3D Object Detection with a Self-supervised Lidar Scene Flow Backbone

Emeç Erçelik[1*], Ekim Yurtsever[2*], Mingyu Liu[1,3], Zhijie Yang[1], Hanzhen Zhang[1], Pınar Topçam[1], Maximilian Listl[1], Yılmaz Kaan Çaylı[1], and Alois Knoll[1]

[1] Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University of Munich, 85748 Garching b. München, Germany
{emec.ercelik, mingyu.liu, zhijie.yang, hanzhen.zhang, pinar.topcam, maximilian.listl, kaan.cayl }@tum.de, knoll@in.tum.de
[2] Ohio State University, Columbus, OH 43212, USA
yurtsever.2@osu.edu
[3] Tongji University, 201804, Shanghai, China

Abstract. State-of-the-art lidar-based 3D object detection methods rely on supervised learning and large labeled datasets. However, annotating lidar data is resource-consuming, and depending only on supervised learning limits the applicability of trained models. Self-supervised training strategies can alleviate these issues by learning a general point cloud backbone model for downstream 3D vision tasks. Against this backdrop, we show the relationship between self-supervised multi-frame flow representations and single-frame 3D detection hypotheses. Our main contribution leverages learned flow and motion representations and combines a self-supervised backbone with a supervised 3D detection head. First, a self-supervised scene flow estimation model is trained with cycle consistency. Then, the point cloud encoder of this model is used as the backbone of a single-frame 3D object detection head model. This second 3D object detection model learns to utilize motion representations to distinguish dynamic objects exhibiting different movement patterns. Experiments on KITTI and nuScenes benchmarks show that the proposed self-supervised pre-training increases 3D detection performance significantly. https://github.com/emecercelik/ssl-3d-detection.git

Keywords: 3D detection, self-supervised learning, scene flow, lidar point clouds

1 Introduction

Lidar promises accurate distance measurement, which is crucial for real-time systems such as 3D perception modules of automated vehicles. Supervised learning methods have dominated benchmarks created for challenging downstream 3D vision tasks [5,12,40,9]. However, high-performing models need a copious amount

* Authors contributed equally.

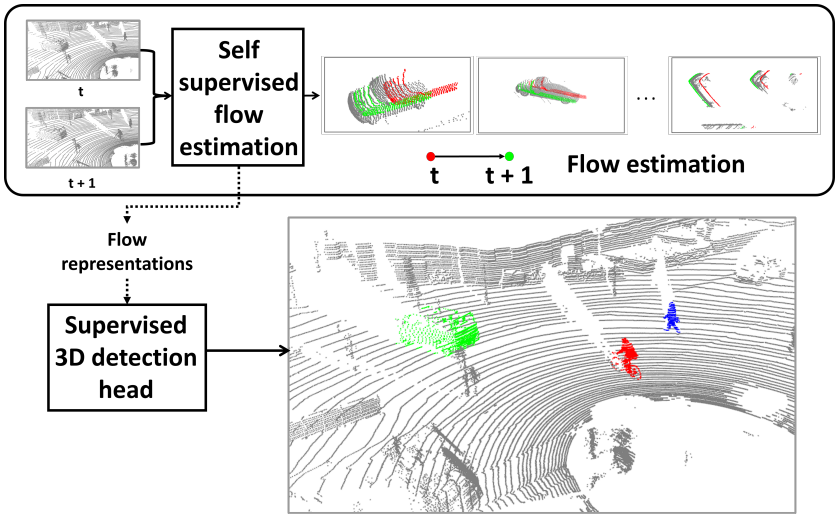

Fig. 1. This study shows the relationship between self-supervised flow representations and supervised 3D detection hypotheses. We illustrate the importance of defining 3D objects-of-interest hypotheses in a spatio-temporal context. For example, a car should not just be defined by its shape but also by its capability to move in space and time. To this end, a scene flow estimation network is trained with cycle consistency in a self-supervised manner. Then, the backbone of this pre-trained model is used to feed a supervised 3D object detection head. The proposed strategy improves detection performance significantly compared to baselines when the amount of labeled data is less for supervised learning.

of labeled data for training. Annotating lidar data is labor-intensive and is a bottleneck for real-world deployment.

