FESTA: Flow Estimation via Spatial-Temporal Attention for Scene Point Clouds

Haiyan Wang¹,²*, Jiahao Pang¹*, Muhammad A. Lodhi¹, Yingli Tian², Dong Tian¹
¹InterDigital  ²The City College of New York
hwang005@citymail.cuny.edu, jiahao.pang@interdigital.com
muhammad.lodhi@interdigital.com, ytian@ccny.cuny.edu, dong.tian@interdigital.com

Abstract

Scene flow depicts the dynamics of a 3D scene, which is critical for various applications such as autonomous driving, robot navigation, AR/VR, etc. Conventionally, scene flow is estimated from dense/regular RGB video frames. With the development of depth-sensing technologies, precise 3D measurements are available via point clouds which have sparked new research in 3D scene flow. Nevertheless, it remains challenging to extract scene flow from point clouds due to the sparsity and irregularity in typical point cloud sampling patterns. One major issue related to irregular sampling is identified as the randomness during point set abstraction/feature extraction—an elementary process in many flow estimation scenarios. A novel Spatial Abstraction with Attention (SA²) layer is accordingly proposed to alleviate the unstable abstraction problem. Moreover, a Temporal Abstraction with Attention (TA²) layer is proposed to rectify attention in temporal domain, leading to benefits with motions scaled in a larger range. Extensive analysis and experiments verified the motivation and significant performance gains of our method, dubbed as Flow Estimation via Spatial-Temporal Attention (FESTA), when compared to several state-of-the-art benchmarks of scene flow estimation.

1. Introduction

Our world is dynamic. To promptly predict and respond to the ever changing surroundings, humans are able to perceive a moving scene and decipher the 3D motion of individual objects. This capability to capture and infer from scene dynamics is also desirable for computer vision applications. For instance, a self-driving car can maneuver its actions upon perceiving the motions in its surroundings [19]; whereas a robot can exploit the scene dynamics to facilitate its localization and mapping process [2]. Moreover, with advances in depth-sensing technologies, especially the LiDAR technologies [7], point cloud data have become a common configuration in such applications.

* Authors contributed equally. Work done while Haiyan Wang was an intern at InterDigital.
two differently down-sampled point clouds, the subsequent grouping and abstraction would lead to two dissimilar sets of local features. Thus, it becomes intractable to estimate the scene flow when comparing the features extracted via FPS.

To resolve this problem, we propose a Spatial Abstraction with Attention (SA\(^2\)) layer which adaptively down-samples and abstracts the input point clouds. Compared to FPS, our SA\(^2\) layer utilizes a trainable Aggregate Pooling (AP) module to generate much more stable down-sampled points, e.g., blue points in Figure 1c. They define attended regions [6] (e.g., green circles in Figure 1d) for subsequent processing.

**Motion coverage:** Similar to many deep matching algorithms for stereo matching and optical flow estimation, it is difficult to have a single DNN that can accurately estimate both large-scale motion and small-scale motion [11, 23]. To tackle this problem, we iterated the network for flow refinement with a proposed Temporal Abstraction with Attention (TA\(^2\)) layer. It shifts the temporal attended regions to the more correspondent areas according to the initial scene flow obtained at the first iteration.

In summary, we adaptively shift the attended regions when seeking abstraction from one point cloud spatially, and when fusing information across two point clouds temporally. We name our proposal Flow Estimation via Spatial-Temporal Attention, or FESTA for short. The main contributions of our work are listed as follows:

(i) We propose the SA\(^2\) layer for stable point cloud abstraction. It shifts the FPS down-sampled points to invariant positions for defining the attended regions, regardless of how the point clouds were sampled from the scene manifold. Effectiveness of the SA\(^2\) layer is verified both theoretically and empirically.

(ii) We propose the TA\(^2\) layer to estimate both small- and large- scale motions. It emphasizes the regions that are more likely to find good matches between the point clouds, regardless of the scale of the motion.

(iii) Our proposed FESTA architecture achieves the state-of-the-art performance for 3D point cloud scene flow estimation on both synthetic and real world benchmarks. Our method significantly outperforms the state-of-the-art methods of scene flow estimation.

**2. Related Work**

Recent studies on scene flow estimation mainly extend methodologies for 2D optical flow estimation to 3D point clouds. We first review the related research on optical flow estimation [27], then turn to deep learning methods for point cloud processing and scene flow estimation.

