

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

Haiyan Wang<sup>1,2\*</sup>    Jiahao Pang<sup>1\*</sup>    Muhammad A. Lodhi<sup>1</sup>    Yingli Tian<sup>2</sup>    Dong Tian<sup>1</sup>  
<sup>1</sup>InterDigital    <sup>2</sup>The 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 ( $SA^2$ ) layer is accordingly proposed to alleviate the unstable abstraction problem. Moreover, a Temporal Abstraction with Attention ( $TA^2$ ) 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.

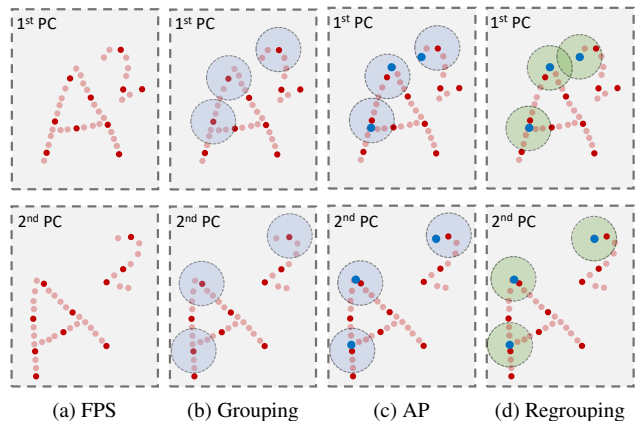


Figure 1: Given two consecutive point clouds, down-sampled points produced by (a) Farthest Point Sampling (FPS, in dark red) are different which make them intractable in scene flow estimation. However, by appending our (c) Aggregate Pooling (AP), stable corresponding points (in blue) are synthesized for scene flow estimation.

To describe the motion of individual points in the 3D world, scene flow extends 2D optical flow to the 3D vector field representing the 3D scene dynamics [28]. Hence, just like 2D optical flow needs to be estimated from video frames comprising images [31, 10], 3D scene flow needs to be inferred from point cloud data [9]. However, it is non-trivial to accurately estimate scene flow from point clouds.

**Unstable abstraction:** Pioneered by PointNet [25] and its extension, PointNet++ [26], deep neural networks (DNNs) have recently been enabled to directly consume 3D point clouds for various vision tasks. As shown in Figure 1a and Figure 1b, the grouping based on the Farthest Point Sampling (FPS) is widely utilized during the feature extraction process. It is treated as a basic point set abstraction unit for segmentation as well as scene flow estimation, e.g., FlowNet3D [16] and MeteorNet [17]. The naive FPS is simple and computationally affordable, but problematic. Given two object point clouds, both representing the same manifold, FPS would likely down-sample them differently [21] (see Figure 1a). This inconsistency due to randomness in naive FPS is undesired for vision and machine tasks. With

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<sup>2</sup>) layer which adaptively down-samples and abstracts the input point clouds. Compared to FPS, our SA<sup>2</sup> 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 iterate the network for flow refinement with a proposed *Temporal Abstraction with Attention* (TA<sup>2</sup>) 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<sup>2</sup> 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<sup>2</sup> layer is verified both theoretically and empirically.
- (ii) We propose the TA<sup>2</sup> 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<sup>2</sup> 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<sup>2</sup> 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<sup>2</sup> layer to retrieve invariant down-sampled points,