Recent work showed the importance of self-supervised learning to build large backbones by exploiting the structure of data. For example, the temporal contextual changes in videos can be exploited in contrastive learning strategies [41]. Contrastive methods have also been used with data augmentation [31] for similar purposes. MoCo [16] classifies images in binary form as positive and negative to learn useful representations. Another approach is to quantize representations from a teacher network [6], [1]. However, these works focus solely on the RGB image domain. Not much work focuses on unsupervised or self-supervised 3D object detection. Point cloud sparsity poses additional challenges, as the structure of data is significantly different from denser modalities.

The main body of state-of-the-art 3D object detection with point cloud literature comprises supervised learning methods [62,55,15,37,56,38,57,39,10,22,58].

Point cloud scene flow is another important 3D vision task. Initially, supervised learning was shown to be superior for the task [45,26,14,52,25,51,32]. More recently, self-supervised and unsupervised 3D scene flow and stereo flow methods have been introduced [49,2,19,60,64,21,7,13,34,50,28]. However, these

developments have not been utilized for the 3D object detection task up until now.

We propose to employ a scene flow backbone trained in a self-supervised fashion to learn useful representations that other downstream head models can utilize after fine-tuning, such as a 3D object detection head (Fig. 1). First, we follow the cycle-consistency approach [30] to train a FlowNet3D-based [25] scene-flow backbone using self-supervised learning. We introduce architectural changes to the FlowNet3D module to incorporate a point cloud backbone that can also be utilized with a detection head. We explore several training and loss strategies, including auxiliary training, to find the best layout. Empirical evidence obtained from KITTI [12] and nuScenes [5] datasets show that the proposed strategy increases 3D detection performance significantly.

Our method differs from [30] in two important ways. To the best of our knowledge, lidar point cloud-based 3D object detection has not been successfully achieved with a self-supervised backbone up until now. Our modifications to the FlowNet3D [25] architecture enables the integration of point-level temporal changes with 3D detection. Secondly, our combined auxiliary training cycle consistency and supervised 3D detection losses lead to learning more general representations as well as motion representations, which identify objects based on their contextual motion patterns.

A summary of our main contributions is as follows:

- Employing self-supervised point cloud scene flow estimation to learn motion representations for 3D object detection in tandem with supervised fine-tuning
- We show that auxiliary training is the best strategy for using self-supervised cycle-consistency loss along with supervised 3D detection loss.
- The proposed strategy is especially effective with a lesser amount of supervised data. We obtained a significant performance boost when only a smaller part of supervised training data was used for the 3D detection task.

2 Related Work

Scene flow. Scene flow was first introduced as an extension of optical flow in the third dimension and was estimated with a linear computation algorithm[42]. Stereo cameras [18,43] and RGB-D were also [36] utilized to derive scene flow. Current state-of-the-art uses lidar point clouds and deep neural networks to estimate scene flow with supervised learning [25,51,14]. Most commonly, two subsequent lidar frames are used to estimate the flow vectors of each point in the scene. Building ground truth for such vectors is labor-intensive. As such, synthetic datasets are more popular for scene flow benchmarking [32].

Self-supervised scene flow estimation. Self-supervised scene flow estimation is a relatively understudied angle. A recently proposed solution [30] is to use cycle-consistency and nearest neighbors losses to train an estimation network. Several other distance metrics and regularization techniques such as using

chamfer distance, smoothing, and regularization [53] have also been employed for the same task. A more recent study showed that self-supervised scene flow could also be combined with motion segmentation [3].

Self-supervised 3D Object Detection. A monocular 3D object detection model has been trained with self-supervised learning using shape priors and 2D instance masks [4]. Another monocular 3D object detection model with weak supervision has been trained using shape priors [48]. [35] generates random synthetic point cloud scenes for pre-training to learn useful representations from CAD models. [54,23,17] mainly use contrastive pre-training to learn geometrical point cloud representations with different views of the same scene. However, there is not much work focusing on self-supervised 3D object detection considering motion representations with point clouds. We aim to fill this gap in the literature.

3 Method

Backbone of a 3D object detector is mainly used to extract point, voxel, or region features to detect possible objects in that vicinity. Due to the limitation in labelled dataset sizes, we aim to train the backbone on a large unlabelled dataset using self-supervision to obtain good motion-aware point representations. Afterwards, it is possible to use the pre-trained backbone for the 3D detection supervised training with a smaller dataset. Thus, the 3D detection network can benefit from the initialized point motion representations to distinguish objects based on movement patterns. We summarize our method in Fig. 2.