**Optical flow estimation:** Optical flow estimation and its variant, stereo matching, both look for pixel-wise correspondence given a pair of 2D images. Though conventionally solved with hand-crafted pipelines, recent proposals based on end-to-end DNNs achieve unprecedented performance. Among these methods, FlowNet [4, 18] is the very first trial, which adopts the popular hour-glass structure with skip connections. This basic DNN architecture is remarkably successful for finding correspondence on images [34, 3]. It is even extended to 3D point clouds for scene flow estimation, e.g., FlowNet3D [16] and HPLFlowNet[8]. However, it is difficult to estimate both small- and large-scale motions using one hour-glass architecture. Thus a succeeding work, FlowNet2 [11], stacks independent FlowNet modules to boost the performance, at the price of a larger model. Differently, we resolve the problem with the TA\(^2\) layer, which efficiently reuses part of the network for refinement.

**Deep learning on point clouds:** Point cloud data is usually preprocessed, e.g., voxelized, so as to comply with deep learning frameworks justified for regular images/videos. Emerging techniques for native learning on point clouds relieve this need for format conversion. The seminal work, PointNet [25] directly operates on input points and produces a feature depicting the object geometry. The learned features achieve point permutation-invariance through a pooling operation. PointNet++ [30] applies FPS, followed by nearest-neighbor (NN) grouping and PointNet to abstract an input point cloud. This abstraction step has become a popular elementary unit to digest point clouds. Recent works, such as [35, 22, 15], propose complicated DNN architectures for the abstraction step; while our SA\(^2\) layer is a lightweight module to serve the same purpose. Moreover, these works limit themselves by selecting existing points from the point cloud while we synthesize new points to better represent the underlying geometry.

**Scene flow estimation:** The task of 3D scene flow estimation was first introduced by Vedula et al. [28]. It is conventionally estimated from RGB-D videos [31] or stereo videos [10]. Only with the advent of deep learning, has the 3D scene flow estimation problem directly over point cloud data been enabled [9].

FlowNet3D [16] is the first work directly learning scene flow from 3D point cloud data. It “converts” the FlowNet [4] architecture from the 2D image domain (with the convolutional neural network) to point cloud data (with PointNet). A follow-up work, FlowNet3D++ [26], improves the performance by explicitly supervising the flow vectors with both their magnitudes and orientations; while the recently proposed PointPWC-Net [32] estimates the scene flow in a coarse-to-fine manner by fusing the hierarchical point cloud features. Other notable methods include HPLFlowNet [8] applying the concepts of permutohedral lattice [1] to extract structural information and [20] which is a self-supervised approach. However, most of the efforts apply FPS to down-sample the input point clouds and introduce the unstable abstraction problem as mentioned. In contrast, we propose the SA\(^2\) layer to retrieve invariant down-sampled points,
set which greatly benefit the subsequent matching process.

3. Framework Overview

3.1. Architecture Design

The architecture of our proposal, FESTA, is shown in Figure 2, which follows the backbone of FlowNet3D [16] and FlowNet [4] with an hour-glass structure. Each feature produced by network layers consists of a representative point accompanied by a local descriptor, e.g., the Spatial Abstraction with Attention (SA²) layer generates \( n_1/8 \) such features for the first point cloud. Given two input point clouds, they are respectively consumed by shared SA² layers to extract two sets of features, which we call the spatial features. Then a proposed Temporal Abstraction with Attention (TA²) layer serves as a coupling module to fuse the spatial features with the first point cloud serving as the reference. Its output is another set of features that we called the temporal feature. Different from the spatial features, the temporal features fuse information of both point clouds, from which the 3D scene flow can be extracted. After that, several Set Abstraction layers and Set Up-Conv layers from FlowNet3D [16] are appended to digest the temporal features, which complete the hour-glass structure. The outputs of the last Set Up-Conv layer are a set of point-wise features, associated with each point in the first point cloud. To extract point-wise scene flow, we simply apply shared MLP layers to convert each point-wise feature to a scene flow vector.

Inspired by [12], we also estimate a binary mask indicating the existence of the scene flow vector for each point in the first point cloud. In practice, the scene flow vectors may not be available due to occlusion and motion out of the field of view, etc. The indication of their existence may serve as side information to help subsequent tasks. Similar to the computation of 3D scene flow, dedicated MLP layers are applied to convert the point-wise features to the existence mask.

