Just Go with the Flow: Self-Supervised Scene Flow Estimation - Workshop Overview

Himangi Mittal, Brian Okorn, David Held

***Abstract*—When interacting with highly dynamic environments, scene flow allows autonomous systems to reason about the non-rigid motion of multiple independent objects. Current state-of-the-art methods require annotated scene flow data from autonomous driving scenes to train scene flow networks with supervised learning. As an alternative, we present a method of training scene flow that uses two self-supervised losses, based on nearest neighbors and cycle consistency. These self-supervised losses allow us to train our method on large unlabeled autonomous driving datasets; the resulting method matches current state-of-the-art supervised performance using no real world annotations and exceeds state-of-the-art performance when combining our self-supervised approach with supervised learning on a smaller labeled dataset.**

I. Introduction

For an autonomous vehicle, understanding the dynamics of the surrounding environment is critical to ensure safe planning and navigation. It is essential for a self-driving vehicle to be able to perceive the actions of various entities around it, such as other vehicles, pedestrians, and cyclists. In the context of data recorded as 3D point clouds, a motion can be estimated for each 3D point; this is known as scene flow, which refers to the 3D velocity of each 3D point in a scene.

Recent state-of-the-art methods learn to estimate the scene flow from 3D point clouds [4, 3, 10, 12]. However, these methods are fully supervised and require annotated datasets for training. Such annotations are costly to obtain as they require labeling the motion for every point in a scene. To compensate for the lack of real world data, learning-based methods for scene flow have been trained primarily on synthetic datasets. This requirement of labeled training data limits the effectiveness of such methods in real world settings.

To overcome this limitation, we propose a self-supervised method for scene flow estimation. Using a combination of two self-supervised losses, we are able to mimic the supervision created by human annotation. Specifically, we use a cycle consistency loss, which ensures that the scene flow produced is consistent in time (*i.e.* we ensure that a temporal cycle ends where it started). We also use a nearest neighbor loss, considering the nearest point to the predicted translated point, in the temporally next point cloud, as the pseudo-ground truth association. We show that this combination of losses can be used to train a scene flow network over large-scale, unannotated datasets containing sequential point cloud data. An overview of our method can be found in Figure 1.

We test our self-supervised training approach using the neural network architecture of a state-of-the-art scene flow method [4]. The self-supervision allows us to train this network on large-scale, unlabeled autonomous driving datasets. Our method matches the current state-of-the-art performance when no real world annotations are given. Moreover, our method exceeds the performance of state-of-the-art scene flow estimation methods when combined with supervised learning on a smaller labeled dataset.

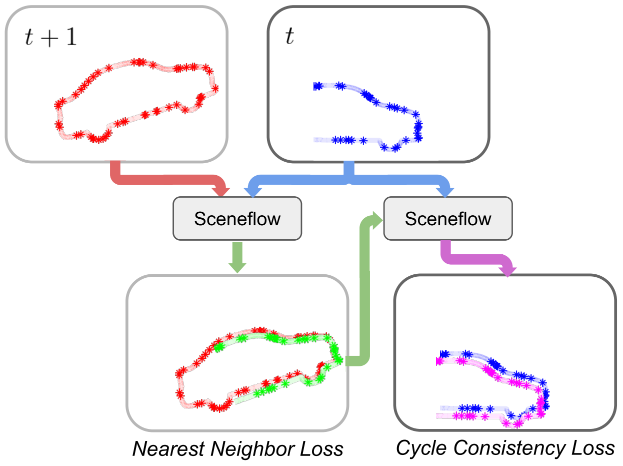


Fig. 1: We use two self-supervised losses to learn scene flow on large unlabeled datasets. The “nearest neighbor loss” penalizes the distance between the predicted point cloud (green) and each predicted point’s nearest neighbor in the second point cloud (red). To avoid degenerate solutions, we also estimate the flow between these predicted points (green) in the reverse direction back to the original point cloud (blue) to form a cycle. The new predicted points from the cycle (purple) should align with the original points (blue); the distance between these two set of points forms our second self-supervised loss: “cycle consistency.”

