2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) | 978-1-6654-4899-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CVPRW53098.2021.00315

Occlusion Guided Scene Flow Estimation on 3D Point Clouds

Bojun Ouyang
Tel Aviv University
bojungouyang@mail.tau.ac.il

Dan Raviv
Tel Aviv University
darav@tauex.tau.ac.il

Abstract

3D scene flow estimation is a vital tool in perceiving our environment given depth or range sensors. Unlike optical flow, the data is usually sparse and in most cases partially occluded in between two temporal samplings. Here we propose a new scene flow architecture called OGSF-Net which tightly couples the learning for both flow and occlusions between frames. Their coupled symbiosis results in a more accurate prediction of flow in space. Unlike a traditional multi-action network, our unified approach is fused throughout the network, boosting performances for both occlusion detection and flow estimation. Our architecture is the first to gauge the occlusion in 3D scene flow estimation on point clouds. In key datasets such as Flyingthings3D and KITTI, we achieve the state-of-the-art results.[1] [2]

1. Introduction

Scene flow estimation is a core challenge in computer vision which aims to find the 3D motion between points from consecutive temporal frames. While flows in between images, also known as optical flow, still have an important part in modern vision systems, the rise of depth sensors shifts the focus towards geometric flows. The two tasks are similar in spirit but with one fundamental gap - the data source for optical flow are regular dense samples given on top of a grid, while most depth sensors, especially outdoor, provide a sparse set of points in space. Algorithmic-wise, in the deep networks era, that gap shifts us from image-based convolutions towards graph neural network architectures.

Early attempts to solve 3D model alignment minimized the point-to-point or point-to-plane energy and were referred to as Iterative-Closest-Point (ICP) algorithms [5, 9], where during iterative steps one searches for the closest set of matched points and minimizes the energy on that subset. Rigid alignment [5] was first introduced, then rapidly non-rigid deformations were solved by adding adequate regularization [3]. Many different approaches for alignment appeared over the years. Just stating a few - [17, 7, 20, 14, 11] focused on RGB-D between stereo images, [35] introduced a more robust cost function, [36] considered the alignment as a quadratic assignment task and [1] added intrinsic long geodesics to enrich the process with global features.

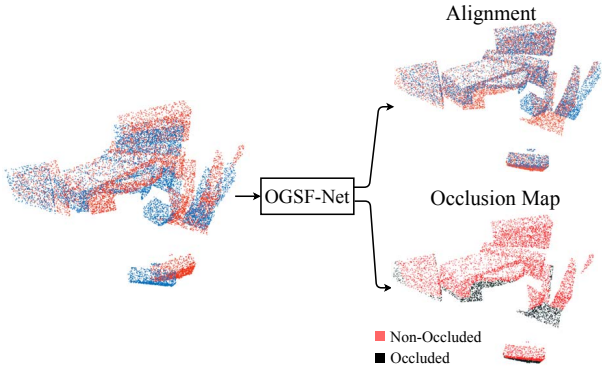


Figure 1: **Multi-task model**. OGSF-Net directly consumes the point clouds from two different frames as its input. It predicts the scene flow and occlusion map of the source relative to the target.

Moving from axiomatic methods towards learning-based approach became feasible lately, where graph convolutions evolved as well as rich and deep enough networks were capable of sensing more of the scene. FlowNet3D [23] was probably the first robust learnable deep network for aligning 3D point clouds. It utilizes a PointNet++ [33] structure and computes the correlation between the point clouds by using the flow embedding layer. Following that line of thought PointPWC-Net [53], based on an optical flow mechanism [44], uses Feature Pyramid Network on top of local correlations with a new cost volume and cost function, showing superior results all across the benchmarks. Lately, we have seen attempts to handle larger sets of deformations by correlating all points with all points [31] but those models require a massive increase in memory resources and suffer from outliers that need to be cleaned.

When calculating flow in between objects, we encounter in many cases the challenge of occlusions, where some regions in one frame do not exist in the other. Due to the displacement between the sensor and the object, the sen-

[1] Our code will be publicly available upon publication.
[2] https://github.com/BillOuyang/OGSFNet.git

978-1-6654-4899-4/21/$31.00 ©2021 IEEE
DOI 10.1109/CVPRW53098.2021.00315

sor does not see the entire object in all time steps. Incorrect treatment of the occluded area would reduce the performance of the flow estimation. That is true for optical flow tasks in images and of course for scene flow. Classical methods usually regularize incoherent motion to propagate flow from non-occluded pixels to the occluded region [40, 26]. That is also true in the deep learning era where occlusions were learned in addition to flow estimation. Those attempts worked well on regular grids but traditionally failed on a sparse set of points due to numerical challenges. In this work, we focus on that exact task and show for the first time that if we couple the task for flow and occlusion tight enough in a guided approach we can gain in both worlds; getting a more accurate flow and understanding what is occluded.

The main contributions of our work are:

- We propose a deep learning model called OGSF-Net which can jointly estimate the scene flow and occlusion map from point clouds.
- We utilize an occlusion handling mechanism inside our Cost Volume layer.
- We present a new residual multi-scale architecture in place of traditional multi-scale flow schemes.
- We show state-of-the-art performance on Flyingthings3D and KITTI Scene Flow 2015.

