C(P^i, P^{i+1}) = (C(p_i^t, p_{i+1}^t)) \) between the two consecutive point clouds.

With the estimated point correlations, we adopt the Sinkhorn algorithm to estimate soft correspondences and predict flows for \( P^t \), following FLOT.

**Training Losses**

We train our model to regress the scene flow in a supervised fashion, on top of which we propose a Feature-aware Spatial Consistency loss, named “FSC loss”, that enforces similar features to have similar flow. The FSC loss provides a better generalization and transfer capability between the training and testing datasets.

**Supervised loss.** We define in Equation (6) our supervised loss \( E^s \) that minimize the \( L_1 \)-norm difference between the estimated flow and the ground truth flow for the non-occluded points. \( u_i^* \) and \( u_i \) are respectively the ground-truth and predicted motion flow for the point \( p_i \in P \) and \( m_i \) is a binary indicator for the non-occlusion of this point, i.e., \( m_i = 0 \) if \( p_i \) is occluded, otherwise \( m_i = 1 \).

\[
E^s = \sum_i m_i \| u_i - u_i^* \|
\]

**Feature-aware Spatial Consistency (FSC) loss**. Given a local region, points from the same object usually has consistent motions, resulting in similar flows. To model such phenomena, we propose a consistency loss that ensures points within an object/local object part to have similar predicted flows. Yet, object annotations are not necessarily available. Instead, we propose to control flow consistency according to feature similarity. That is, given a local region, if two points are of larger feature similarity, they are of the higher probability that belongs to the same object. In particular, given a point \( p_i \) with predicted flow \( u_i \) and its local neighborhood \( N(p_i) \), we enforce the flow \( u_j \) of \( p_j \in N(p_i) \) to be similar to \( u_i \), if the feature \( F_j \) of \( p_j \) is similar to \( F_j \) of \( p_j \). Formally, we define the FSC loss as follows:

\[
E^c = \sum_i \frac{1}{K} \sum_{p_j \in N(p_i)} s(F_i, F_j) \cdot \| u_i - u_j \|_2
\]

where the similarity function \( s(F_i, F_j) \) of \( p_i \) and \( p_j \) is defined as \( 1 - \exp(-|F_i - F_j|/\tau) \) with \( \tau \) being a temperature hyper-parameter, \( K \) is the number of points in \( N(p_i) \).

A naive implementation of the FSC loss would inevitably lead to degenerate cases. In particular, the FSC loss is a product of two objectives: (i) a similarity \( s(F_i, F_j) \) between the features \( F_i \) and \( F_j \) and (ii) a difference \( \| u_i - u_j \|_1 \) between their flows. The scope of this loss is to train the flows to be similar if they have similar features. However, to minimize the FSC loss, the model would make the features \( F_i \) and \( F_j \) be orthogonal (i.e., \( F_j \cdot F_i = 0 \)), such that \( s(F_j, F_i) = 0 \) (i.e., \( E^c = 0 \)). Obviously, it is against our aim.

To circumvent this limitation, we propose a stop-gradient for the FSC loss, taking inspiration form recent advances in
self-supervised learning (Chen and He 2020). As illustrated in Figure 4, our architecture stops the propagation of the gradient in the branch extracting the feature similarity. By such architecture, our FSC loss avoids optimizing the features, while optimizing solely the flows similarities \( \|u_j - u_i\|_1 \) for neighboring points with similar features.