To enhance the scene flow estimation accuracy, especially to tackle motion of all ranges, we partially re-iterate our network with a feedback connection. Though it is possible to run TA² many iterations, we found that running it twice achieves a good trade-off between computational cost and estimation accuracy. Note that similar to [16], our FESTA architecture can be easily adapted to take additional attributes (e.g., RGB colors) as inputs. Please refer to the supplementary material for more details of our architecture.

3.2. Loss Function Design

To effectively train the proposed FESTA in an end-to-end fashion, we evaluate both iteration outputs against the ground truth flow. For each iteration, we first compute an \( \ell_2 \) loss between the ground-truth scene flow and the estimated one [16]. This \( \ell_2 \) loss is denoted by \( L_F^{(1)} \), with \( i \) being the iteration index. Then the existence mask estimation is cast as a point-wise binary classification problem [12], and a cross-entropy loss can be calculated against the ground-truth existence mask, denoted by \( L_M^{(1)} \). The loss of the \( i \)-th iteration is finally given by:

\[
L^{(i)} = \mu L_F^{(i)} + (1 - \mu) L_M^{(i)}.
\] (1)

Our total loss for end-to-end training aggregates losses of both iterations, i.e.,

\[
L_{tot} = (1 - \lambda)L^{(1)} + \lambda L^{(2)}.
\] (2)

Note that in (1) and (2), the hyper-parameters \( \mu, \lambda \in [0, 1] \). Empirically, we set \( \mu = 0.8 \) and \( \lambda = 0.7 \).

4. Spatial-Temporal Attention

4.1. Spatial Abstraction with Attention

Design of the SA² layer: Key steps of our proposed Spatial Abstraction with Attention (SA²) layer are illustrated
in Figure 1. Farthest Point Sampling (FPS) followed by the Nearest Neighbor (NN) grouping (Figure 1a, Figure 1b) are inherited from PointNet++ [26] to divide the point cloud into groups as initial steps. However, as mentioned in Section 1, pure FPS-based abstraction produces unstable down-sampled points that would tamper the scene flow estimation.

Herein, the design of the abstraction is motivated to reflect the intrinsic geometry of the manifold $\mathcal{M}$, that is invariant to the randomness in the sampling pattern. In this work, we approach this goal with a proposed Aggregate Pooling (AP) module. Following FPS grouping, a point set is down-sampled using a synthesized point (Figure 1c). Then each newly down-sampled point defines its own attended region via another NN grouping step, leading to a new grouping scheme more suitable for a subsequent point-wise task (Figure 1d), i.e., scene flow estimation in our case. Similar to PointNet++ [26], finally we feed the new groups of points to a shared PointNet to extract their local descriptors. The descriptors and the associated down-sampled points constitute the output of the SA$^2$ layer, i.e., the spatial features. For example, see the \( \frac{64}{8} \times 67 \) matrix produced by the SA$^2$ layer in Figure 2.

**Aggregate Pooling:** The proposed AP module consumes a group of $k$ points and generates a synthesized point to represent the whole group. As shown in Figure 3, it consists of a PointNet [25] and a point aggregation step. The PointNet computes $k$ point-level descriptors with MLPs and a group-level descriptor with a max-pooling operator. This PointNet is shared among all groups of points in the point cloud. The point aggregation step then computes the weighted average of all the points in the group to synthesize a representative point.

Specifically, the point aggregation step measures the representativeness of a point (say, the $i$-th point) in the group with the similarity between its point-level descriptor (denoted by $f_s$) and the group-level descriptor (denoted by $f_g$). Among different ways to measure the vector similarity (e.g., Euclidean distance, correlation coefficient), we choose the dot-product metric like [29] for its simplicity. The obtained affinity values are then passed to a softmax function for normalization, resulting in a set of weights summed up to 1. Mathematically, the weight $w_i$ for the $i$-th point is

\[
    w_i = \exp \left( f_s^T f_g \right) \cdot \left[ \sum_{j=1}^{k} \exp \left( f_s^T f_j \right) \right]^{-1}. \tag{3}
\]

Suppose the points in the group are $s_i = (x_i, y_i, z_i), 1 \leq i \leq k$, then the synthesized point is simply $\sum_{i=1}^{k} w_i \cdot s_i$.