2. Related Work

Deep Learning on Point Clouds. Deep learning has been proved to be one of the most successful learning tools in image processing and swept the community towards new achievements over axiomatic modelling. Graph neural networks, focusing on a more generalized structure, where vertices and edges represent our data, followed the revolution, presenting exciting new tools to handle irregular data. In computer vision, point cloud is one very common way to represent geometry acquired by range sensors or generated in virtual worlds. We have seen papers coping with new challenges by sampling the data and projecting the points into volumetric lattices [54, 2, 32, 25] and later on focusing on point convolutions or a combination of edge and points pulling layers, known as message passing [48, 8, 33, 34, 42, 45, 43, 16, 12, 47, 22, 25]. Interesting follow-up papers appeared rapidly, trying to solve the main challenge arising from the permutation challenge in graphs. We do see recently different sampling strategies or different pulling methods. MLP layers and MAX-pooling are two relevant and popular building blocks for that [8, 34]. Another interesting and popular approach was using the points as raw data input [8], followed by a hierarchical architecture which can capture the local structure of the point clouds [33]. Treating point cloud as graph and performing convolution over local neighbourhood make a lot of sense, and several successful approaches were introduced lately focusing on the convolution engine [6, 41, 49, 15, 50, 52]. In our work, we use the PointConv suggested by [52, 53] to perform the convolutions on the point clouds.

Scene Flow Estimation on Point Clouds. The increasing popularity of range data gave birth to the need for fast and accurate mapping of point clouds. [10, 4, 39, 46] suggested estimating the scene flow directly from real LiDAR scans. [10, 4] consider the scene flow as a rigid motion, while [49, 23, 13, 53, 24, 31] remove those restrictions. Based on the [33] architecture, FlowNet3D [23] introduced a novel flow embedding layer that aggregates the features from different frames. However, they only applied the flow embedding at a certain scale which limits the allowed feasible gap between the frames. [53] introduced a neural network based on [44] which can predict the scene flow in a coarse-to-fine manner, showing superior results both for large and small flows. However, they do not have any treatment for the occlusions, and their accuracy decrease significantly when there are occluded regions in the point clouds. Recently, [38] suggests estimating the scene flow using both RGB and LiDAR data to overcome ambiguity by providing an additional layer of information. [31] proposed an interesting approach focusing on all-to-all correlation using graph matching.

Occlusion estimation in Scene Flow. Scene flow estimation and occlusion are treated as a chicken-and-egg problem as they are highly related to each other and one influence the other. Many papers [19, 20, 21, 40] suggest to predict the occlusion mask jointly with the flow and to refine the flow estimation by using the predicted occlusion mask. [18, 51, 28] suggest to predict both the forward and backward flow and to find the occluded region based on the warped images. In [21], they proposed an unsupervised training framework which can predict the optical flow and occlusion from multiple frames. Based on [44], PWOC-3D [40] suggests a self-supervised strategy for the occlusion estimation by masking the warped feature inside the Cost Volume layer using the occlusion map.

In this work, we suggest entangling the two aspects together all across the network, not just only in the cost function. We claim that the occlusion should guide the flow and vice-versa as part of the architecture itself to gain the most out of the two. To the best of our knowledge, we are the first to estimate the occlusion in 3D scene flow estimation on point clouds and the first to present a guided linked unit in the pipeline to solve the flow-occlusion coupled task. We present state-of-the-art alignment results over all methods described above on known datasets.

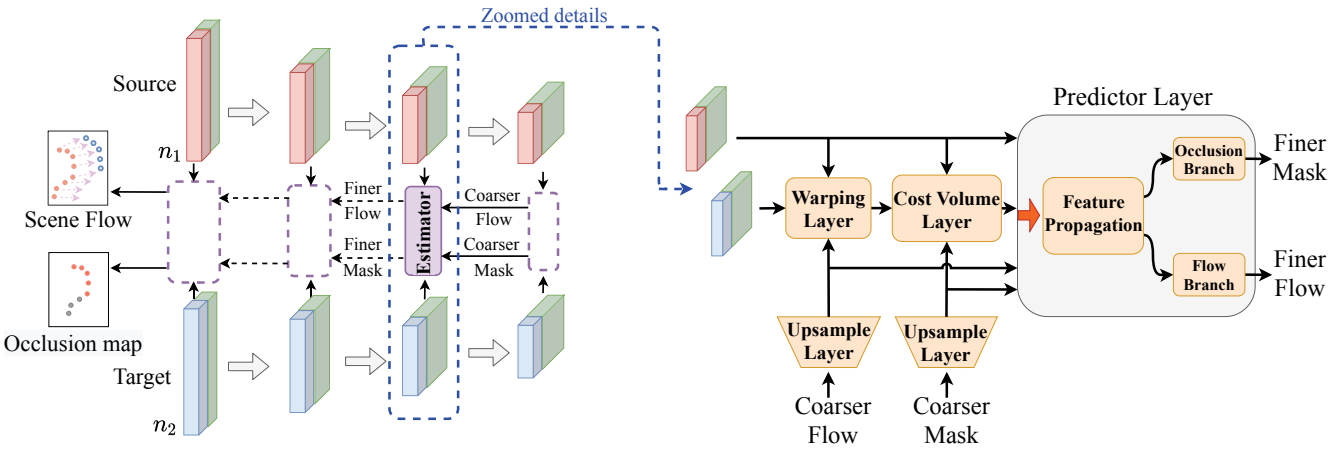


Figure 2: **Architecture**. On the left, we show the entire pipeline of OGSF-Net. It takes the input point cloud on the left and uses the PointConv+FPS to downsample the point cloud at each level. On the right, we show the finer details at each level. We first warp the target towards the source in order to construct our Cost Volume. Using the PointConv and MLP in the Feature propagation layer, we create the shared input features for the Flow/Occlusion branch.