II. Related Work

Scene Flow: Vedula *et al.* [9] first introduced the task of scene flow estimation. They propose a linear algorithm to compute it from optical flow. State-of-the-art scene flow methods today use deep learning to improve performance. FlowNet3D [4] builds on PointNet++ [8, 7] to estimate scene flow directly from a pair of point clouds. Gu *et al.* [3] produced similar results using a permutohedral lattice to encode the point cloud data in a sparse, structured manner.

Self-Supervised Learning: Wang *et al.* [11] used self-supervised learning for 2D tracking on video. They propose a tracker which takes a patch of an image at time t and the entire image at time $t-1$ to track the image patch in the previous frame. They define a self-supervised loss by tracking the patch forward and backward in time to form a cycle while penalizing the errors through cycle consistency and feature similarity. We take inspiration from this work for our self-supervised flow estimation on point clouds. Concurrent to our work, Wu *et al.* [12] showed that Chamfer distance, smoothness constraints, and Laplacian regularization can be used to train scene flow in a self-supervised manner.

III. Method

A. Problem Definition

For the task of scene flow estimation, we have a temporal sequence of point clouds: point cloud $\mathcal{X}$ as the point cloud captured at time t and point cloud $\mathcal{Y}$ captured at time $t+1$. Each point $p_i = \{x_i, f_i\}$ in point cloud $\mathcal{X}$ contains the Cartesian position of the point, $x_i \in \mathbb{R}^3$.

The scene flow, $\mathcal{D} = \{d_i\}^N, d_i \in \mathbb{R}^3$ between these two point clouds describes the movement of each point x_i in point cloud $\mathcal{X}$ to its corresponding position x'_i in the scene described by point cloud $\mathcal{Y}$, such that $x'_i = x_i + d_i$, and N is the size of point cloud $\mathcal{X}$. Scene flow is defined such that x_i and x'_i represent the same 3D point of an object moved in time. Unlike optical flow estimation, the exact 3D position of x'_i may not necessarily coincide with a point in the point cloud $\mathcal{Y}$, due to the sparsity of the point cloud, and the sizes of $\mathcal{X}$ and $\mathcal{Y}$ may be different.

Supervised Loss: The true error associated with our task is the difference between the estimated flow $g(\mathcal{X}, \mathcal{Y}) = \hat{\mathcal{D}} = \{\hat{d}_i\}^N$ and the ground truth flow $\mathcal{D}^* = \{d^*_i\}^N$,

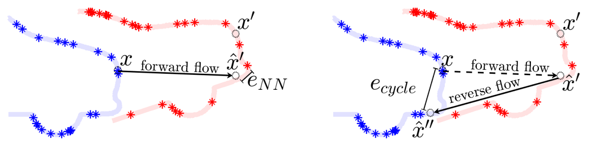
$$\mathcal{L}_{gt} = \frac{1}{N}\sum_{i}^{N} \|d^*_i - \hat{d}_i\|^2. \tag{1}$$

The loss in Equation 1 is useful because it is mathematically equivalent to the end point error, which we use as our evaluation metric. However, computing this loss requires annotated ground truth flow d^*_i. While training on purely synthetic data is possible, large improvements can often be obtained by training on real data from the domain of the target application. For example, Lui *et al.* [4] showed an 18% relative improvement after fine-tuning on a small amount of annotated real world data. This result motivates our work to use self-supervised training to train on large unlabeled datasets.

