3D SCENEFLOWNET: SELF-SUPERVISED 3D SCENE FLOW ESTIMATION BASED ON GRAPH CNN

Yawen Lu[1]*, Yuhao Zhu*[2]*, Guoyu Lu*[1]
[1]Intelligent Vision and Sensing Lab, Rochester Institute of Technology, USA
[2]Computer Science Department, University of Rochester, USA

2021 IEEE International Conference on Image Processing (ICIP) | 978-1-6654-4115-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIP42928.2021.9506286

ABSTRACT

Despite deep learning approaches have achieved promising successes in 2D optical flow estimation, it is a challenge to accurately estimate scene flow in 3D space as point clouds are inherently lacking topological information. In this paper, we aim at handling the problem of self-supervised 3D scene flow estimation based on dynamic graph convolutional neural networks (GCNNs), namely **3D SceneFlowNet**. To better learn geometric relationships among points, we introduce EdgeConv to learn multiple-level features in a pyramid from point clouds and a self-attention mechanism to apply the multi-level features to predict the final scene flow. Our trained model can efficiently process a pair of adjacent point clouds as input and predict a 3D scene flow accurately without any supervision. The proposed approach achieves superior performance on both synthetic ModelNet40 dataset and real LiDAR scans from KITTI Scene Flow 2015 datasets.

Index Terms— 3D Scene Flow Estimation; Graph CNN; Self-supervised Learning; 3D Scene Understanding.

1. INTRODUCTION

Scene flow describes the 3D motion of points, which can be used to understand dynamic scenes. With the accurate scene flow, various high-level applications in 3D computer vision are able to be improved such as visual odometry, scene reconstruction, object segmentation, etc. Estimating scene flow directly from two point clouds in 3D space is still demanding, because most existing works [1] [2] [3] [4] typically generate 3D scene flows from 2D optical flows and stereo/RGB-D information, which still mainly rely on 2D images and misses the direct 3D structure information. Such methods cannot be applied with 3D input data.

There are some existing methods focusing on 3D point cloud and flow estimation, especially in robotics [5] [6]. Ushani et al. [5] proposed to apply expectation-maximization (EM) to study scene flow estimation from LiDAR Data. Dewan et al. [6] formulated the estimation of dense rigid scene flow from 3D LiDAR scans as a problem of minimizing the energy function. However, these approaches mainly build a multi-stage system to extract hand-crafted features, and did not benefit from deep neural networks.

Most deep learning based approaches for processing point cloud target at 3D classification and segmentation tasks. For example, PointNet [7] is a novel network to directly process unordered point data. It learns a spatial encoding of each scattered point individually and then aggregates them together to obtain a global feature distribution. An improved version PointNet++ [7] processed a group of 3D points in a hierarchical fashion. In this way, the method is able to capture both local and global features at multiple scales to achieve robust and detailed capturing. PointCNN [8] proposed the X-conv operator to replace commonly-used Conv layers in grid-based CNN to better handle irregular and unordered point clouds. Wang et al. [9] exploited local geometric features by building a graph and proposed a convolution-like operator on the connected edges for 3D segmentation. However, these works all focus on dealing with static point clouds, and mainly target classification and segmentation.

There are limited attempts on 3D scene flow estimation with deep learning networks most recently. FlowNet3D [10], as the 3D counterpart of FlowNet [11] and FlowNet 2.0 [12], learns deep hierarchical features of point clouds and flow embeddings based on [7]. HPLFlowNet [13] estimated scene flow from large-scale point groups by introducing three schemes Down Bilateral Convolutional Layers (DownBCL), UpBCL, and CorrBCL to achieve a fast and computational cost-saving method. However, such methods are still in a supervised manner with point-to-point correspondence to regress ground truth scene flow, and are not suitable for dynamically computing point features at each layer.

In this work, we propose a novel deep learning architecture for tackling the scene flow estimation problem. The proposed network dynamically estimates scene flow directly on 3D point clouds without supervision. Our network first computes edge features on top of PointNet from two input point clouds to help construct a local graph and edge embeddings, which promote the network to learn both global and local geometric features. The network model computes the dynamic graph and then subsequently applies CNN to the graph to recompute it at each layer. Then the extracted features learned from both local and global context are fed into an attention mechanism to further learn co-contextual information by combing embedding features from both source and target point clouds. In the generation of the final scene flow, we propose a new module called EdgeDeConv to perform the accumulation the features to produce the scene flow for 3D