**Analysis:** Now we attempt to understand the mechanism of the SA$^2$ layer to generate more stable points. With the FPS and the grouping steps (Figure 1a and Figure 1b), we have two lists of point sets from the input point cloud pair. We first focus on a pair of point sets among the two lists that are sampled over the same Riemannian manifold patch $\mathcal{M}$ through the same sampling probability distribution $p(s)$. Each point set is to be processed by the AP module.

Given sufficient number of points in both point sets characterizing the geometry $\mathcal{M}$, their group-level descriptors should be similar [25], denoted by $f_g$. In particular, if increasingly many 3D points are sampled from $\mathcal{M}$ according to $p$, then by definition, the synthesized points from both groups converge to the following integration on $\mathcal{M}$ [14]:

\[
    s' = \frac{1}{\alpha} \int_{\mathcal{M}} w \left( f(s)^T f_g \right) p(s) \cdot s \, ds, \tag{4}
\]

where $f(s)$ is the point-level descriptor of the point $s$, function $w(\cdot)$ converts the dot-product measure $f(s)^T f_g$ to weight as is done by (3), and $\alpha = \int_{\mathcal{M}} w \left( f(s)^T f_g \right) p(s) \, ds$ is a normalization factor. Please refer to the supplementary material for more detailed analysis.

Since the AP module converges to a fixed location $s'$ over $\mathcal{M}$, the SA$^2$ layer is expected to converge over a point cloud scene. Empirical evidence is to be provided in Section 5.1 through a segmentation experiment.

Lastly, as the weight $w \left( f(s)^T f_g \right)$ is computed by a learnable network, the SA$^2$ layer is adaptable to down-stream tasks by generating novel, task-aware down-sampled points.

**4.2. Temporal Abstraction with Attention**

**Mechanisms of the TA$^2$ layer:** The Temporal Abstraction with Attention (TA$^2$) layer aims to aggregate the spatial features of both point clouds given an initial scene flow. During the first iteration without an initial scene flow presented, it behaves the same as the Flow Embedding layer in FlowNet3D [16]. Specifically, for each point (say, $A$) in the first down-sampled point cloud, we first perform a NN grouping step from the second point cloud, which forms a group of points (centered at $A$) from the second point cloud, as shown in the left of Figure 4. Then the grouped points, the point $A$ and its associated descriptor, are sent to a subsequent PointNet to extract another local descriptor. More details of this extraction step can be found at [16]. The down-sampled point cloud and the new descriptors form the temporal features, e.g., the $\frac{64}{8} \times 131$ matrix produced by the TA$^2$ layer in Figure 2.
During the second iteration, we reuse the spatial features generated by the SA² layers at the first iteration and feed them to the TA² layer (Figure 2). However, now with an initial scene flow corresponding to the first down-sampled point cloud available, we translate the search regions according to each of the scene flow vectors. Specifically, suppose the coordinate of $A$ is $(x_A, y_A, z_A)$ and its initial scene flow vector is $(u_A, v_A, w_A)$, then the NN grouping is performed centered at $(x_A + u_A, y_A + v_A, z_A + w_A)$, see the right of Figure 4. Note that to obtain the initial scene flow for the first down-sampled point cloud, it requires an extra interpolation step, i.e., the Flow Interpolation module in Figure 2. We implement it as a simple deterministic module without trainable parameters. To estimate the scene flow vector at a certain point, it computes the average scene flow vector in the neighborhood of that point. Please refer to the supplementary material for more details.

**Analysis:** Intuitively, the NN grouping step search for all points in the second point cloud that “appear close” to $A$; its search range defines an attended region (the blue circle in Figure 4). By only estimating the scene flow in one pass, it requires to choose a universal search radius for all ranges of motion. However, when the attended region (or equivalently, the search radius $r_1$ in Figure 4) is too small, it fails to capture large-scale motion; while for a large attended region or a large $r_1$, it includes too many candidates from the second point cloud and harms the granularity of the estimation (especially for small-scale motion). This problem generally exists for not only scene flow estimation but also related problems such as stereo matching [23] and optical flow estimation [11].

By introducing a second iteration, our TA² layer accordingly shifts the attended regions to confident areas that are more likely to observe good matches from the second point cloud. Consequently, for the first iteration, it is more critical to identify a “correct” direction than a “correct” result. It is reflected in the selection of hyper-parameter $\lambda = 0.7$. Moreover, with a rough knowledge about how the second point cloud moves, the attended region at the second iteration (or radius $r_2$ in Figure 4) can be further reduced to search for more refined matching candidates.