Nearest Neighbor (NN) Loss: For large unlabeled datasets, since we do not have information about d^*_i, we cannot compute the loss in Equation 1. In lieu of annotated data, we use the nearest neighbor of our transformed point $\hat{x}'_i = x_i + \hat{d}_i$ as an approximation for the true correspondence. For each transformed point $\hat{x}'_i \in \hat{\mathcal{X}}'$, we find its nearest neighbor $y_j \in \mathcal{Y}$ and compute the Euclidean distance to that point, illustrated as e_{NN} in Figure 2a:

$$\mathcal{L}_{NN} = \frac{1}{N}\sum_{i}^{N} \min_{y_j \in \mathcal{Y}} \|\hat{x}'_i - y_j\|^2. \tag{2}$$

Assuming that the initial flow estimate is sufficiently close to the correct flow estimate, this loss will bring the transformed



(a) Nearest Neighbor Loss (b) Cycle Consistency Loss

Fig. 2: Example of our self-supervised losses between consecutive point clouds $\mathcal{X}$ (blue) and $\mathcal{Y}$ (red). (a) Nearest Neighbor Loss is computed between the projected point $\hat{x}'$, predicted by the forward flow, and the closest point in $\mathcal{Y}$. (b) The Cycle Consistency Loss tracks this transformed point back onto its original frame, as point $\hat{x}''$, using the reverse flow, and computes the distance to its original position x.

point cloud and the target point cloud closer. This loss can, however, have a few drawbacks if imposed alone. The true position of the point x_i transformed by the ground truth flow, $x'_i = x_i + d^*_i$, may not be the same as the position of the nearest neighbor to $\hat{x}'$ (transformed by the *estimated* flow) due to potentially large errors in the estimated flow, as illustrated in Figure 2a, and this loss does not penalize degenerate solutions where many of the points in $\mathcal{X}$ map to the same point in $\mathcal{Y}$. To address these issues, we use an additional self-supervised loss: cycle consistency loss.

Cycle Consistency Loss: To avoid the above issues, we incorporate an additional self-supervised loss: cycle consistency loss, illustrated in Figure 2b. We first estimate the "forward" flow as $\hat{\mathcal{D}} = g(\mathcal{X}, \mathcal{Y})$. Applying the estimated flow $\hat{d}_i \in \hat{\mathcal{D}}$ to each point $x_i \in \mathcal{X}$ gives an estimate of the location of the point x_i in the next frame: $\hat{x}'_i = x_i + \hat{d}_i$. We then compute the scene flow in the reverse direction: for each transformed point $\hat{x}'_i$ we estimate the flow to transform the point back to the original frame, $\hat{\mathcal{D}}' = g(\hat{\mathcal{X}}', \mathcal{X})$. Transforming each point $\hat{x}'_i$ by this "reverse" flow $\hat{d}'_i$ gives a new estimated point $\hat{x}''_i$. If both the forward and reverse flow are accurate, this point $\hat{x}''_i$ should be the same as the original point x_i. The error between these points, e_{cycle}, is the "cycle consistency loss," given by

$$\mathcal{L}_{cycle} = \sum_{i}^{N} \|\hat{x}''_i - x_i\|^2. \tag{3}$$

A similar loss is used as a regularization in [4].

However, we found that implementing the cycle loss in this way can produce unstable results if only self-supervised learning is used without any ground-truth annotations. These instabilities appear to be caused by errors in the estimated flow which lead to structural distortions in the transformed point cloud $\hat{\mathcal{X}}'$, which is used as the input for computing the reverse flow $g(\hat{\mathcal{X}}', \mathcal{X})$. This requires the network to simultaneously learn to correct any distortions in $\hat{\mathcal{X}}'$, while also learning to estimate the true reverse flow. To solve this problem, we use the nearest neighbor y_j of the transformed point $\hat{x}'_i$ as an anchoring point in the reverse pass. Using the nearest neighbor y_j stabilizes the structure of the transformed cloud while still maintaining the correspondence around the cycle.

Specifically, we compute the anchored reverse flow as follows. First, we compute the forward flow as before, $\hat{\mathcal{D}} = g(\mathcal{X}, \mathcal{Y})$, which we use to compute the transformed point cloud $\hat{x}_i' = x_i + \hat{d}_i$. We then compute anchor points $\bar{\mathcal{X}}' = \{\bar{x}_i'\}^N$ as a convex combination of the transformed point and its nearest neighbor $\bar{x}_i' = \lambda \hat{x}_i' + (1 - \lambda) y_j$. In our experiments, we find that $\lambda = 0.5$ produces the most accurate results. Finally, we compute the reverse flow using these anchored points: $\bar{\mathcal{D}}' = g(\bar{\mathcal{X}}', \mathcal{X})$. The cycle loss of Equation 3 is then applied to this anchored reverse flow. By using anchoring, some of the structural distortion of the predicted point cloud $\hat{\mathcal{X}}'$ will be removed in the anchored point cloud $\bar{\mathcal{X}}'$, leading to a more stable training input for the reverse flow.