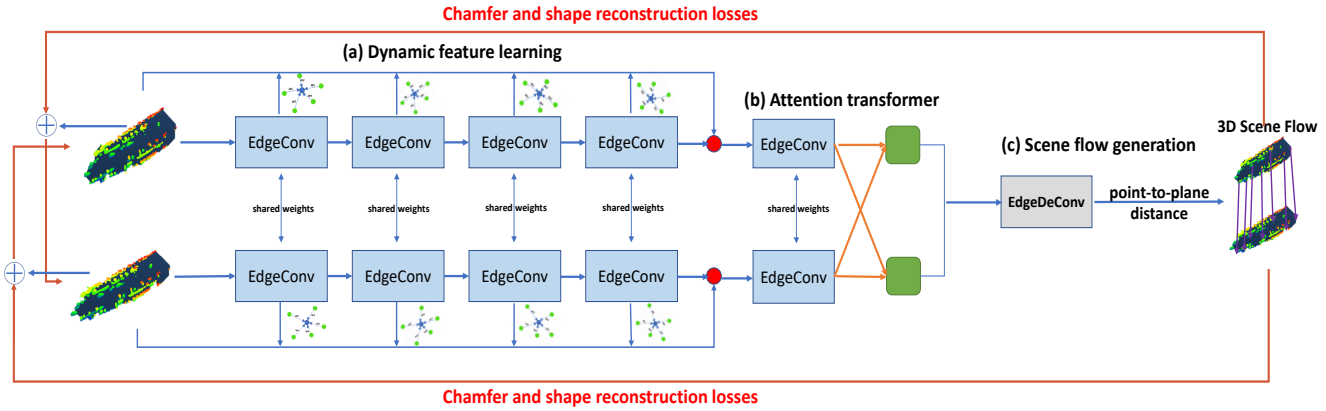

Fig. 1. Overview of our self-supervised training framework. Our proposed graph convolutional network directly predicts scene flow motion in 3D space by taking two point clouds as input. Our network is composed of three major components: (a) Dynamic feature learning; (b) Attention-based transformer; (c) Scene flow generation.

flow generation. Finally, the second input point cloud is able to be synthesized by warping the first input point cloud and the predicted scene flow, to construct our self-supervised 3D reconstruction constraint. The entire framework architecture is provided in Fig. 1.

We summarize our contributions as follows: (a) To the best of our knowledge, our work is highlighted to be the first one to leverage dynamic graph convolutional neural network on self-supervised scene flow estimation; (b) We introduce two 3D reconstruction losses: Chamfer reconstruction Loss and Laplacian shape reconstruction loss, to synthesize the second input point cloud from the predicted scene flow; (c) We propose a new EdgeDeConv module, followed by the attention transformer to recover the embedded features and produce the final scene flow.

2. SCENE FLOW ESTIMATION FRAMEWORK

The proposed self-supervised scene flow estimation method, **3D SceneFlowNet**, is composed of three major components: Dynamic point feature learning, attention transformer and scene flow generation. In this section, we will describe each process and our designed self-supervised loss constraints for the network training.

2.1. Dynamic feature learning

Given point clouds as input, usually the first step for most existing methods is to embed each set of 3D points into a higher dimensional space by feature transformation, and then predicting a global vector for the entire point cloud by aggregating point features with max pooling operation [7] [14].

However, we target at embedding each point as one feature independently and incorporating both local structure with global feature together to build the representation inspired by [9]. Specifically, given two input point clouds, we first construct a k-NN graph on k nearest neighbored points, and then compute the edge features on those points by applying a nonlinear function to each of the two edge endpoints to generate an edge-wise weight to connect each of the two vertices, and save the weights as trainable parameters to help with the network learning process later. The accumulated output from multiple EdgeConv blocks is an aggregation of vertex points at each layer. To simultaneously combine the global features from all layers and learn local information from connected vertex edges (x_i^m, x_j^m), the updated output for vertex i at each level m can be expressed as:

$$x_i^m = ReLU(\alpha_\theta(x_{i-1}^m - x_{j-1}^m) + \beta_\theta(x_{i-1}^m)) \qquad (1)$$

where α_θ and β_θ are trainable weights to be updated dynamically in the graph network. The combined edge features for the entire group of points are a combination of extracted features on each point at every layer: $\mathbf{X} = \sum_{m=1}^{p}\sum_{i=1}^{n} x_i^m$. Different from previous works, which directly feed the embedded point features into a multi-layer perception (MLP), we further perform the last EdgeConv layer without pooling to generate the refined edge features with the dimension of $n \times (3{+}c)$ (n is the number of 3D points, c is the channel number of the mixed features).