3. Problem Definition

Given two samplings of a 3D scene we wish to estimate the movement in space between the source and the target and to identify the points in the source that do not appear in the target. We represent the two sampled scenes, source S and target T as point clouds. Specifically, $S = \{p_i | p_i \in \mathbb{R}^3\}_{i=1}^{n_1}$ with n_1 points and $T = \{q_j | q_j \in \mathbb{R}^3\}_{j=1}^{n_2}$ with n_2 points. Each point can also have a feature vector such as color or local normal to the surface. To ease notations, we denote $c_i \in \mathbb{R}^d$ the feature of point $p_i \in S$ and $g_j \in \mathbb{R}^d$ for the point $q_j \in T$.

Given that S represents a sampling of the source domain, we wish to find the flow in space of each point in S. We denote by $f_i \in \mathbb{R}^3$ the flow vector of point p_i which is shifted towards $p_i + f_i$ in the target domain. Note that we do not learn a correspondence between the source points and the target points but a flow representation of each point on the source. Due to a potential occlusion, some points in the source may not appear in the target frame. We note the occlusion of a point p_i in the source using a binary scalar $occ_i \in \{0, 1\}$, where 0 means occluded and 1 means non-occluded. Our goal is to find the scene flow $\{f_i\}_{i=1}^{n_1}$ and occlusion label $\{occ_i\}_{i=1}^{n_1}$ for every point in the source.

4. Architecture

Inspired by the architecture of [53], our network utilizes a feature pyramid structure and uses the point cloud from two different time frames as its inputs, where each point can have a rich feature vector such as color or normal to the surface. In the examples shown in this paper, we use the RGB color as our input point feature. See Figure 2 for network architecture. In each pyramid level, we first apply a backward warping of the target point cloud T towards the source S by using the upsampled flow from the previous level. Then, by using the features from the point clouds and the upsampled occlusion mask from the previous level, we construct our cost volume for each point in S. The cost volume is a widely used concept in stereo matching [44, 37]. It stores the point-wise matching cost and measures the correlation between the different frames. Finally, we predict the finer flow and mask by using the cost volume, features from S, upsampled flow and mask.

Feature Pyramid Structure. In order to extract the semantically strong features for the accurate flow and occlusion mask prediction, we construct a 4-level pyramid of features with the input at the top (*zeroth*) level. For each pyramid level *l*, we downsample the point clouds for the coarser level (*l+1*) by using the farthest point sampling (FPS) [33]. Followed by a PointConv [52] operation, we create and increase the number of the features for each downsampled point. The finer prediction of flow and mask at each level are made by using the upsampled prediction from its coarser level (except for the bottom level).

Warping. At each pyramid level, we first do a backward warping of the target points towards the source by using the upsampled scene flow from the previous coarser level. We use the same Upsample layer as in [53]. Since the warping layer brings the target "closer" to the source, neighborhood searching of the target around the source point would be more accurate during the cost volume construction. Denote the upsampled flow from the coarser layer as $\{f_i^{up}\}_{i=1}^{n_1}$. Inside the warping layer, we first do a forward warping from the source to the target:

$$S_w = \{p_{w,i} = p_i + f_i^{up}\}_{i=1}^{n_1} \quad (1)$$

For each point q_j in the target T, we compute its backward

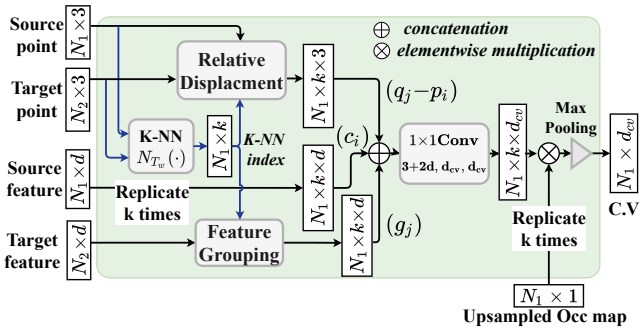


Figure 3: **Cost Volume Layer**. For each point in the source, we first find its k-NN point in the target. Then, we group the relative displacement $(q_j - p_i)$ and the features (g_j) of the neighborhood. After we calculate the matching cost, we apply the occlusion masking and the Max-pooling to construct the Cost Volume.

flow by using the weighted average of the upsampled flow:

$$f_j^b = \frac{\sum_{p_i \in N_{Sw}(q_j)} w(p_i, q_j) \times (-f_i^{up})}{\sum_{p_i \in N_{Sw}(q_j)} w(p_i, q_j)} \tag{2}$$

Where $N_{Sw}(q_j)$ is the K nearest neighbor (k-NN) of q_j on S_w, weight $w(p_i, q_j) = \frac{1}{d(p_i, q_j)}$ is simply the inverse of the euclidean distance between p_i and q_j. Finally, the warped target point will be the element-wise addition of the backward flow and itself:

$$T_w = \{q_{w,j} = q_j + f_j^b\}_{j=1}^{n_2} \tag{3}$$

Cost volume with Occlusion mechanism. Traditionally, occlusions play an essential role in the scene flow estimation on the 2D stereo frames. When it comes to the 3D point clouds, occlusion issues still exist due to the motion of the object and the camera position. The main impact of the occlusion is on the cost volume since the matching cost for the occluded point is not available. Similar to the images, the occlusion in the source point cloud relative to the target can be modeled as a map: $OCC_{S-T} : S \rightarrow [0, 1]$ where 0 stands for the occluded point, 1 stands for the non-occluded. FlowNet3D [23] uses a flow embedding layer to aggregate the features and spatial relationships for each neighbor in the target around the source. Since their model only finds the neighboring points within a certain radius, it is somehow robust to the occlusion as the relative displacement between the occluded point and target is usually large. PointPWC-Net [53] suggests a novel cost volume that can aggregate the features of both input point cloud in a patch-to-patch manner. However, for the occluded regions in the source, this feature aggregation operation can be incorrect since they do not have a correspondence in the target frame. Inspired by PWOC-3D [40], we suggest a novel occlusion mechanism that helps the construction of our cost volume.