### 5. Experimentation

This section first verifies how the SA² layer serves as a stable point cloud abstraction unit. Then the proposed FESTA architecture is evaluated for 3D scene flow estimation. Finally, we inspect how the key components contribute to the FESTA framework by an ablation study.

#### 5.1. Abstraction with the SA² Layer

The proposed SA² layer generally provides an alternative abstraction than FPS-like methods. It is desired to explicitly study its stability as pointed out in Section 4.1. In this test, we design a dedicated object segmentation process, as this per-point task has minimum additional procedure than abstraction, if compared to FESTA framework. Note that the segmentation serves as a test bed to verify the stability of SA², rather than to claim state-of-the-art segmentation.

**Set-up:** We build up a scene point cloud dataset using object point clouds from ModelNet40 [33]. This scene point cloud dataset contains $10^4$ scenes; each scene contains 3 to 6 objects packed within a sphere-shaped container of radius 3. Moreover, all objects in a scene are normalized within a sphere of radius $r$, and are situated with a distance of at least 2 between their object centers. Evidently, by enlarging the objects with a bigger radius $r$, they are more likely to collide with one another, making it more challenging to distinguish and segment the objects. We prepare 4 versions of the scene point cloud dataset with different object radii $r$ ranging from 1 to 1.8. Our object segmentation network is built upon the PointNet++ [26] by replacing the FPS groupings with our proposed SA² layers.

**Abstraction stability:** We first evaluate the stability of our down-sampled point clouds using the above segmentation framework with dataset radius $r = 1.2$. Given a scene in our multi-object dataset, we randomly pick $n \in [256, 2048]$ points from it for 100 different times, resulting in 100 input
point clouds representing the same 3D scene. Then we feed these point clouds as inputs to both FPS-based and SA²-based segmentation network to obtain down-sampled point sets containing only 64 points. For a stable down-sampling procedure, the down-sampled results from the 100 point clouds should be similar to one another. To evaluate the similarity, we compute the Chamfer Distances (CD) \[ r \] between any two down-sampled point clouds, then take an average to characterize the stability. A smaller average CD means more stable down-sampling. Further averaging over 30 scenes randomly selected from our dataset is performed. Finally, as shown in Figure 5, SA² always produces more stable down-sampled results than FPS. Especially, for \( n > 1000 \), our approach even reduces the average CD of FPS by about 50%, which confirms the superior stability of the proposed SA² layer. The plot for SA² also verifies the analysis in Section 4.1, which indicated that as the sampling becomes denser the down-sampled points become more stable.

**Evaluation:** Having tested the SA² layer, we turn to understand how its stable abstraction benefits the segmentation performance. Based on FPS and the SA² layer, two segmentation networks are trained on all the 4 versions of the dataset (object radii \( r \) ranging from 1 to 1.8) with the cross-entropy loss. Table 1 compares the performance of SA² based segmentation with FPS-based segmentation, i.e., PointNet++, showing that SA² always provides higher segmentation accuracies. As segmentation difficulty increases, our approach out-performs PointNet++ by a larger margin, e.g., our accuracy is 26% higher than PointNet++ when \( r = 1.8 \).

We demonstrate the down-sampled points of scene point clouds in Figure 6, where the grey points depict the input point cloud; while the red and the blue points are those sampled by FPS and SA², respectively. Thanks to the stable abstraction as verified earlier, the SA² layer consistently generates points belonging to distinctive objects and exhibits clearer separation between objects, which is highly preferred for segmentation.

5.2. Scene Flow Estimation with FESTA

Combining the SA² and the TA² layers, we evaluate the proposed FESTA architecture for scene flow estimation.

**Datasets:** Our experiments are conducted on two popular datasets, the FlyingThings3D [18] and the KITTI Scene Flow [7] (referred to as KITTI) datasets. Both of them are originally designed for matching tasks in the image domain (e.g., stereo matching). Recently, Liu et al. [16] converted them for scene flow estimation from 3D point clouds. The FlyingThings3D dataset is a synthetic dataset with 20,000 and 2,000 point cloud pairs for training and testing, respectively. In addition to the point cloud geometry, the RGB colors and the binary existence masks are also available. Different from the FlyingThings3D, KITTI is a real dataset collected by LiDAR sensors, and contains incomplete objects. The KITTI dataset has 150 point cloud pairs with available ground-truth scene flow. Similar to [16, 8] and others, we only use the geometry (point coordinates) when computing the 3D scene flow.