Following the extracted and combined edge features, we dynamically recompute the entire graph at each layer during the training phase. From the dynamic updates, the graph CNN is able to adaptively construct the graph instead of fitting a fixed setting, which facilitates to deal with various motions, like scene flow in this work. The effectiveness of dynamic updating is demonstrated in the ablation analysis parts of Table 1 and 2.

2.2. Attention transformer

In order to better learn the flow motion from the two input point clouds jointly, we adopt an attention mechanism to establish the inner links between the embeddings of point groups X and Y together, instead of computing them separately. The motivation to leverage attention model is to combine integrated features from both point clouds to address the shortcomings of focusing on one point cloud only.

The process is depicted in the attention transformer component in Fig. 1. Specifically, if the embedded features learned from the last EdgeConv module are X and Y for the two input point clouds respectively, the designed attention mechanism is able to learn a mapping representation from X and Y to X' and Y', where X' and Y' are new transformed embeddings incorporating both self information and mutual information. A simple version of our attention mechanism is to compute a linear combination $\tau(\cdot)$ on the embedded features as:

$$X' = X + \tau(X, Y); \qquad Y' = Y + \tau(Y, X) \tag{2}$$

To improve the linear combination and aggregate the inputs over multiple times with different transformation relationships, we modify the position-wise linear combination $\tau(\cdot)$ to a non-linear function $\gamma(\cdot)$ and add additional learnable coefficients α and β to indicate the weights assigned for self-attention and cross-attention. The improved expression then turns into:

$$\begin{aligned} X' &= X + \alpha\, ELU(GN(X, Y)) \\ Y' &= Y + \beta\, ELU(GN(Y, X)) \end{aligned} \tag{3}$$

where γ is a non-linear transformation by a Group Normalization [15] followed by ELU [16] activation function.

2.3. Scene flow generation

With the encoded features from the attention model, we introduce the **EdgeDeConv** for the first time to perform feature accumulation to produce the final predicted scene flow in 3D space. By decoding the attentioned embeddings as an opposite process of edge-wise encoding, the feature dimension reverses from $n \times (3+c)$ to $n \times 3$, in order to recover the same size as the input. To determine the exact correspondences between the two point clouds, we calculate and minimize the point-to-plane distance, which is the sum of the squared distance between each point in the first point set and the tangent plane at the second point set, to find the closest matching. Therefore, we are able to build the self-supervised losses by formulating the scene flow estimation as a 3D reconstruction problem. Assuming the first input point cloud is P_{src} and the second input point cloud is P_{tgt}, the predicted scene flow between the point clouds is $\widetilde{F}$ and thus the synthesized second point cloud $\widetilde{P}_{tgt}$ can be expressed as $\widetilde{P}_{tgt} = P_{src} + \widetilde{F}$. We expect the vector of Laplacian coordinates of the synthesized point cloud should be consistent with the source point cloud. The differential coordinate vector can be written as: $\delta(p_i) = \frac{1}{d_i} \sum_{j \in d_i} (p_j - p_i)$, where $\delta(p)$ is the Laplacian coordinate vector at point p. d is a set of points near the center vertex. To build a consistency between the two local shape characteristics of the surface, as a rough alignment in shape, the reconstructed shape regularization term is defined as:

$$L_{shape} = \sum_{l} \sum_{m \in P_{tgt}, n \in \widetilde{P}_{tgt}} \|\delta(p_m) - \delta(q_n)\|^2 \tag{4}$$

In addition to the shape consistency term, the reconstructed Chamfer loss is further introduced, as a fine adjustment, to best preserve the overall structure and point distribution at each level l, it can be expressed as:

$$\begin{aligned} L_{chamfer} = \sum_{l} (& \sum_{x \in P^l_{tgt}} min \left\| P^l_{tgt} - \widetilde{P}^l_{tgt} \right\|^2 + \\ & \sum_{y \in \widetilde{P}^l_{tgt}} min \left\| P^l_{tgt} - \widetilde{P}^l_{tgt} \right\|^2) \end{aligned} \tag{5}$$

The overall self-supervised constraint on two point sets is a weighted sum of the two proposed reconstructed losses across every level: $L_{self} = \lambda_1 L_{chamfer} + \lambda_2 L_{shape}$.

3. EXPERIMENTS

To evaluate our models, we test on both synthetic ModelNet40 dataset and real-world KITTI Scene Flow dataset, with compairsons with other state-of-the-art methods. An ablation study is conducted to analyze the contribution of each key component.