## Experiments

### Dataset

We conduct our experiments on two datasets that are widely used to evaluate scene flow. **FlyingThings3D** (N.Mayer et al. 2016a) is a large-scale synthetic stereo video datasets, where synthetic objects are selected from ShapeNet (Chang et al. 2015) and randomly assigned various motions. We generate 3D point clouds and ground truth scene flows with their associated camera parameters and disparities. Following the same preprocessing as in (Puy, Boulch, and Marlet 2020; Gu et al. 2019; Wu et al. 2020), we randomly sample 8192 points and remove points with camera depth greater than 35 m. We use the same 19640/3824 pairs of point cloud (training/testing) used in the related works (Puy, Boulch, and Marlet 2020; Gu et al. 2019; Wu et al. 2020). **KITTI Scene Flow** (Menze, Heipke, and Geiger 2018; Choy, Gwak, and Savarese 2019) is a real-world Lidar scan dataset for scene flow estimation from the KITTI autonomous navigation suite. Following the preprocessing of (Gu et al. 2019), we leverage 142 point cloud pairs of 8192 points for testing. For a fair comparison, we also remove ground points by discarding points whose height is lower than \(-1.4\text{m}\), following the setting of existing methods (Puy, Boulch, and Marlet 2020; Wu et al. 2020; Gu et al. 2019).

### Evaluation Metrics

To evaluate the performance of our approach, we adopt the standard evaluation metrics used in the related methods (Puy, Boulch, and Marlet 2020; Wu et al. 2020), described as follows: The \( \text{EPE3D}(m) \) (3D end-point-error) is calculated by computing the average \( L_2 \) distance between the predicted and GT scene flow, in meters. This is our main metric. The \( \text{Acc3DS} \) is a strict version of the accuracy which estimated as the ratio of points whose EPE3D < 0.05 m or relative error < 5\%. The \( \text{Acc3DR} \) is a relaxed accuracy which is calculated as the ratio of points whose EPE3D < 0.10m or relative error < 10\%. The \( \text{Outliers} \) is the ratio of points whose EPE3D > 0.30m or relative error > 10\%.

### Implementation Details

We implement our method in PyTorch (Paszke et al. 2019). We train our method on FlyingThings3D then evaluate on FlyingThings3D and KITTI. We minimize a cumulative loss \( E = E_s + \lambda E_c \) with \( \lambda = 0.30 \) a weight that scale the losses. We use the Adam optimizer (Kingma and Ba 2014) with an initial learning rate of \( 10^{-3} \), which is dropped to \( 10^{-4} \) after the 50th epoch. First, we train for 40 epochs only using the supervised loss. Then we continue the training for 20 epochs with both the supervision loss and the FSC loss, for a total on 60 epochs. We

### Table 1: Comparison with the state-of-the-art on FlyingThings3D and KITTI

Best results in bold. Our proposed model SCTN reaches highest performances in all metrics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>EPE3D(m)</th>
<th>Acc3DS</th>
<th>Acc3DR</th>
<th>Outliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlyingThings3D</td>
<td>FlowNet3D (Liu, Qi, and Guibas 2019)</td>
<td>0.114</td>
<td>0.412</td>
<td>0.771</td>
<td>0.602</td>
</tr>
<tr>
<td></td>
<td>HPLFlowNet (Gu et al. 2019)</td>
<td>0.080</td>
<td>0.614</td>
<td>0.855</td>
<td>0.429</td>
</tr>
<tr>
<td></td>
<td>PointPWC (Wu et al. 2020)</td>
<td>0.059</td>
<td>0.738</td>
<td>0.928</td>
<td>0.342</td>
</tr>
<tr>
<td></td>
<td>EgoFlow (Tishchenko et al. 2020)</td>
<td>0.069</td>
<td>0.670</td>
<td>0.879</td>
<td>0.404</td>
</tr>
<tr>
<td></td>
<td>FLOT (Puy, Boulch, and Marlet 2020)</td>
<td>0.052</td>
<td>0.732</td>
<td>0.927</td>
<td>0.357</td>
</tr>
<tr>
<td></td>
<td>SCTN (ours)</td>
<td><strong>0.038</strong></td>
<td><strong>0.847</strong></td>
<td><strong>0.968</strong></td>
<td><strong>0.268</strong></td>
</tr>
</tbody>
</table>

| KITTI        | FlowNet3D (Liu, Qi, and Guibas 2019) | 0.177    | 0.374  | 0.668  | 0.527    |
|              | HPLFlowNet (Gu et al. 2019)   | 0.117    | 0.478  | 0.778  | 0.410    |
|              | PointPWC (Wu et al. 2020)     | 0.069    | 0.728  | 0.888  | 0.265    |
|              | EgoFlow (Tishchenko et al. 2020) | 0.103    | 0.488  | 0.822  | 0.394    |
|              | FLOT (Puy, Boulch, and Marlet 2020) | 0.056    | 0.755  | 0.908  | 0.242    |
|              | SCTN (ours)                 | **0.037** | **0.873** | **0.959** | **0.179** |