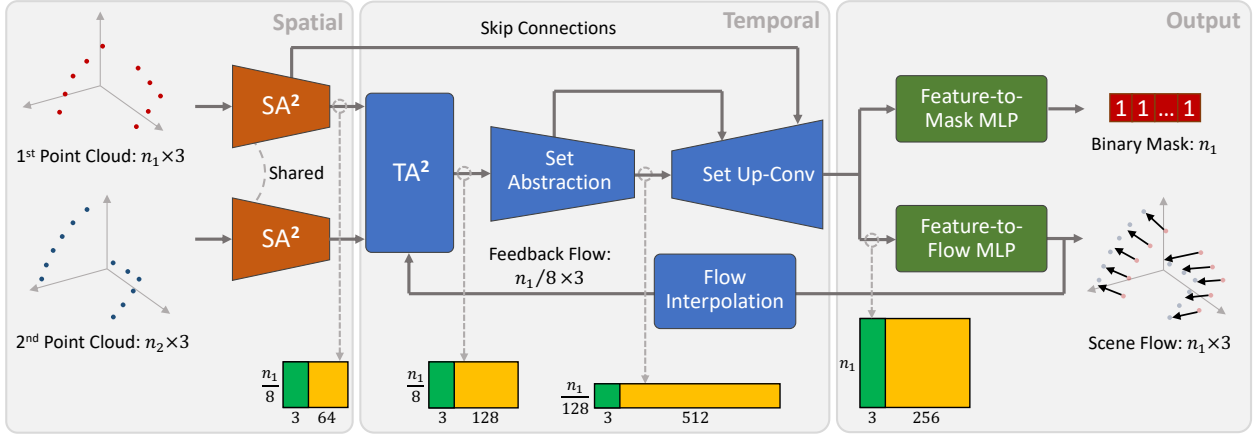


Figure 2: Our proposed FESTA architecture. On top of the FlowNet/FlowNet3D backbone, we specially incorporate the spatial-temporal mechanism with the proposed  $SA^2$  and  $TA^2$  layers.

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 ( $SA^2$ ) layer generates  $n_1/8$  such features for the first point cloud. Given two input point clouds, they are respectively consumed by shared  $SA^2$  layers to extract two sets of features, which we call the *spatial features*. Then a proposed Temporal Abstraction with Attention ( $TA^2$ ) 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  $TA^2$  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  $L_F^{(i)}$ , 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  $L_M^{(i)}$ . The loss of the  $i$ -th iteration is finally given by:

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

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

$$L_{\text{tot}} = (1 - \lambda) L^{(1)} + \lambda L^{(2)}. \quad (2)$$

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  $SA^2$  layer:** Key steps of our proposed Spatial Abstraction with Attention ( $SA^2$ ) 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<sup>2</sup> layer, *i.e.*, the spatial features. For example, see the  $\frac{n_1}{8} \times 67$  matrix produced by the SA<sup>2</sup> 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  $\mathbf{f}_i$ ) and the group-level descriptor (denoted by  $\mathbf{f}_g$ ). 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

$$w_i = \exp(\mathbf{f}_i^T \mathbf{f}_g) \cdot \left[ \sum_{j=1}^k \exp(\mathbf{f}_j^T \mathbf{f}_g) \right]^{-1}. \quad (3)$$

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

**Analysis:** Now we attempt to understand the mechanism of the SA<sup>2</sup> 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

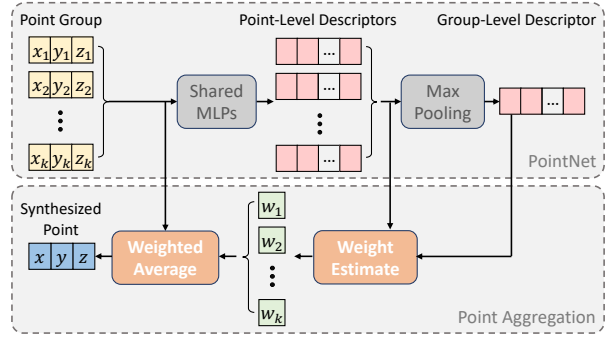


Figure 3: Block diagram of the Aggregate Pooling (AP) module.