3.1. Dataset and setup

For training, we select the KITTI scene flow dataset [17] [18], which is originally used to evaluate of stereo based algorithms. To generate point clouds, We randomly sample n points from each frame in a non-corresponding manner, which means there are no point-to-point correspondences between the points for the first frame and the points sampled from the second frame. To evaluate the performance of 3D scene flow calculation, we use the sampled point clouds from adjacent frames, remove the ground points and apply 150 point clouds of all the 200 point clouds for training, since there is no ground truth in the test split.

We also test on the ModelNet40 [19] dataset, which consists of totally 12311 3D CAD models of 40 categories. We utilize the official 9843 shapes for training and the rest 2468 for testing. To generate the scene flow, following the similar setting as in the KITTI dataset, we randomly transform the raw point cloud with a rotation degree in the range of [0, 45] and a translation ranging from -0.5 and 0.5, sampled the raw and transformed point clouds in a non-corresponding manner, and take the sampled point clouds together as our input each time.

During the training phase, we build a four-level feature pyramid in the graph network from the two input point clouds. The weights for λ_1 and λ_2 are set to be 1.0 and 0.25 based on testing. We train our model starting from a learning rate of 0.001 and reducing it to half every 60 epochs. Each point cloud is sampled to 2048 points for efficient and fast learning. For evaluation, we adopt 3D average End-Point-Error (EPE) and flow estimation accuracy (ACC) to restrict the percentage of points with a relative error less than 5% as our evaluation metrics and compare with ground truth scene flow and other recent algorithms.

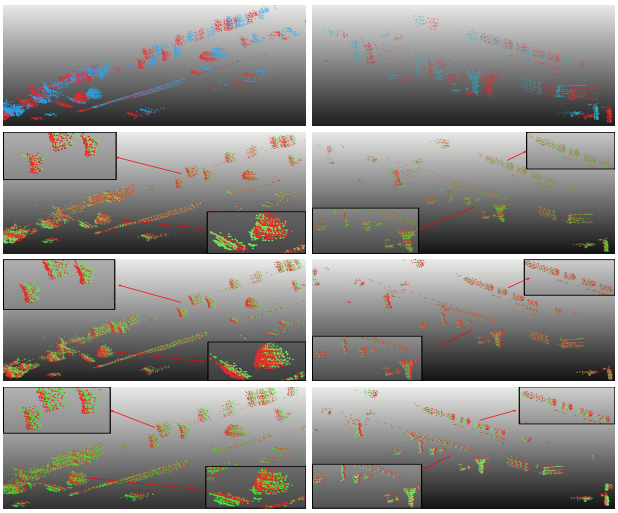

Fig. 2. Results on the KITTI Scene Flow dataset. In each sample column, first to last row: 2 input point clouds PC1 and PC2 are drawn in Blue and Red, respectively; Our predicted PC2 (green) with the estimated scene flow and ground truth PC2 (red); Same for [20] (3rd row) and [10] (4th row).

Method	Input	EPE3D	ACC
ICP [20]	points	0.385	57.6%
Dewan et al. [6]	points	0.587	28.3%
Splatnet [21]	points	0.147	84.7%
FlowNet3D [10]	points	0.122	**94.4%**
Ours **w/o** dynamic updating	points	0.109	90.3%
Ours **w/o** attention mechanism	points	0.117	86.9%
Ours **w/o** surface regularization	points	0.120	87.5%
Ours full	points	**0.102**	92.6%

Table 1. Quantitative result on the KITTI Scene Flow dataset. For error metric 3D End-point-error (EPE3D), lower is better. For accuracy metric ACC, the higher the better.

3.2. Experimental results

Table 1 and Table 2 report the scene flow estimation results on KITTI Scene Flow and ModelNet40 dataset, respectively. We first compared with ICP algorithm [20] and scene flow tracking [6]. It can be observed our algorithm achieves much higher performance than these two methods in KITTI dataset because they are more suitable for rigid transform but not well adapted to moving scenes or objects, especially for [6]. Compared with deep learning based supervised methods Splatnet [21] and FlowNet3D [10], our pipeline still shows slightly better performance especially for EPE3D (in meters). The reason can be explained as that the proposed network is able to learn better the represented features from each point group and relationships between both point clouds efficiently benefiting from the dynamic updating and attention mechanism.

Table 1 and Table 2 also provide the effects of each key component in the proposed 3D SceneFlowNet. Comparing with different settings without one component at each time (w/o dynamic updating, attention and shape regularization), our proposed full pipeline gets the best performance, with (5.7% accuracy improvement compared to the setting w/o attention mechanism on KITTI and 9.6% accuracy improvement on ModelNet40). Figure 2 and 3 provide visual results of our scene flow prediction together with the results from other classic and deep learning methods. We can see that compared with [20] and [10], our network can recover the 3D flows for dynamic objects and small details more accurately, while the compared methods only roughly obtain a rigid flow for the entire point cloud, especially for [20].