3.1 Problem Definition

Given two subsequent lidar point cloud frames $\mathcal{P}_t = \{\mathbf{p}_i\}_M, \mathbf{p}_i \in \mathbb{R}^3$ and $\mathcal{P}_{t+1} \in \mathbb{R}^{N\times 3}$, we are first interested in estimating the scene flow $\mathcal{F}_{t\to t+1} = \{d_i\}_M$, where $\mathbf{d}_i = \mathbf{p}'_i - \mathbf{p}_i$. $\mathbf{p}'_i$ denotes the new position at time $t+1$ of point i in the first point cloud $\mathcal{P}_t$. It should be noted that the second point cloud may or may not contain a point corresponding to $\mathbf{p}'_i$ due to sparsity. The second objective is to map $\mathcal{P} \to \{T_j\}_U$, where T_j is the 3D object detection tuple containing class id and bounding box shape and coordinates, using previously-learned spatio-temporal representations. U is the total number of objects in the point cloud frame.

We aim to benefit from the point motion representations learned by the 3D feature extractor, g, during self-supervised scene flow training $\mathcal{F}_{t\to t+1} = s(g(\mathcal{P}_t, \mathcal{P}_{t+1}))$, where s is the scene flow head. In this way, the 3D detection head, h, can use the spatio-temporal motion representations learned in g to better identify complex object point patterns for meaningful detection results such that $\{T_j\}_U = h(g(\mathcal{P}))$.

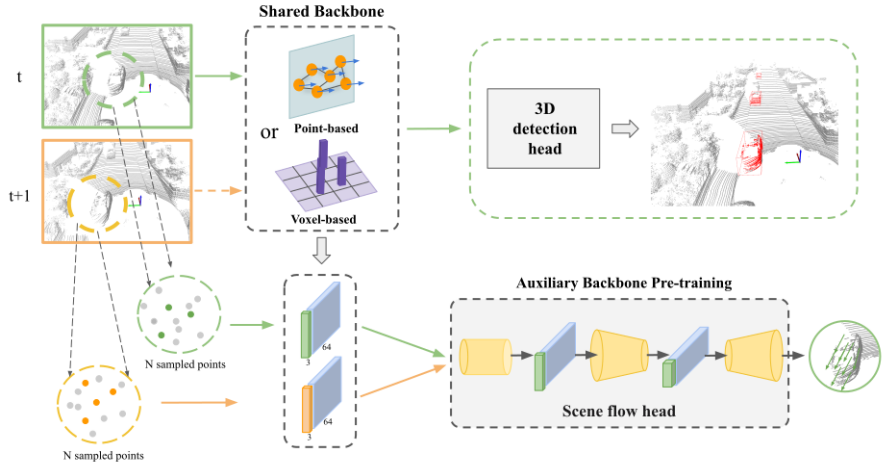

Fig. 2. Our self-supervised 3D object detection pre-training: The auxiliary scene flow head is used to train the 3D detection backbone (point- or voxel-based) for motion-aware point cloud representations with self-supervised cycle consistency loss [30]. The motion representations learned without labelled data can help distinguish objects based on their motion patterns for a 3D downstream task. Then, we further train the pre-trained backbone and a 3D detection head for 3D detection with labelled data.

3.2 Self-supervised Scene Flow

Backbone: We first follow the cycle-consistency approach [30] to train a scene flow estimator. We use a 3D detector's backbone to extract local features of sampled points from two consecutive frames. This allows the self-supervised scene flow gradients to be backpropagated through the backbone. Hence, the backbone learns point representations encoding object movement patterns. The learned spatio-temporal features can be further used to distinguish objects from the background and other objects for the 3D perception task.

Scene Flow Head: The scene flow head based on FlowNet3D [25] generates flow embeddings from local point features provided by the 3D backbone. The shape of input points, $\mathcal{P}_t$, is reconstructed by applying *set upconv layers* to local flow embeddings for the final scene flow estimations $\mathcal{F}_{t\to t+1}$.

Training with Cycle Consistency: We use 3D detection backbone as the feature extractor for the scene flow head. Both the backbone and the scene flow head are trained with the self-supervised cycle consistency loss given in [30]. For the cycle consistency, the scene flow is calculated in forward and backward directions, meaning $\mathcal{F}_{t\to t+1}$ and $\mathcal{F}_{t+1\to t}$. The $\mathcal{F}_{t+1\to t}$ makes use of the new positions of the propagated points $\mathbf{p}'_i$ to close the cycle. The $\mathbf{p}''_i$ is the estimated positions of the $\mathbf{p}_i$ in the backward direction with the $\mathcal{F}_{t+1\to t}$. The mismatch between the $\mathbf{p}_i$ and the $\mathbf{p}''_i$ at frame t allows training of the backbone in a

self-supervised way. With the self-supervised training, 3D backbone learns to generate regional flow and motion features from the given set of point clouds.