One of the critical components of cost volume is the matching cost. It measures the similarity between the source point and the target point. Since we believe that the correlation between the points is highly related to their features and relative displacement, for the *non-occluded* point p_i, the matching cost between p_i and q_j is calculated by

$$cost(p_i, q_j) = h(c_i, g_j, q_j - p_i) \tag{4}$$

Where $h(\cdot)$ is simply a concatenation of its input followed by 1×1 convolution layers, c_i and g_j are the corresponding features of $p_i \in S$ and $q_j \in T_w$. When it comes to the *occluded* points p_i , we expect to get a matching cost of 0, as they do not have a correspondence in the target frame. As shown in Figure 3, by using our definition of occlusion map, we can calculate our matching cost of p_i with q_j as:

$$cost(p_i, q_j) = OCC_{S-T}(p_i) h(c_i, g_j, q_j - p_i) \tag{5}$$

In our case, we use the upsampled predicted occlusion mask from its coarser layer as the occlusion map in Eq. 5.

After we calculate the matching cost, we can aggregate them to form the cost volume. Theoretically, we can use all possible pairs of (p_i, q_j) in our calculation, but this is inefficient in terms of the computation. With the help of the Warping layer, we can assume that the correct corresponding point pairs between source and target are relatively close to each other. For this reason, we only aggregate the matching cost of the nearest target neighbor for every point in the source. It can be summarized in the following form:

$$CV(p_i) = \underset{q_j \in N_{Tw}(p_i)}{Aggregation}\{cost(p_i, q_j)\} \tag{6}$$

where $N_{Tw}(p_i)$ is the nearest neighborhood of the source point p_i in the warped target T_w.

In the cost volume layer of [53], they use a learnable weighted sum based on the relative distance as the aggregation function to calculate their Point-to-Patch cost. This implies that the proportion of the matching cost between (p_i, q_j) in $CV(p_i)$ only depends on their relative displacement $(q_j - p_i)$. However, in many cases, the correlation between the points depends on their features but not their relative displacement, the correct corresponding pairs can have less contribution to the cost volume by using this aggregation design. In our work, we decide to use the max-pooling to aggregate the matching cost. The intuition is that, to make an accurate prediction of the flow and mask, the model needs the matching cost of the correct corresponding pairs to have the highest contribution in the cost volume. Using max-pooling can force their matching cost to have the highest value among the neighborhood $N_{Tw}(p_i)$ during training. This choice of design also agrees with our definition of the matching cost above. To summarize, we calculate the cost volume for every point p_i by using the following equation:

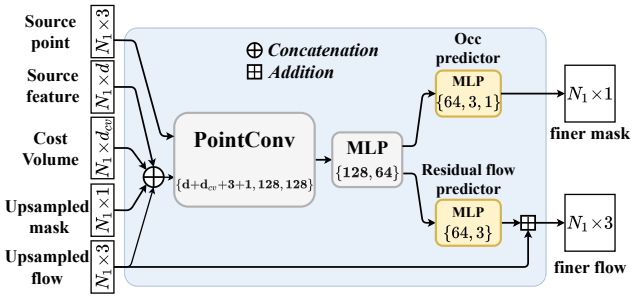


Figure 4: **Predictor Layer**. Our Predictor layer takes several inputs and produces the scene flow and occlusion mask at current level. These outputs will be upsampled and used as one of the inputs in the the next pyramid level.

$$CV(p_i) = \underset{q_j \in N_{Tw}(p_i)}{MAX} \{cost(p_i, q_j)\} \tag{7}$$

Predictor Layer. In order to make the final prediction of the flow and occlusion mask at each pyramid level, we use a predictor layer. As shown in Figure 4, this layer contains a feature propagation module followed by two predictor branches. In the feature propagation module, we first concatenate all its inputs along the feature dimension. Then by using several PointConv and Multilayer perceptron (MLP), we generate the final features for the flow and mask prediction. The inputs of the feature propagation module are the features of the source, masked cost volume described above, upsampled flow and upsampled occlusion mask. After the feature propagation layer, we connect a flow predictor and occlusion predictor in parallel. Since we believe that the scene flow and occlusion are highly related to each other, we decide to use the shared input features for the two branches. Our flow predictor consists of a single MLP layer such that the output tensor has a dimension of $(n_1, 3)$. Unlike the PointPWC-Net [53], our flow predictor only predict a *residual* flow vector such that the final scene flow is the element-wise addition of the upsampled flow from the previous level and the residual flow for every point in the source. By using this residual flow design, we solve the scene flow estimation problem in an iterative approach and we get a stronger correlation between the consecutive pyramid levels. Shifting from multi-scale flow estimation to multi-scale residual improve the results significantly and we show that in the ablation study.

For the occlusion branch, we use a 2-layer MLP with leaky-ReLU activation in the middle to process the input features. We also connect a sigmoid activation layer in the end. This ensures the output to be a probability distribution with value in the range [0, 1].