IV. Experiments

We run several experiments to validate our self-supervised method for scene flow estimation for various levels of supervision and different amounts of data. First, we show that our method, with self-supervised training on large unlabeled datasets, can perform as well as supervised training on the existing labeled data. Next, we investigate how our results can be improved by combining self-supervised learning with a small amount of supervised learning, exceeding the performance of purely supervised learning. Finally, we explore the utility of each element of our method through an ablation study. For all data configurations (our method and the baseline), we initialize our network with the parameters of the Flownet3D model [4] pre-trained on the FlyingThing3D dataset [5]. We compare our self-supervised training procedure to a baseline which uses supervised fine-tuning on the KITTI dataset [2].

A. Datasets

KITTI Vision Benchmark Suite: KITTI [2] is a real-world self-driving dataset. There are 150 scenes of LIDAR data in KITTI collected using seven scans of a Velodyne 64 LIDAR, augmented using 3D models, and annotated with ground truth scene flow [6]. For our experiments under both self-supervised and supervised settings, we consider 100 out of 150 scenes for training and the remaining 50 scenes for testing. Ground points are removed from every scene using the pre-processing that was performed in previous work [4]. To reduce forward motion bias in all training sets, we flip the point clouds with probability 0.5, *i.e.* reversing the flow.

nuScenes: The nuScenes [1] dataset is a large-scale public dataset for autonomous driving. It consists of 850 publicly available driving scenes in total from Boston and Singapore. The LIDAR data was collected using a Velodyne 32 LIDAR. Since the nuScenes dataset [1] does not contain scene flow annotations, we must use self-supervised methods when working with this dataset. In our experiments, out of the 850 scenes available, we use 700 as the train set and the rest 150 as the validation set. Similar to KITTI, we remove the ground points from each point cloud using a manually tuned height threshold, and flip the input point clouds.

B. Results

We use three standard metrics to quantitatively evaluate the predicted scene flow when the ground truth annotations of scene flow are available. Our primary evaluation metric is End Point Error (EPE) which describes the mean Euclidean distance between the predicted and ground truth transformed points, described by Equation 1. We also compute accuracy at two threshold levels, Acc(0.05) as the percentage of scene flow prediction with an EPE $<$ 0.05m or relative error $< 5\%$ and Acc(0.1) as percentage of points having an EPE $<$ 0.1m or relative error $<$ 10%, as was done for evaluation in previous work [4].

Self-supervised training: Unlike previous work, we are not restricted to annotated point cloud datasets; our method can be trained on any sequential point cloud dataset. There are many point cloud datasets containing real LIDAR captures of urban scenes, but most of them do not contain scene flow annotations. Due to lack of annotations, these datasets can not be utilized for supervised scene flow learning. In contrast, our self-supervised loss allows us to easily integrate them into our training set. The combination of these datasets (KITTI + NuScenes) contains 5x more real data than using KITTI alone.

The results are shown in Table I. To show the value of self-supervised training, we evaluate the performance of our method without using any ground-truth annotations. We first pre-train on the synthetic FlyingThings3D dataset; we then perform self-supervised fine-tuning on the large nuScenes dataset followed by further self-supervised fine-tuning on the smaller KITTI dataset (4th row: "Ours: nuScenes (Self-Supervised) + KITTI (Self-Supervised)"). As can be seen, using no real-world annotations, we are able to achieve an EPE of 0.105 m. This outperforms the baseline of only training on synthetic data ("No Fine Tuning"). Even more impressively, our approach performs similarly to the baseline which pre-trains on synthetic data and then does supervised fine-tuning on the KITTI dataset ("KITTI (Supervised)"); our method has a similar EPE and outperforms this baseline in terms of accuracy, despite not having access to any annotated training data. This result shows that our method for self-supervised training, with a large enough unlabeled dataset, can match the performance of supervised training.