Figure 5: Qualitative comparison results. Green points indicate the first point cloud in (a), and blue points indicate the second point cloud in (a)(b)(c)(d). In (b)(c)(d), green points are the ones in the first point cloud warped by correctly predicted flows, while red points are the ones warped by incorrect flows (the first point cloud + incorrect scene flow whose EPE3D > 0.1m).
use a voxel size of resolution 0.07m.

**Runtime.** We evaluate the running time of our method. Table 3 reports the evaluated time compared with recent state-of-the-art methods. FLOT (Puy, Boulch, and Marlet 2020) is the most related work to our method, since we both adopt point-wise correlations to generate predicted flows. Our method consumes lower running time than FLOT, although the transformer module is equipped.

**Quantitative Evaluation**

We compare our approach with recent deep-learning-based methods including FlowNet3D (Liu, Qi, and Guibas 2019), HPLFlowNet (Gu et al. 2019), PointPWC (Wu et al. 2020) and FLOT (Puy, Boulch, and Marlet 2020). These methods are state-of-the-art in scene flow estimation from point cloud data and do not leverage any additional labels such as ground-truth ego motions or instance segmentation.

**Results on FlyingThings3D.** We train and evaluate our model on the FlyingThings3D datasets. As shown in Table 1, our method outperforms all methods in every metric by a significant margin. It is worth noting that our method obtains an EPE3D metric below 4cm, with a relative improvement of 26.9% and 35.5% over the most recent methods FLOT (Puy, Boulch, and Marlet 2020) and PointPWC (Wu et al. 2020), respectively. The performance shows that our method is effective in predicting flows with high accuracy.

**Results on KITTI without Fine-tune.** Following the common practice (Puy, Boulch, and Marlet 2020; Wu et al. 2020), we train our model on FlyingThings3D and directly test the trained model on KITTI Scene Flow dataset, without any fine-tuning, to evaluate the generalization capability of our method. We report in Table 1 the highest accuracy of scene flow estimation on KITTI Scene Flow dataset for our SCTN method. Again, we reduce the EPE3D metric below 4cm, with a 33.9% relative improvement over FLOT (Puy, Boulch, and Marlet 2020). In the Acc3DS metrics, our method outperforms both FLOT (Puy, Boulch, and Marlet 2020) and PointPWC (Wu et al. 2020) by 13.5% and 16.6% respectively. These results highlight the capability of our method to generalize well on real-world datasets.

**Qualitative Evaluation**

To qualitatively evaluate the quality of our scene flow predictions, we visualize the predicted scene flow and ground-truth one in Figure 5. Since FLOT (Puy, Boulch, and Marlet 2020) is most related to our method, we compare the qualitative performances of our SCTN with FLOT (Puy, Boulch, and Marlet 2020).

<table>
<thead>
<tr>
<th>VIFE</th>
<th>PTFE</th>
<th>FSC loss</th>
<th>EPE3D(m) ↓</th>
<th>Acc3DS ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td>0.045</td>
<td>0.835</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>0.042</td>
<td>0.853</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
<td>0.040</td>
<td>0.863</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.037</td>
<td>0.873</td>
</tr>
</tbody>
</table>

Table 2: Ablation for SCTN. We further analyse the performances of our three components: VIFE, PTFE and FSC loss on KITTI. We highlight in bold the best performances.

Table 3: Running time comparisons. The runtime of FLOT and our SCTN are evaluated on a single GTX2080Ti GPU. We used the official implementation of FLOT.

Table 3: Running time comparisons. The runtime of FLOT and our SCTN are evaluated on a single GTX2080Ti GPU. We used the official implementation of FLOT.