5. Loss functions

We train our model in a supervised manner with the ground truth scene flow and occlusion mask. Since the existing scene flow dataset with real scans is too small for the training, we adopt a similar training scheme as in the previous work [23, 53]. We first train our model with the synthetic data from the FlyingThings3D [27], then we test it with the real LiDAR scans from the KITTI [29, 30]. We show that OGSF-Net has the best generalization ability to the unseen data from KITTI in the experiment section. In order to predict both the scene flow and occlusion map, we use two loss terms to train our model.

Scene flow loss. We use a similar loss function as in [23] and [53] for the flow estimation. Let f_i' be the ground truth flow and f_i be the predicted flow for the point $p_i \in S$. Let occ_i' be the ground truth occlusion label for p_i with the value in $\{0, 1\}$. We use a multi-level loss for the flow as below

$$F_{loss}(\Theta) = \sum_{l=0}^{3} \alpha_l \sum_{p_i \in S_l} occ_i' \left\| f_i - f_i' \right\|_2 + \left\| f_i - f_i' \right\|_2 \tag{8}$$

Where Θ is the learnable parameters of OGSF-Net, S_l is the sampled point cloud at pyramid level l, and α_l is the weight for each level. The $first$ term in the inner summation penalizes the L_2 norm of the errors in estimated flow for non-occluded regions. Since we also want to predict the flow for occluded regions, we add the $second$ term which penalizes the error for all points in every S_l and it improves the performance through our experiments.

Occlusion loss. At each pyramid level, we use the predicted occlusion map to construct our masked cost volume. It means accurate mask prediction is also important for flow estimation at each level. Let occ_i' be the ground truth occlusion label and occ_i be the predicted label for the point $p_i \in S$. We use a similar occlusion loss as the flow loss:

$$O_{loss}(\Theta) = \sum_{l=0}^{3} \beta_l \sum_{p_i \in S_l} \left\| occ_i - occ_i' \right\| \tag{9}$$

The overall loss function we used is simply the combination of the flow and occlusion loss from each pyramid level:

$$L(\Theta) = F_{loss}(\Theta) + \lambda \cdot O_{loss}(\Theta) \tag{10}$$

We use the λ as a weight to control the balance between the flow loss and occlusion loss.

6. Experiments

In this section, firstly, we compared the performance of our OGSF-Net with previous work on the FlyingThings3D [27] synthetic dataset on several evaluation metrics. Without any fine-tuning, we also test our model's generalization ability on the real scans from KITTI [29, 30].

Dataset	Method	EPE$_{full}\downarrow$	EPE$\downarrow$	ACC$_{05}\uparrow$	ACC$_{10}\uparrow$	Outliers$\downarrow$
Flyingthings3D	ICP [5]	0.5048	0.4848	0.1215	0.2558	0.9441
	FlowNet3D [23]	0.2119	0.1577	0.2286	0.5821	0.8040
	HPLFlowNet [13]	0.2012	0.1689	0.2629	0.5745	0.8123
	FLOT($K = 1$) [31]	0.2502	0.1530	0.3965	0.6608	0.6625
	PointPWC-Net [53]	0.1953	0.1552	0.4160	0.6990	0.6389
	Ours	**0.1634**	**0.1217**	**0.5518**	**0.7767**	**0.5180**
KITTI	ICP [5]	0.3801	-	0.1038	0.2913	0.8307
	FlowNet3D [23]	0.1834	-	0.0980	0.3945	0.7993
	HPLFlowNet [13]	0.3430	-	0.1035	0.3867	0.8142
	FLOT($K = 1$) [31]	0.1303	-	0.2788	0.6672	0.5299
	PointPWC-Net [53]	0.1180	-	0.4031	0.7573	0.4966
	Ours($without\ ft$)	0.0751	-	0.7060	0.8693	0.3277
	Ours($with\ ft$)	**0.0333**	-	**0.8913**	**0.9517**	**0.1915**

Table 1: **Performance on Flyingthings3D and KITTI**. All the models in the table are trained on the occluded Flyingthings3D using 8192 points. We test it on KITTI (with occlusion) using 8192 points from each frame $without$ any fine-tuning. Notice that we outperforms all other methods by a large margin. In the last column, we also present our fine-tuned results on KITTI.

By further fine-tuning on KITTI, we show improvements in the results and present visualization on KITTI. In the previous works, there are two versions of FlyingThings3D and KITTI that have been proposed. The first one is suggested by [13], where the occluded point is removed from the processed point cloud and many difficult examples in the Flyingthings3D have been removed. The second version is suggested by FlowNet3D [23]. The occluded region remains and the occlusion map for FlyingThings3D is provided. Since our work is highly related to the occlusion, we adopt the FlyingThings3D and KITTI proposed by [23], which is more challenging than the first version. Secondly, in the ablation study, we test our design choices and show the effectiveness of all the novel components in our work. Finally, we evaluate our occlusion estimation. To the best of our knowledge, we are the first one to evaluate the occlusion on scene flow estimation on point clouds. We present here state-of-the-art results compared to those of previous reported methods.