$\mathcal{M}$  through the same sampling probability distribution  $p(\mathbf{s})$ . 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  $\mathbf{f}_g$ . 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]:

$$\mathbf{s}' = \frac{1}{\alpha} \int_{\mathcal{M}} w(\mathbf{f}(\mathbf{s})^T \mathbf{f}_g) p(\mathbf{s}) \cdot \mathbf{s} \, ds, \quad (4)$$

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

Since the AP module converges to a *fixed* location  $\mathbf{s}'$  over  $\mathcal{M}$ , the SA<sup>2</sup> 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  $w(\mathbf{f}(\mathbf{s})^T \mathbf{f}_g)$  is computed by a learnable network, the SA<sup>2</sup> 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<sup>2</sup> layer:** The Temporal Abstraction with Attention (TA<sup>2</sup>) 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  $\frac{n_1}{8} \times 131$  matrix produced by the TA<sup>2</sup> layer in Figure 2.

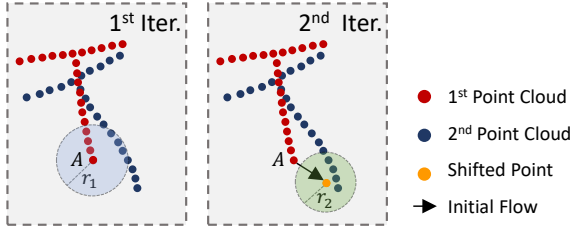


Figure 4: Two iterations of the  $TA^2$  layer. The blue circle in the left one demonstrates the attended region for the 1<sup>st</sup> iteration, which is inaccurate for the points correspondence. The green circle in the right one drags the attended region to the correspondent area by the initial flow estimated in the 1<sup>st</sup> iteration.

During the second iteration, we reuse the spatial features generated by the  $SA^2$  layers at the first iteration and feed them to the  $TA^2$  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  $TA^2$  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

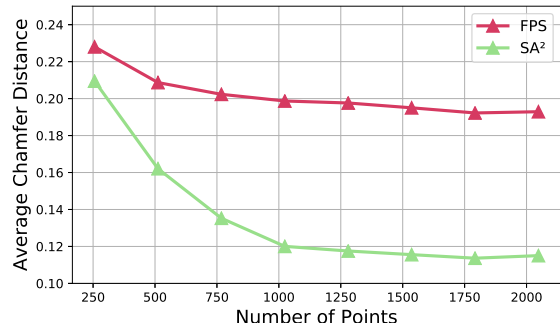


Figure 5: Compared to FPS, our proposal generates more stable down-sampled point clouds.

more refined matching candidates.

## 5. Experimentation

This section first verifies how the  $SA^2$  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 $SA^2$ Layer

The proposed  $SA^2$  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  $SA^2$ , 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  $SA^2$  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

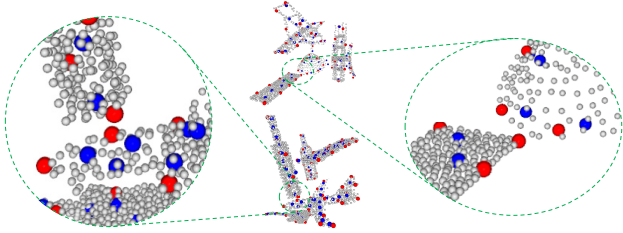


Figure 6: Down-sampled points of FPS (red) and our  $SA^2$  layer (blue) for the object segmentation task.

Table 1: Object segmentation accuracy (%).

Methods	Object radius $r$			
	1.0	1.2	1.5	1.8
PointNet++ (FPS)	92.56	88.74	65.87	43.07
Ours ( $SA^2$ )	<b>93.21</b>	<b>90.48</b>	<b>80.10</b>	<b>69.18</b>

point clouds representing the same 3D scene. Then we feed these point clouds as inputs to both FPS-based and  $SA^2$ -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) [5] 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,  $SA^2$  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  $SA^2$  layer. The plot for  $SA^2$  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  $SA^2$  layer, we turn to understand how its stable abstraction benefits the segmentation performance. Based on FPS and the  $SA^2$  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  $SA^2$  based segmentation with FPS-based segmentation, *i.e.*, PointNet++, showing that  $SA^2$  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  $SA^2$ , respectively. Thanks to the stable abstraction as verified earlier, the  $SA^2$  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  $SA^2$  and the  $TA^2$  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], PointPWCNet [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*.<sup>1</sup> 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 FlyingThing3D. Quantitative and qualitative evaluations are performed.