Points in a local region from the same object usually have similar ground-truth flows. Yet, FLOT introduces prediction errors in local regions, highlighting the inconsistency in the scene flow predictions. For example, FLOT inaccurately predicts scene flow for some regions in the background, even though those points have similar flows, as shown in Figure 5. In contrast, our method is more consistent in the prediction for points in the same object, achieving better performance, e.g., for the background objects with complex structure.

**Ablation Study**

To study the roles of the proposed VIFE, PTFE and FSC loss, we ablate each proposed component of our model and evaluate their performance on KITTI. For all the experiments, we follow the same training procedure than in the main results. Table 2 reports the evaluation results.

**VIFE module.** Table 2 shows that our approach with the sole VIEF convolution module already outperforms the state-of-the-art methods listed in Table 1. Different from existing methods directly applying convolution on original point clouds, our VIFE extracts feature from voxelized point cloud, which reduces the non-uniform density of point cloud, while ensuring that points in a local region have consistent features, to some extent. The results show that such features are favorable for scene flow estimation.

**PTFE module.** Compared with only using VIFE module, adding PTFE improves both metrics on KITTI as reported in the third row in Table 2. For example, EPE3D is improved by 11.1%, compared with only using the VIEF module. Our PIFE module explicitly learns point relations, which provides rich contextual information and helps to match corresponding points even for objects with complex structures.

**FSC loss.** Table 2 shows that adding the FSC loss helps to achieve better scene flow estimation on KITTI. Our FSC loss improves the generalization capability of our method.

**Conclusion**

We present a Sparse Convolution-Transformer Network (SCTN) for scene flow estimation. Our SCTN leverages the VIFE module to transfer irregular point cloud into locally smooth flow features for estimating spatially consistent motions in local regions. Our PTFE module learns rich contextual information via explicitly modeling point relations, which is helpful for matching corresponding points and benefits scene flow estimation. A novel FSC loss is also proposed for training SCTN, improving the generalization ability of our method. Our approach achieves state-of-the-art performances on FlyingThings3D and KITTI datasets.
Acknowledgments
This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding. We thank Hani Itani for his constructive suggestions and help.

References
Lei, H.; Akhtar, N.; and Mian, A. 2020. Spherical kernel for efficient graph convolution on 3d point clouds. TPAMI.

N. Mayer; E. Ilg; P. Häs­s­er; P. Fischer; D. Cre­mers; A. Do­so­vi­tski­y; and T. Brox. 2016a. A Large Dataset to Train Convolu­tion­nal Net­works for Dis­par­i­ty, Opt­i­cal Flow, and Scene Flow Estimation. In CVPR.

N. Mayer; E. Ilg; P. Häusser; P. Fischer; D. Cremers; A. Dosovitskiy; and T. Brox. 2016b. A Large Dataset to Train Convolu­tion­nal Net­works for Dis­par­i­ty, Opt­i­cal Flow, and Scene Flow Estimation. In CVPR.


Puy, G.; Boulch, A.; and Marlet, R. 2020. FLOT: Scene Flow on Point Clouds Guided by Optimal Trans­port. In ECCV.


Tang, H.; Liu, Z.; Zhao, S.; Lin, Y.; Lin, J.; Wang, H.; and Han, S. 2020. Search­ing Ef­fi­cient 3D Ar­chi­tec­tures with Spar­se Point­Voxel Con­vo­lu­tion. In ECCV.


Wei, Y.; Zhang, Z.; Rao, Y.; Lu, J.; and Zhou, J. 2021. PV-RAFT: Point-Voxel Cor­re­la­tion Field­s for Scene Flow Es­ti­ma­tion of Point Clouds. In CVPR.


Zhang, Z.; Hua, B.-S.; and Yeung, S.-K. 2019. ShellNet: Ef­fi­cient Point Cloud Convolu­tion­nal Neu­ral Net­works Us­ing Con­cen­tric Shells Sta­tis­tics. In ICCV.

Zhao, H.; Jia, J.; and Koltun, V. 2020. Ex­plor­ing Self­At­ten­tion for Image Rec­og­ni­tion. In CVPR.