Evaluation Metric. We first adopt the four evaluation metrics used in [13, 23, 53, 31]: averaged end point error (EPE); two accuracy measurement with a different threshold on EPE; outlier ratio with a threshold on the EPE. In [23, 31], the above metrics are evaluated on the non-occluded points only, while in our work, we evaluate results for all the points, include occluded and non-occluded ones. The details of the evaluation metrics are as follow:

- EPE_{full}(m): $\left\|f_i - f_i'\right\|_2$ averaged over **all** $p_i \in S$.
- EPE(m): $\left\|f_i - f_i'\right\|_2$ averaged over all **non occluded** points.
- ACC_{05}: percentage of points whose $EPE_i < 0.05$m or EPE_i / $\left\|f_i'\right\|_2 < 5\%$
- ACC_{10}: percentage of points whose $EPE_i < 0.1$m or EPE_i / $\left\|f_i'\right\|_2 < 10\%$
- $Outlier$: percentage of points whose $EPE_i > 0.3$m or EPE_i / $\left\|f_i'\right\|_2 > 10\%$

Implementation Details. Our OGSF-Net utilizes the same feature pyramid structure as in [53] to process the input point clouds, while the number of points we used in each downsampled point cloud is $[2048, 512, 256, 128]$. We choose the weight α in the Eq. 8 to be $\alpha = [\alpha_l]_{l=0}^{3} = [0.02, 0.04, 0.08, 0.16]$. The weight β in Eq. 9 is set to be $\beta_l = 1.4\alpha_l$ for every pyramid level l. The number of features d at each level is set to be $[64, 96, 192, 320]$ and d_{cv} at each level is $[32, 64, 128, 256]$. All hyper-parameters are selected according to the validation set of Flyingthings3D. We trained our model with 2×GTX2080Ti GPU on FlyingThings3D with batch size of 8 and 120 training epochs, and it took one day to train. We start with a learning rate of 0.001 and reduce it after every 10 epochs with a decay rate 0.85. We further reduce the decay rate to 0.8 after 75 epochs. The balancing weight λ is 0.3 initially. In order to improve the occlusion accuracy, we increase the λ gradually to 0.6 in the first 45 epochs.

6.1. Evaluation on Flyingthings3D

Since the acquisition of dense flow and occlusion mask from the real scene is difficult, to the best of our knowledge, there is no real-world large-scale scene dataset published with ground truth flow and mask. Thus, by following the similar evaluation process in [23, 31, 53, 13], we trained our model on the synthetic FlyingThings3D [27] dataset. As mentioned before, we use the same dataset suggested by [23], it contains 20000 pairs of the point cloud in the training set and 2000 in the test set.

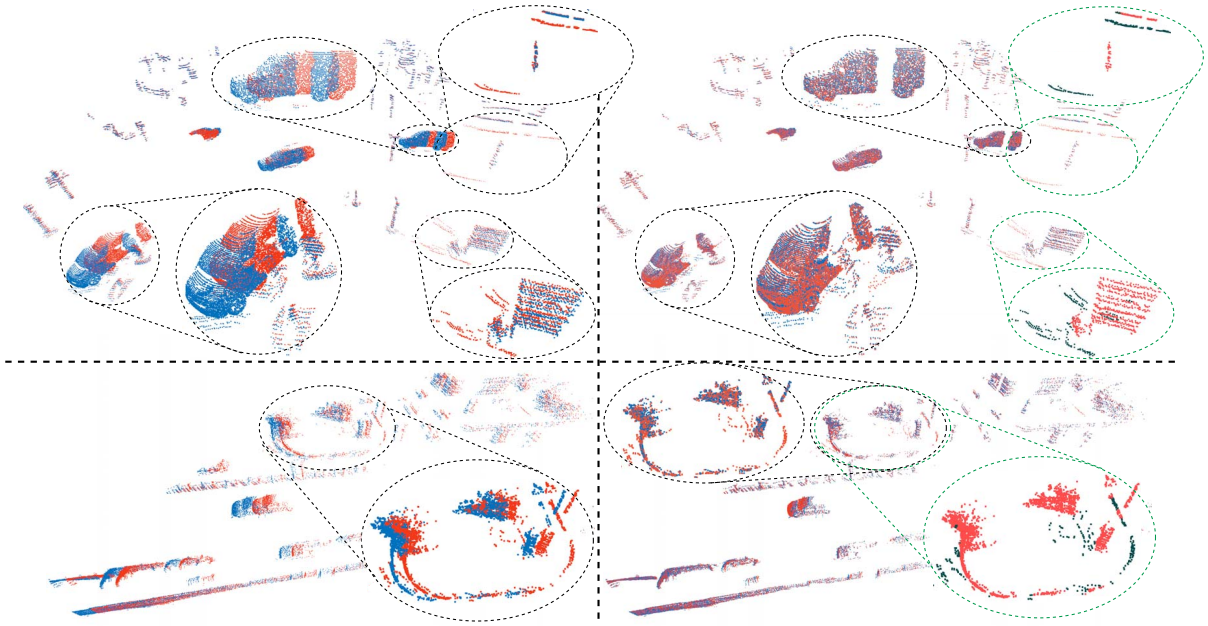


Figure 5: **Visualization on KITTI Scene Flow 2015**. For the images on the left, we show the source (red) and target (blue) point cloud on the same 3D space before the alignment. For the images on the right, we align the source towards the target by using the predicted flow from OGSF-Net (source+scene flow). The zoom-in view for the region circled by black is shown. We also provide the zoom-in detail of our predicted occlusion map for the region circled by green. We can see that our OGSF-Net can predict the map for the occluded points (black) and non-occluded points(red) correctly, it can also estimate the accurate flow for both occluded and non-occluded regions.

Since both the ground truth scene flow and occlusion mask are provided in this dataset, we use the loss function in Eq.10 to train our model.

The detailed comparison results are shown in Table 1. We compared our model with the previous state-of-the-art methods on point cloud scene flow estimation. All the methods were trained on Flyingthings3D proposed by [23], we use $n_1 = n_2 = 8192$ points for each point cloud for the training and evaluation. It is clear to see that our method outperforms the previous work in all evaluation metrics. As mentioned in the related work, when compared the numbers in Table 1 to the reported results in their own paper, we can see that the performance of [53, 13] is highly degraded due to the existence of occlusion in the input. Notice that the performance of FlowNet3D [23] and FLOT [31] is acceptable on the EPE, but they perform much worse on the EPE$_{full}$. This is because they removed the errors for the occluded region in their loss function and they are not able to predict the flow for the occluded points.