Self-supervised + Supervised: We show the value of combining our self-supervised learning method with a small amount of supervised learning. For this analysis, we perform self-supervised training on NuScenes as above, followed by supervised training on the much smaller KITTI dataset. The results are shown in the last row of Table I.

As can be seen, this approach of self-supervised training followed by supervised fine-tuning outperforms all other methods on this benchmark, obtaining an EPE of 0.091, outperforming the previous state-of-the-art result which used only supervised training. This shows the benefit of self-supervised training on large unlabeled datasets to improve scene flow accuracy, even when scene flow annotations are available.

Qualitative Analysis: We perform a qualitative analysis to visualize the performance of our method. We compare our method (synthetic pre-training + self-supervised training on nuScenes + self-supervised training on KITTI) compared to the baseline of synthetic training only. Results on KITTI

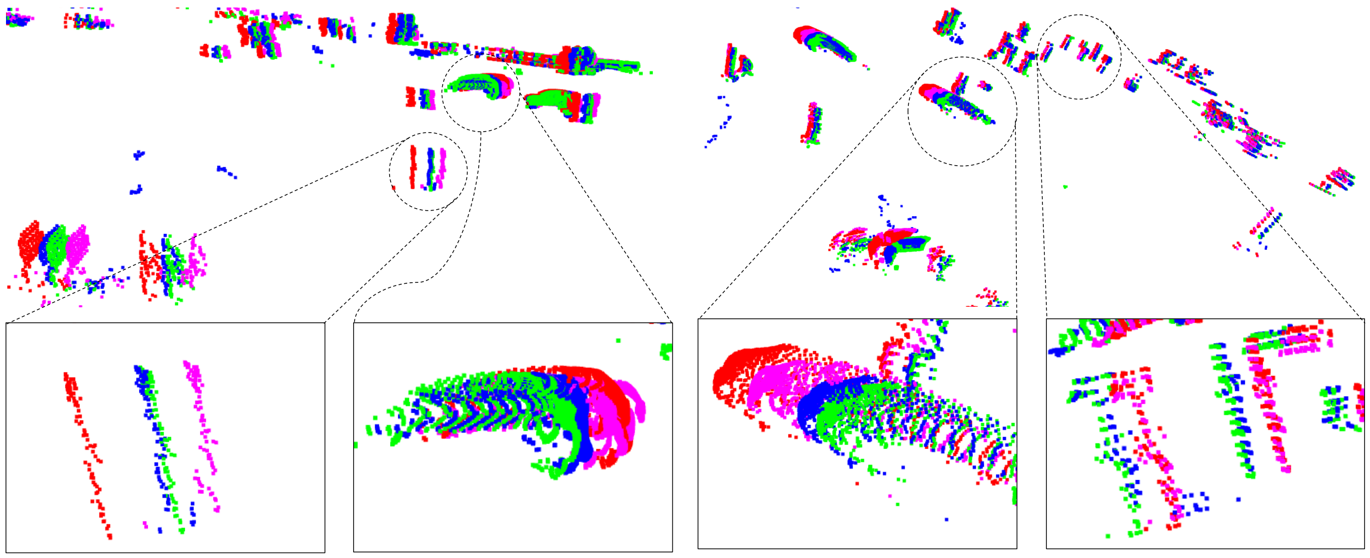


Fig. 3: Scene flow estimation between point clouds at time t (red) and $t+1$ (green) from the KITTI dataset trained without any labeled LIDAR data. Predictions from our self-supervised method, trained on nuScenes and fine-tuned on KITTI using self-supervised learning is shown in blue; the baseline with only synthetic training is shown in purple. In the absence of real-world supervised training, our method clearly outperforms the baseline method, which overestimate the flow in many regions.