3.3 Downstream Task: 3D Object Detection

We are interested in the 3D object detection as the 3D downstream task. The scene flow head, s, and the 3D object detection head, h, use the same backbone, g, as seen in Fig. 2. Also, point- or voxel-based 3D backbone encodings can be used. We initialize 3D detector's backbone weights with the pre-trained weights from the auxiliary self-supervised scene flow training. In this way, we assume that the pre-trained backbone from scene flow can already provide good geometry- and motion-aware point features. The 3D detection head takes distinguishable spatio-temporal point cloud features based on different object motion patterns. Hence, the 3D detection network can detect objects more accurately even after supervised training with a smaller labelled dataset. We show the efficacy of our approach in section 5. Note that the scene flow head is for the auxiliary scene flow training and is not used during the 3D detection training and inference.

4 Implementation Details

Our self-supervised auxiliary backbone pre-training approach can be used with different 3D detector architectures. We evaluate our method with mainly three different 3D detectors, Point-GNN[4][39], CenterPoint[5][59], and PointPillars[5][22], which are point-, voxel-based approaches. For the self-supervised scene flow task, we add the modified FlowNet3D as well as the cycle-consistency loss[6] [30] to 3D detectors' training pipelines.

4.1 Pre-training with Self-supervised Scene Flow

The FlowNet3D takes a set of points from two successive frames as input and estimates the flow vectors. The network extracts the point features with two cascaded *PointNet Set Abstraction* modules, each with a 3-layer MLP. We remove the first *PointNet Set Abstraction* module and feed in the point features from 3D detector's backbone to the second *PointNet Set Abstraction* module.

Point cloud backbone: We use Point-GNN, PointPillars, and CenterPoint backbones for our main and ablation results. The CenterPoint and PointPillars backbones have the same architecture as we use mmdetection3d [8] implementations. Point-GNN extracts keypoint features from a 3-level graph network used as a backbone, from which we obtain keypoint features of two consecutive point clouds. After sampling N points from each frame, we apply bilinear interpolation to get features of the sampled points from keypoint features according to their

[4] https://github.com/WeijingShi/Point-GNN
[5] https://github.com/open-mmlab/mmdetection3d/
[6] https://github.com/HimangiM/Just-Go-with-the-Flow-Self-Supervised-Scene-Flow-Estimation

positions. We use the settings provided for the best performing Point-GNN with $T = 3$, which represents the number of graph levels. For the CenterPoint and PointPillars detectors, we follow a similar approach and use their voxel encoders to obtain features of the sampled points from two consecutive frames without making any changes to the 3D detector's architecture.

Scene flow head: Scene flow head is responsible for estimating 3D motion of the points between two sequential frames. The FlowNet3D's scene flow head consists of *flow embedding*, *set conv layers*, and *set upconv layers* followed by fully-connected layers for estimating point flow vectors. The scene flow head takes the local point features as inputs. We remove only the final *set upconv layer* that takes skip connections from the first *PointNet Set Abstraction* module, which we replace with the 3D detector's backbone.

Training strategy: We train the point cloud backbone and scene flow head end-to-end using the self-supervised scene flow loss. For the scene flow training on Point-GNN backbone, we initialize our scene flow head weights with the pre-trained FlowNet3D weights on the supervised FlyingThings3D [29] simulation data, we use Stochastic Gradient Descent (SGD) optimizer with 6.25×10^{-5} learning rate. We train the scene flow network for 80k steps on the KITTI tracking dataset without using any labels. The model is trained on a single Nvidia Tesla V100 GPU. We train the PointPillars scene flow network on KITTI tracking dataset using optimizer AdamW with 0.001 learning rate. The training is done with batch size 2 on one Nvidia RTX 2080 GPU for 10 epochs. For the PointPillars- and CenterPoint-based scene flow training on nuScenes dataset, we use AdamW optimizer using the voxel encoders as the bakcbone with a 0.001 learning rate. Our batch size is 2. We train the network for 4 epochs on one Nvidia RTX 3090 GPU. For all networks, we sample $N = 2048$ points per frame.