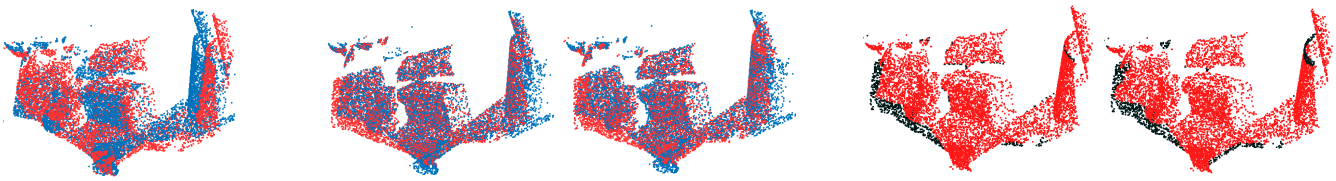
6.2. Evaluation on KITTI

In order to test the generalization ability on real scans, we first trained our model on Flyingthings3D then test it on all the 150 examples with $n_1 = n_2 = 8192$ points from KITTI Scene Flow 2015 [29, 30] without any fine-tuning. Since they do not provide the ground truth occlusion map for the source, we cannot evaluate the EPE on the KITTI.

As shown in Table 1, our model has the best generalization ability compared to the previous work. On the last row in the Table 1, we split the data to 100 training samples for fine-tuning and 50 samples for test, we show a further improvement in the performance. Since there is no ground truth occlusion mask, we only use $\sum \alpha_l \sum \left\| f_i - f_i' \right\|_2$ ($second$ term of the $F_{loss}(\Theta)$) as the loss function for the fine-tuning.

6.3. Ablation Study

We performed several ablation studies to validate our model's design choices, occlusion guided mechanism, and loss functions. In Table 2 (a), we report the EPE of a different combination of the design choices on the Flyingthings3D dataset. When we use Max-pooling to aggregate the matching cost in the Cost Volume layer, we obtain significantly better results in terms of the EPE. By further using our residual flow prediction design instead of the full scene flow prediction, we got a 19% improvement in the performance. In the last two rows, we show that our model's performance on the occluded dataset improved by a large margin by utilizing the occlusion estimation mechanism. In Table 2 (b), we train our model using the different loss functions and present the EPE and EPE$_{full}$ on the Flyingthings3D and KITTI respectively. As shown in the bottom row, OGSF-Net can distinguish between the occluded and non-occluded regions by training with the occlusion loss. It improves the performance on the Flyingthings3D and we got a better generalization ability on KITTI.



a) b) gt c) pred d) gt e) pred

Figure 6: **Flow/Occlusion Visualization on Flyingthings3D**. An example from the test set of Flyingthings3D. a) shows the source (red) and target (blue) frames, b) and c) show the alignment results by using the ground truth and predicted flow, d) and e) show the ground truth and predicted occlusion map where the non-occluded region is marked in red and the occluded region is marked in black.

Aggregation	Occ. Predictor	Masked C.V	Flow branch	EPE↓
Weighted sum	✗	✗	Full	0.1610
Weighted sum	✓	✗	Full	0.1541
Weighted sum	✓	✓	Full	0.1512
Max-Pooling	✗	✗	Full	0.1503
Max-Pooling	✗	✗	Residual	0.1304
Max-Pooling	✓	✓	Residual	**0.1217**

(a) Design choice

FLow loss	Occlusion loss	Flyingthings3D	KITTI
✓	✗	0.1337	0.0794
✓	✓	**0.1217**	**0.0751**

(b) Loss function

Table 2: **Ablation Studies for the model design**.(a) we show the different combination of design choices, and ours can get the best performance.(b) by training with the occlusion, we can get a much better generalization on the real scans from KITTI.

6.4. Occlusion Estimation

An accurate occlusion prediction is important for our occlusion-guided mechanism and important for some applications like 3D object reconstruction. In this section, we evaluate the performance of occlusion estimation on the Flyingthings3D only as there is no other public dataset on point cloud that provides the ground truth occlusion mask. We use the standard occlusion estimation metrics, accuracy and F1-score, as our evaluation metrics. We first convert the predicted occlusion probabilities to the label $\{0, 1\}$ using threshold value 0.5. Then, we measure the two metrics and we get 94.91% and 0.824 respectively. We also showed some visualization of the occlusion estimation results in Fig 5 and 6.

6.5. Outlier ratios

In the scene flow estimation, outlier ratios are important as they measure the robustness of the model. In Table 3,

Method	Threshold for Outlier (m)				
	0.1	0.2	0.3	0.4	0.5
FlowNet3D [23]	67.87	29.15	14.41	7.83	4.46
FLOT [31]	37.34	16.69	9.22	5.37	3.37
Ours	**13.82**	**6.54**	**4.78**	**3.85**	**3.27**

Table 3: **Outlier ratio**. We measure the outlier ratios with different threshold values. We only compared our model with the FlowNet3D and FLOT as they are the only models trained and tested with occluded data in their works.

we show outlier ratios on KITTI Scene Flow 2015 [29, 30] with different threshold values for different models. We calculate the ratio by simply finding the percentage of the point whose EPE_{full} is greater than the given threshold. As we can see, the performance of [31] and ours is much better than [23]. For all the threshold values from 0.1 to 0.5, our model has the smallest outlier ratio compared to the FlowNet3D [23] and FLOT [31].