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

<sup>1</sup>They are called *Acc3D Strict* and *Acc3D Relax* in [8]. We remove “3D” since we do not need to distinguish with 2D cases.

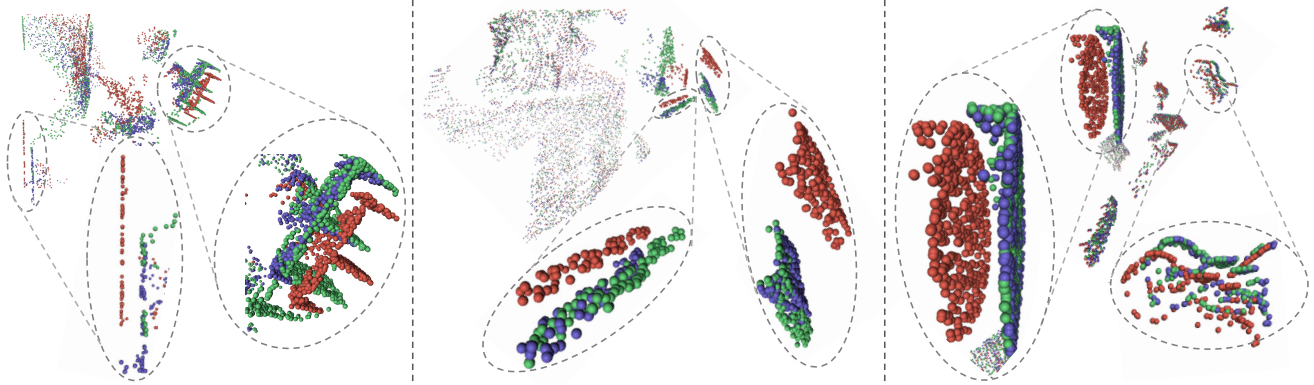


Figure 7: Scene flow estimation on the FlyingThings3D dataset between 1<sup>st</sup> PC (in red), 2<sup>nd</sup> PC (in green). The results of our proposed FESTA architecture is shown with warped PC (in blue) – 1<sup>st</sup> PC warped by the scene flow.

Table 2: Quantitative evaluation on the FlyingThings3D and the KITTI datasets.

Methods	FlyingThings3D, geo.+RGB			FlyingThings3D, geo.-only			KITTI, geo.-only		
	EPE (m)	Acc S. (%)	Acc R. (%)	EPE (m)	Acc S. (%)	Acc R. (%)	EPE (m)	Acc S. (%)	Acc R. (%)
FlowNet3D [16]	0.1694	25.37	57.85	0.1705	23.71	56.05	0.1220	18.53	57.03
HPLFlowNet [8]	0.1318	32.78	63.22	0.1453	29.46	61.91	0.1190	30.83	64.76
PointPWC-Net [32]	0.1205	39.45	67.81	0.1310	34.22	65.78	0.1094	35.98	73.84
MeteorNet [17]	-	-	-	0.2090	-	52.12	0.2510	-	-
FlowNet3D++ [30]	0.1369	30.33	63.43	0.1553	28.50	60.39	0.2530	-	-
Just Go w/ Flow [20]	-	-	-	-	-	-	0.1220	25.37	57.85
FESTA (Ours)	<b>0.1113</b>	<b>43.12</b>	<b>74.42</b>	<b>0.1253</b>	<b>39.52</b>	<b>71.24</b>	<b>0.0936</b>	<b>44.85</b>	<b>83.35</b>

ported 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 TA<sup>2</sup>, 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

Table 3: Evaluation of model size and run time. F - FlowNet3D [16]; H - HPLFlowNet [8]; P - PointPWC-Net [32].