4.2 3D Detection Fine-tuning

3D detection heads: We use Point-GNN, CenterPoint, and PointPillars as the 3D detectors for our results to show the efficacy of our self-supervised scene flow pre-training approach. We initialize detectors' backbone weights with the pre-trained backbone weights from the auxiliary scene flow task for a better point feature representation.

Training strategy: After initializing weights of the 3D detector backbone from the scene flow task, we further train the backbone and the 3D detection heads with the 3D detection loss. We apply an alternating training strategy between the self-supervised scene flow and supervised 3D detection trainings: **(i)** Train the backbone and the scene flow head for self-supervised scene flow, **(ii)** train the pre-trained backbone and the detection head with 3D detection training, **(iii)** train the backbone from step (ii) and the scene flow head from step (i) for the scene flow, and finally **(iv)** train the backbone from step (iii) and the detection head from step (ii) for 3D detection.

We train the Point-GNN baseline for $1400k$ steps using the SGD optimizer with a learning rate of 0.125 as done in the Point-GNN paper. For the training in step **(ii)** and step **(iv)**, we use SGD with a learning rate of 0.1. The trainings

took place on an Nvidia Tesla V100 GPU. We use batch size 4 for all Point-GNN trainings. All the PointPillars and CenterPoint detectors training are on one RTX3090 GPU. For PointPillars on KITTI, we set batch size 18 using AdamW as optimizer with learning rate 0.001. The PointPillars detector on nuScenes is trained for 24 epochs with a learning rate of 0.001 using AdamW optimizer. We use batch size 4 for the detection training. The training of CenterPoint detector is trained for 20 epochs with batch size 20 and the learning rate of 0.001 using the AdamW optimizer.

4.3 Datasets

We use KITTI 3D Object Detection, KITTI Multi-object Tracking datasets [12] as well as nuScenes dataset [5] for the self-supervised scene flow and the supervised 3D detection training and validation.

KITTI 3D Object Detection: KITTI 3D Object Detection dataset consists of 7481 training frames sampled from different drives. Since the provided frames are not sequential, we use lidar point clouds only for 3D object detection training. Only objects visible in the camera-view are annotated. We utilize the common train-val split with 3712 training and 3769 validation samples. For the evaluation, the KITTI average precision (AP) metric is used for three different difficulty levels with $IoU = 0.7$ for the car class.

KITTI Multi-object Tracking: This dataset contains 21 training and 29 testing drives, each of which consists of several sequential frames. We use the tracking dataset only for the self-supervised scene flow training without using any annotations. Therefore, we combine all the training and testing drive data for training except the training drives 11, 15, 16, and 18, which are used for observing cycle consistency validation loss. This gives us 11902 frames for self-supervised scene flow training.

nuScenes: nuScenes is also an autonomous driving dataset, which consists of 700 training and 150 validation drives. The annotations are provided at 2 Hz for 360-degree objects and the lidar sweeps are collected at 20 Hz. nuScenes is a larger dataset than KITTI and it is collected from denser and more challenging environments. There are 10 classes annotated in the nuScenes dataset. The main metrics are the average precision (AP) per class, mean average precision (mAP) among all classes, and the nuScenes detection score (NDS). Since the provided data contains sequential lidar point clouds, we use this dataset for both self-supervised scene flow and 3D detection trainings.

4.4 Loss

For the 3D object detection training, we keep the same loss functions used for the 3D detectors.

Point-GNN[39]: Point-GNN combines localization, classification, and regularization losses. Classification loss is calculated with the average cross-entropy loss among four classes, which are *background*, horizontal and vertical anchor box classes, and a *don't care* class. The network regresses 7 bounding box parameters

for the center, size, and orientation. The Huber loss and $L1$ regularization are used as regression and regularization losses, respectively. We use the original loss coefficients for the total loss.

PointPillars[22]: Similarly, PointPillars regresses the 7 bounding box parameters and utilizes the Huber loss as a regression loss. The orientation is predicted from a set of discrete bins, for which the softmax classification loss is utilized. The focal loss is used as a classification loss for the object classes. All the loss values are combined with respective coefficients for the total loss and we use the default values given in the *mmdetection3d* [8] repository.

CenterPoint[59]: CenterPoint is trained with the focal loss for the heatmap-based classification and the binary cross entropy loss for the IoU-based confidence score. Huber loss is used for the regression of box parameters. We keep the default values given in the *mmdetection3d* [8] repository.