Training Method	EPE (m) ↓	ACC (0.05) ↑	ACC (0.1) ↑
No Fine Tuning	0.122	25.37%	57.85%
KITTI (Supervised)	0.100	31.42%	66.12%
Ablation: KITTI (Self-Supervised)	0.126	32.00%	73.64%
Ours: nuScenes (Self-Supervised) + KITTI (Self-Supervised)	0.105	46.48%	79.42%
Ours: nuScenes (Self-Supervised) + KITTI (Supervised)	**0.091**	**47.92%**	**79.63%**

TABLE I: Comparison of levels of supervision on KITTI dataset. The nearest neighbor + anchored cycle loss is used for nuScenes (self-supervised) and KITTI (self-supervised). All methods are pretrained on FlyingThings3D[5] and ground points are removed for KITTI and nuScenes datasets.

NN Loss	Cycle Loss	Anchor	EPE (m)↓	ACC (0.05)↑	ACC (0.1)↑
	✓	✓	0.1572	18.50	52.80
✓		✓	0.1090	34.88	71.32
✓	✓		0.0932	28.18	66.10
✓	✓	✓	**0.0912**	**47.92**	**79.63**

TABLE II: We study the effect of removing a single component of self-supervised loss and data augmentation. Models use self-supervised training on nuScenes and KITTI.

are shown in Figure 3. The figure shows the point clouds captured at time t and $t+1$ in red and green, respectively. The predictions from our method are shown in blue and the baseline predictions are shown in purple. As shown, our scene flow predictions (blue) have a large overlap with the point cloud at time $t+1$ (green). On the other hand, the baseline predictions (purple) do not overlap with the point cloud at time $t+1$. The baseline, trained only on synthetic data, fails to generalize to the real-world KITTI dataset. On the contrary, our self-supervised approach can be fine tuned on any real world environment and shows a significant improvement over the baseline.

Ablation Studies: We test the importance of each component of our method by running a series of ablation studies. Table II shows the effect of removing a single portions of our method using the purely self-supervised training.

V. Conclusion

In this work, we propose a self-supervised method for training scene flow algorithms using a combination of cycle consistency in time and nearest neighbor losses. Our purely self-supervised method is able to achieve a performance comparable to that of the supervised methods on the widely used KITTI self-driving dataset. We further show that when supervised training is augmented with self-supervision on a large-scale, unannotated dataset, the results exceed the current state-of-the-art performance. Our self-supervision method opens the door to fine-tuning on arbitrary datasets that lack scene flow annotations. Please see our full paper for more information [1].

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint*

[1]https://arxiv.org/abs/1912.00497

arXiv:1903.11027, 2019. URL https://arxiv.org/pdf/1903.11027.pdf.

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[3] 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 *CVPR*, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/papers/Gu_HPLFlowNet_Hierarchical_Permutohedral_Lattice_FlowNet_for_Scene_Flow_Estimation_on_CVPR_2019_paper.pdf.

[4] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/papers/Liu_FlowNet3D_Learning_Scene_Flow_in_3D_Point_Clouds_CVPR_2019_paper.pdf.

[5] 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 *CVPR*, 2016. URL http://openaccess.thecvf.com/content_cvpr_2016/papers/Mayer_A_Large_Dataset_CVPR_2016_paper.pdf.

[6] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140, 2018.

[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*, pages 652–660, 2017. URL http://openaccess.thecvf.com/content_cvpr_2017/papers/Qi_PointNet_Deep_Learning_CVPR_2017_paper.pdf.

[8] 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*, pages 5099–5108, 2017. URL https://arxiv.org/pdf/1706.02413.pdf.

[9] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV*, 1999.

[10] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/papers/Wang_Deep_Parametric_Continuous_CVPR_2018_paper.pdf.

[11] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_Learning_Correspondence_From_the_Cycle-Consistency_of_Time_CVPR_2019_paper.pdf.

[12] 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. 2019. URL https://arxiv.org/pdf/1911.12408.pdf.