Metrics	F.	H.	P.	FESTA w/o TA <sup>2</sup>	FESTA (Ours)
Size (MB)	14.9	231.8	30.1	16.1	16.1
Time (ms)	34.9	93.1	38.5	35.2	67.8

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:

- (i) Replace the SA<sup>2</sup> layers as the simple FPS grouping followed by feature extraction in PointNet++ [26] (or

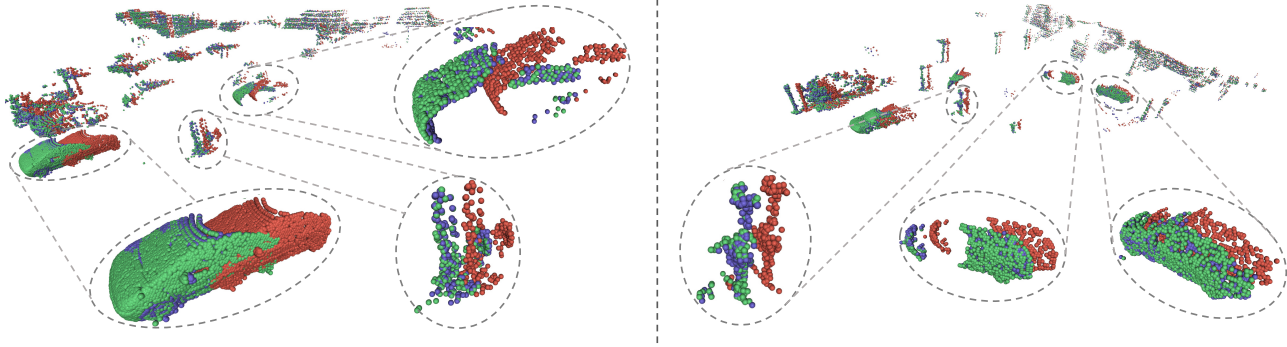


Figure 8: Scene flow estimation on the KITTI dataset between 1<sup>st</sup> PC (in red), 2<sup>nd</sup> PC (in green). The results of our proposed FESTA architecture is shown with warped PC (in blue) – 1<sup>st</sup> PC warped by the scene flow.

Table 4: Evaluation of different variants of FESTA.

Data.	SA <sup>2</sup>	TA <sup>2</sup>	Mask	EPE (m)	Acc S. (%)	Acc R. (%)
Fly.	×	✓	✓	<u>0.1402</u>	33.15	<u>66.14</u>
	✓	×	✓	0.1381	34.70	67.36
	✓	✓	×	0.1289	37.91	69.05
	✓	✓	✓	<b>0.1253</b>	<b>39.52</b>	<b>71.24</b>
KI.	×	✓	✓	<u>0.1163</u>	30.10	<u>70.85</u>
	✓	×	✓	0.1027	41.07	80.04
	✓	✓	×	0.0955	42.74	80.15
	✓	✓	✓	<b>0.0936</b>	<b>44.85</b>	<b>83.35</b>

FlowNet3D [16]);

- (ii) Replace the TA<sup>2</sup> 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 SA<sup>2</sup> and the TA<sup>2</sup> layers bring substantial gain to our model; while generally, the SA<sup>2</sup> layer is slightly more effective than the TA<sup>2</sup> 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 SA<sup>2</sup> and the TA<sup>2</sup> 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 SA<sup>2</sup> and TA<sup>2</sup> in blue and red, respectively.