7. Conclusion

In this paper, we suggest a deep neural network called OGSF-Net that can jointly estimate the scene flow and occlusion map directly from the point cloud data. We are the first to introduce the idea of occlusion estimation on the point cloud scene flow estimation, and by using our masking operation inside the Cost Volume layer, we show a significant improvement in the flow accuracy. Our occlusion guided flow estimation not only provides an additional layer of information but outperforms previously reported state-of-the-art models by a large margin, on multiple datasets and for different metrics.

8. Acknowledgment

This work is partially funded by the Zimin Institute for Engineering Solutions Advancing Better Lives, the Israeli consortiums for soft robotics and autonomous driving, the Nicholas and Elizabeth Slezak Super Center for Cardiac Research and Biomedical Engineering at Tel Aviv University and TAU Science Data and AI Center.

References

[1] Yonathan Aflalo, Ron Kimmel, and Dan Raviv. Scale invariant geometry for nonrigid shapes. *SIAM Journal on Imaging Sciences*, 6(3):1579–1597, 2013. 1

[2] Michael Allen. Voxnet: Reducing latency in high data rate applications. *Wireless Sensor Networks*, page 115–158, 2010. 2

[3] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid icp algorithms for surface registration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 1

[4] Aseem Behl, Despoina Paschalidou, Simon Donne, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[5] Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In Paul S. Schenker, editor, *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586 – 606. International Society for Optics and Photonics, SPIE, 1992. 1, 6

[6] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks, 2016. 2

[7] Jan Cech, Jordi Sanchez-Riera, and Radu Horaud. Scene flow estimation by growing correspondence seeds. *Cvpr 2011*, 2011. 1

[8] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[9] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. 1

[10] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard. Rigid scene flow for 3d lidar scans. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1765–1770, 2016. 2

[11] Jens-Malte Gottfried, Janis Fehr, and Christoph S. Garbe. Computing range flow from multi-modal kinect data. *Advances in Visual Computing Lecture Notes in Computer Science*, page 758–767, 2011. 1

[12] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018. 2

[13] 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. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 6, 7

[14] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. *2013 IEEE International Conference on Robotics and Automation*, 2013. 1

[15] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics*, 37(6):1–12, Jan 2019. 2

[16] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[17] Frederic Huguet and Frederic Devernay. A variational method for scene flow estimation from stereo sequences. *2007 IEEE 11th International Conference on Computer Vision*, 2007. 1

[18] Junhwa Hur and Stefan Roth. Mirrorflow: Exploiting symmetries in joint optical flow and occlusion estimation. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[19] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *CVPR*, 2019. 2

[20] 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. *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, page 626–643, 2018. 1, 2

[21] Joel Janai, Fatma G"uney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *European Conference on Computer Vision (ECCV)*, volume Lecture Notes in Computer Science, vol 11220, pages 713–731. Springer, Cham, Sept. 2018. 2

[22] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on $\mathcal{X}$-transformed points, 2018. 2

[23] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. *CVPR*, 2019. 1, 2, 4, 5, 6, 7, 8

[24] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, 2019. 2

[25] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019. 2

[26] Kunming Luo, Chuan Wang, Nianjin Ye, Shuaicheng Liu, and Jue Wang. Occinpflow: Occlusion-inpainting optical flow estimation by unsupervised learning, 2020. 2

[27] 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. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5, 6

[28] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, New Orleans, Louisiana, Feb. 2018. 2

[29] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5, 6, 7, 8

[30] M. Menze, C. Heipke, and A. Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS Annals of Photogram-*

metry, Remote Sensing and Spatial Information Sciences, II-3/W5:427–434, 2015. 5, 6, 7, 8

[31] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *European Conference on Computer Vision*, 2020. 1, 2, 6, 7, 8

[32] Charles R. Qi, Hao Su, Matthias Niebner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[33] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 1, 2, 3

[34] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds, 2016. 2

[35] Dan Raviv, Michael M. Bronstein, Alexander M. Bronstein, Ron Kimmel, and Nir Sochen. Affine-invariant diffusion geometry for the analysis of deformable 3d shapes. *Cvpr 2011*, 2011. 1

[36] Dan Raviv, Anastasia Dubrovina, and Ron Kimmel. Hierarchical matching of non-rigid shapes. *Lecture Notes in Computer Science Scale Space and Variational Methods in Computer Vision*, page 604–615, 2012. 1

[37] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR 2011*, pages 3017–3024, 2011. 3

[38] Rishav, Ramy Battrawy, René Schuster, Oliver Wasenmüller, and Didier Stricker. DeepLiDARFlow: A Deep Learning Architecture For Scene Flow Estimation Using Monocular Camera and Sparse LiDAR. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2

[39] Rishav, Ramy Battrawy, René Schuster, Oliver Wasenmüller, and Didier Stricker. Deeplidarflow: A deep learning architecture for scene flow estimation using monocular camera and sparse lidar, 2020. 2

[40] Rohan Saxena, René Schuster, Oliver Wasenmüller, and Didier Stricker. PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019. 2, 4

[41] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[42] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[43] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 2

[44] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. to appear. 1, 2, 3

[45] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[46] A. K. Ushani, R. W. Wolcott, J. M. Walls, and R. M. Eustice. A learning approach for real-time temporal scene flow estimation from lidar data. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5666–5673, 2017. 2

[47] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[48] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[49] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[50] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 2

[51] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[52] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3

[53] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020. 1, 2, 3, 4, 5, 6, 7

[54] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2