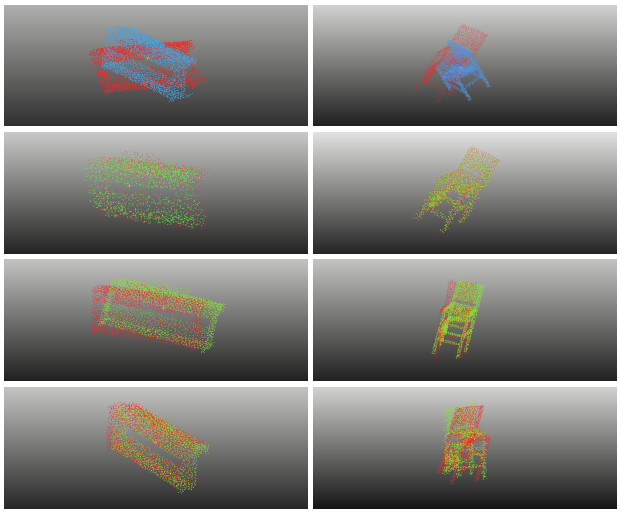

Fig. 3. Results on the ModelNet40 dataset. In each sample column, first to last row: 2 input point clouds PC1 and PC2 are drawn in Blue and Red, respectively; Our predicted PC2 (green) with the estimated scene flow and ground truth PC2 (red); Same for [20] (3rd row) and [10] (4th row).

Method	Input	EPE3D	ACC
ICP [20]	points	0.210	63.9%
Dewan et al. [6]	points	0.463	39.3%
Splatnet [21]	points	0.076	88.0%
FlowNet3D [10]	points	0.054	**94.3%**
Ours **w/o** dynamic updating	points	0.041	91.7%
Ours **w/o** attention mechanism	points	0.047	84.5%
Ours **w/o** surface regularization	points	0.050	86.4%
Ours full	points	**0.036**	94.1%

Table 2. Quantitative result on the ModelNet40 dataset. For error metric 3D End-point-error (EPE3D), lower is better. For accuracy metric ACC, the higher the better.

4. CONCLUSION

This paper presents a novel graph-based self-supervised 3D scene flow estimation approach **3D SceneFlowNet**, which benefits from the multi-level dynamic point learning to better extract features from single point group and multi-level attention mechanism to better learn the relationships between different point clouds. To estimate the scene flow in 3D space without supervision, we propose the EdgeDeConv operator and the two types of 3D reconstruction losses. Our approach achieves extraordinary performance in both accuracy and error metrics on two different datasets.

5. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Award No. 2104032.

6. REFERENCES

[1] Deqing Sun, Erik B Sudderth, and Hanspeter Pfister, "Layered rgbd scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 548–556.

[2] Degui Xiao, Qiuwei Yang, Bing Yang, and Wei Wei, "Monocular scene flow estimation via variational method," *Multimedia Tools and Applications*, vol. 76, no. 8, pp. 10575–10597, 2017.

[3] Fabian Brickwedde, Steffen Abraham, and Rudolf Mester, "Mono-sf: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes," in *IEEE International Conference on Computer Vision*, 2019, pp. 2780–2790.

[4] Junhwa Hur and Stefan Roth, "Self-supervised monocular scene flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7396–7405.

[5] Arash K Ushani, Ryan W Wolcott, Jeffrey M Walls, and Ryan 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)*. IEEE, 2017, pp. 5666–5673.

[6] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard, "Rigid scene flow for 3d lidar scans," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1765–1770.

[7] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[8] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in neural information processing systems*, 2018, pp. 820–830.

[9] 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)*, vol. 38, no. 5, pp. 1–12, 2019.

[10] Xingyu Liu, Charles R Qi, and Leonidas J Guibas, "Flownet3d: Learning scene flow in 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 529–537.

[11] 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 *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[12] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[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," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3254–3263.

[14] Charles Ruizhongtai 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*, 2017, pp. 5099–5108.

[15] Yuxin Wu and Kaiming He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[17] Moritz Menze, Christian Heipke, and Andreas Geiger, "Joint 3d estimation of vehicles and scene flow.," *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 2, 2015.

[18] Moritz Menze, Christian Heipke, and Andreas Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 60–76, 2018.

[19] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015, pp. 1912–1920.

[20] Paul J Besl and Neil D McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*. International Society for Optics and Photonics, 1992, vol. 1611, pp. 586–606.

[21] 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*, 2018, pp. 2530–2539.