By comparing the red and the green curves, we see that the TA<sup>2</sup> layer greatly improves the performance of large-

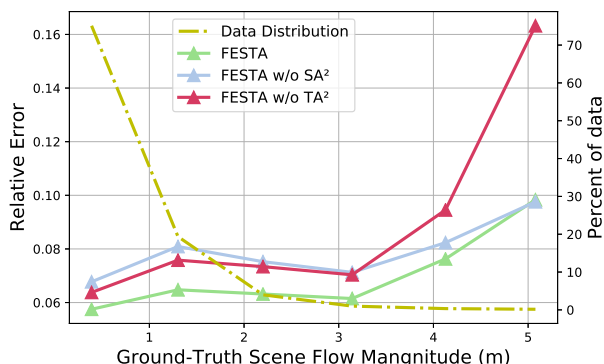


Figure 9: For ground-truth motion with different magnitudes, the variants of FESTA perform differently.

scale motion, which is expected because the TA<sup>2</sup> directly shifts its attended region according to an initial scene flow. Differently, the SA<sup>2</sup> layer benefits scene flow with smaller magnitudes. That is because the SA<sup>2</sup> 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 SA<sup>2</sup> and TA<sup>2</sup> layers for different point cloud processing tasks, such as classification, registration, and compression.



## References

- [1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010. [2](#)
- [2] Pablo F Alcantarilla, José J Yebes, Javier Almazán, and Luis M Bergasa. On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 1290–1297, 2012. [1](#)
- [3] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5410–5418, 2018. [2](#)
- [4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Int. Conf. Comput. Vis.*, pages 2758–2766, 2015. [2](#), [3](#)
- [5] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 605–613, 2017. [6](#)
- [6] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4438–4446, 2017. [2](#), [8](#)
- [7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3354–3361, 2012. [1](#), [6](#)
- [8] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical permutohedral lattice FlowNet for scene flow estimation on large-scale point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3254–3263, 2019. [2](#), [6](#), [7](#)
- [9] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. [1](#), [2](#)
- [10] Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D flow: Dense 3-D motion estimation using color and depth. In *IEEE International Conference on Robotics and Automation*, pages 2276–2282, 2013. [1](#), [2](#)
- [11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2462–2470, 2017. [2](#), [5](#)
- [12] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Eur. Conf. Comput. Vis.*, pages 614–630, 2018. [3](#), [8](#)
- [13] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, 2015. [6](#)
- [14] Erwin Kreyszig. *Introductory Functional Analysis with Applications*, volume 1. Wiley New York, 1978. [4](#)
- [15] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable point cloud sampling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7578–7588, 2020. [2](#)
- [16] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 529–537, 2019. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#)
- [17] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. MeteoNet: Deep learning on dynamic 3D point cloud sequences. In *Int. Conf. Comput. Vis.*, pages 9246–9255, 2019. [1](#), [7](#)
- [18] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4040–4048, 2016. [2](#), [6](#)
- [19] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3061–3070, 2015. [1](#)
- [20] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11177–11185, 2020. [2](#), [6](#), [7](#)
- [21] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. [1](#)
- [22] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12956–12964, 2020. [2](#)
- [23] Jiahao Pang, Wenxiu Sun, Chengxi Yang, Jimmy Ren, Ruichao Xiao, Jin Zeng, and Liang Lin. Zoom and learn: Generalizing deep stereo matching to novel domains. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2070–2079, 2018. [2](#), [5](#)
- [24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems Workshop*, 2017. [6](#)
- [25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. [1](#), [2](#), [4](#)
- [26] Charles R. Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. [1](#), [2](#), [4](#), [5](#), [7](#)
- [27] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.*, 47(1-3):7–42, 2002. [2](#)
- [28] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Int. Conf. Comput. Vis.*, volume 2, pages 722–729, 1999. [1](#), [2](#)
- [29] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7794–7803, 2018. [4](#)
- [30] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. FlowNet3D++: Geometric losses for deep scene flow estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020. [2](#), [6](#), [7](#)

- [31] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Eur. Conf. Comput. Vis.*, pages 739–751. Springer, 2008. [1](#), [2](#)
- [32] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3D point clouds. In *Eur. Conf. Comput. Vis.*, 2019. [2](#), [6](#), [7](#)
- [33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1912–1920, 2015. [5](#)
- [34] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Adv. Neural Inform. Process. Syst.*, pages 794–805, 2019. [2](#)
- [35] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and Gumbel subset sampling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3323–3332, 2019. [2](#)
- [36] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016. [8](#)