**Benchmarks and evaluation metrics:** We compare our FESTA with the following methods, listed in chronological order: FlowNet3D [16], HPLFlowNet [8], PointPWC-Net [32], MeteorNet [16], FlowNet3D++ [30], and a self-supervised method, Just Go with the Flow [20]. The scene flow quality is first evaluated with End-Point-Error (EPE), which calculates the mean Euclidean distance between the ground-truth flow and the prediction. We also adopt two additional metrics from [8], Acc Strict and Acc Relax.¹ Both Acc Strict and Acc Relax aim to measure estimation accuracy, but with different thresholds. Acc Strict measures the percentage of points satisfying EPE < 0.05m or relative error < 5%; while Acc Relax measures the percentage of points with EPE < 0.1m or relative error < 10%.

**Implementation details:** The proposed FESTA is trained with two configurations over FlyingThings3D dataset following the FlowNet3D [16], with geometry-only and with additional RGB attributes. The two configurations are both trained with the Adam optimizer [13] for 500 epochs, with a batch size of 32 and a learning rate of 0.001. The size of the input point clouds are all set to 2048. All experiments are performed in the PyTorch [24] framework. For the geometry-only configuration, the inference is performed on both FlyingThings3D and KITTI. In other words, the model is never tuned over KITTI, similarly done in [16, 8] and others. With RGB attributes available, the inference is conducted on FlyingThings3D. Quantitative and qualitative evaluations are performed.

**Quantitative evaluation:** Quantitative results are re-

---

¹They are called Acc3D Strict and Acc3D Relax in [8]. We remove “3D” since we do not need to distinguish with 2D cases.
Table 2: Quantitative evaluation on the FlyingThings3D and the KITTI datasets.

<table>
<thead>
<tr>
<th>Methods</th>
<th>FlyingThings3D, geo-RGB</th>
<th>FlyingThings3D, geo-only</th>
<th>KITTI, geo-only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EPE (m)</td>
<td>Acc S. (%)</td>
<td>Acc R. (%)</td>
</tr>
<tr>
<td>FlowNet3D [16]</td>
<td>0.1694</td>
<td>25.37</td>
<td>57.85</td>
</tr>
<tr>
<td>HPLFlowNet [8]</td>
<td>0.1318</td>
<td>32.78</td>
<td>63.22</td>
</tr>
<tr>
<td>PointPWC-Net [32]</td>
<td>0.1205</td>
<td>39.45</td>
<td>67.81</td>
</tr>
<tr>
<td>MeteorNet [17]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FlowNet3D++ [30]</td>
<td>0.1369</td>
<td>30.33</td>
<td>63.43</td>
</tr>
<tr>
<td>Just Go w/ Flow [20]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FESTA (Ours)</td>
<td><strong>0.1113</strong></td>
<td><strong>43.12</strong></td>
<td><strong>74.42</strong></td>
</tr>
</tbody>
</table>

Reported in Table 2, where the proposed FESTA consistently outperforms the competing methods with significant gains. For example, regarding to Acc Strict values for geometry only, our FESTA improves over the state-of-the-art method, PointPWC-Net, by 5.3% for FlyingThings3D or 8.9% for KITTI; and improves over our backbone, FlowNet3D, by 15.8% for FlyingThings3D or 26.3% for KITTI. Encouragingly, when inspecting configurations with and without RGB over FlyingThings3D, it is noticed that in most cases, with geometry alone, our FESTA surpasses the competitors even when they take extra RGB attributes. For instance, our FESTA (geometry only) achieves an EPE of 0.1253, lower than that of FlowNet++ (geometry+RGB) which is 0.1369.

We compare our model size and run time to representative methods and report in Table 3, where the run time is evaluated on a Nvidia GTX 1080 Ti GPU with 11 GB memory. We confirm that our superior performance is achieved by a model of size 16.1 MB, similar to FlowNet3D which is 14.9 MB. It is much smaller than other competing methods, PointPWC-Net and HPLFlowNet. Moreover, by removing the TA², we take similar run time as FlowNet3D, but still greatly reduce its EPE from 0.1705 (Table 2) to 0.1381 (to be seen in ablation study, Table 4).

**Qualitative evaluation:** Visualization of the scene flow obtained by FESTA are shown in Figure 7 for FlyingThings3D and in Figure 8 for KITTI. Selected parts are zoomed in for a better illustration. In each example, red points and green points represent the first and the second point cloud frames, respectively. Blue points represent the warped point cloud, generated by translating each point in the first point cloud according to the estimated scene flow vector. With more accurate scene flow vectors, the warped point cloud gets more overlapped with the second point cloud. It can be seen that for all cases in Figure 7, 8, our predicted scene flow produces warped point clouds that are highly overlapped with the second point cloud. It affirms the effectiveness of our proposed FESTA on scene flow estimation. Be reminded that our network has never observed any data from the KITTI dataset; however, it still successfully generalizes to KITTI and captures the dynamics solely based on the point coordinates.

### 5.3. Ablation Study

The ablation study is performed with geometry only configuration. We investigate the benefits of individual components in our FESTA architecture. Specifically, we consider the following three variants:

1. Replace the SA² layers as the simple FPS grouping followed by feature extraction in PointNet++ [26] (or...
Figure 8: Scene flow estimation on the KITTI dataset between 1st PC (in red), 2nd PC (in green). The results of our proposed FEST A architecture is shown with warped PC (in blue) – 1st PC warped by the scene flow.

Table 4: Evaluation of different variants of FESTA.

<table>
<thead>
<tr>
<th>Data</th>
<th>SA²</th>
<th>TA²</th>
<th>Mask</th>
<th>EPE (m)</th>
<th>Acc S (%)</th>
<th>Acc R (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fly</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>0.1402</td>
<td>33.15</td>
<td>66.14</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>0.1381</td>
<td>34.70</td>
<td>67.36</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>0.1289</td>
<td>37.91</td>
<td>69.05</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.1253</td>
<td>39.52</td>
<td>71.24</td>
</tr>
<tr>
<td>KI</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>0.1163</td>
<td>30.10</td>
<td>70.85</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>0.1027</td>
<td>41.07</td>
<td>80.04</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>0.0955</td>
<td>42.74</td>
<td>80.15</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.0936</td>
<td>44.85</td>
<td>83.35</td>
</tr>
</tbody>
</table>

FlowNet3D [16]);
(ii) Replace the TA² layer as the Flow Embedding layer in FlowNet3D, i.e., the second iteration is removed; and
(iii) Remove the outputting of existence mask, i.e., the network is trained only over 3D scene flow.

Performance of the mentioned variants of FESTA are reported in Table 4. The best- and the worst- performing metrics are highlighted with bold faces and with underlines, respectively. We see that both the SA² and the TA² layers bring substantial gain to our model; while generally, the SA² layer is slightly more effective than the TA² layer. Moreover, by jointly estimating an additional existence mask, our scene flow quality further improves. That is because the ground-truth existence mask provides extra clues about the motion [12], which supervises the network to capture the dynamics more precisely.

Additionally, we further examine how the SA² and the TA² layers benefit the estimation of motion at different scales. Specifically, we classify 3D points in the FlyingThings3D test set according to their ground-truth flow vector magnitudes. For each bin of scene flow magnitude, we compute an average of relative error achieved by our FESTA, and count the bin size. In this way, we plot a relative error curve (in green) of FESTA on different scene flow magnitudes, as shown in Figure 9. We similarly plot the curves for the variants without SA² and TA² in blue and red, respectively.

By comparing the red and the green curves, we see that the TA² layer greatly improves the performance of large-scale motion, which is expected because the TA² directly shifts its attended region according to an initial scene flow. Differently, the SA² layer benefits scene flow with smaller magnitudes. That is because the SA² layer is able to adjust its attended region gently—the convex hull of an input point group (Section 4.1), which enhances the granularity of the estimation and benefits mainly small-scale motion.

6. Conclusion

We propose a new spatial-temporal attention mechanism to estimate 3D scene flow from point clouds. The effectiveness of our proposed Flow Estimation via Spatial-Temporal Attention (FESTA) has been proven by our state-of-the-art performance of extensive experiments. Essentially, our spatial-temporal attention mechanism successfully rectifies the region of interest (RoI) based on the feedback from earlier trials. Its rationale resembles existing literature utilizing the attention mechanism, such as [6] for image recognition and [36] for sentence modeling. For future research, we plan to investigate the potentials of the SA² and TA² layers for different point cloud processing tasks, such as classification, registration, and compression.
References