Self-supervised scene flow loss: We utilize the self-supervised loss used in [30] for training the 3D detector backbone and the scene flow head. The loss consists of the nearest neighbor and the cycle consistency losses. The nearest neighbor loss calculates the Euclidean distance of the point $\mathbf{p}'_i$ to its nearest neighbor in frame $t+1$. $\mathbf{p}'_i$ is the point transformed from frame t to $t+1$. For the cycle consistency loss calculation the flow is applied in the forward ($\mathcal{F}_{t\rightarrow t+1}$) and the backward ($\mathcal{F}_{t+1\rightarrow t}$) directions. The distance between the resulting $\mathbf{p}''_i$ point and its anchor $\mathbf{p}_i$ is used for the cycle consistency loss. Both losses are summed up for the total loss and only this loss is used for the training of the backbone and the scene flow head.

4.5 Experiments

We conduct several experiments to show the efficacy of our self-supervised pre-training method on 3D object detection using Point-GNN, PointPillars, CenterPoint, and SSN [63] 3D detectors. First, we compare our self-supervised pre-trained Point-GNN, CenterPoint, and PointPillars with their baselines trained with 100% of the annotated 3D detection data. We show our results on KITTI and nuScenes validation and test sets. Then, we check the performance of our self-supervised detectors and their baselines in the low-data regime by training detectors using only a smaller part of the annotated data in the ablation study. We also report detection accuracy of self-supervised detectors trained with and without alternating training strategy to justify our alternating self-supervised scene flow and supervised 3D detection scheme. Finally, we compare our self-supervised scene flow pre-training method against other self-supervised learning methods.

5 Results & Discussion

In this section, we provide our main results obtained using our scene flow-based self-supervised training with Point-GNN, CenterPoint, and PointPillars 3D detectors on KITTI and nuScenes validation and test sets.

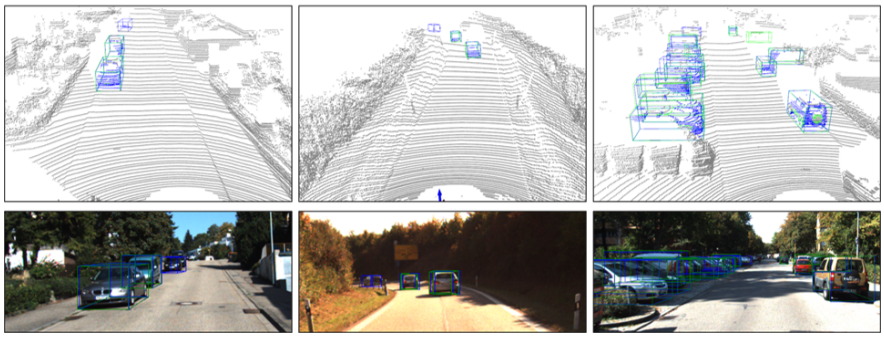

Fig. 3. Qualitative Results: Three different scenes from the KITTI 3D Detection validation set in the columns. Blue and green bounding boxes are for our approach and the baseline Point-GNN [39], respectively. Our method can detect a distant hard object (left-most column) and a moving distant car (middle column) while the baseline misses it. Our method also performs better in a denser environment (right-most column).

Car (IoU=0.7)	3D $AP_{R_{40}}$			BEV $AP_{R_{40}}$		
Method	Easy	Mod	Hard	Easy	Mod	Hard
Point-GNN* [39]	90.44	82.12	77.70	93.03	89.31	86.86
Self-supervised Point-GNN	**91.43**	**82.85**	**80.12**	**93.55**	**89.79**	**87.23**
Improvement	+0.99	+0.73	+2.42	+0.52	+0.48	+0.37
PointPillars	85.41	73.98	67.76	89.93	86.57	85.20
Self-supervised PointPillars	**85.92**	**76.33**	**74.32**	**89.96**	**87.44**	**85.53**
Improvement	+0.51	+2.36	+6.56	+0.03	+0.87	+0.33

Table 1. Self-supervised Point-GNN & PointPillars compared with the baseline on KITTI val. set for car class. (*Reproduced baseline results for $AP_{R_{40}}$.)

5.1 Point-GNN

The self-supervised Point-GNN is pre-trained on the scene flow task with cycle consistency loss using KITTI Tracking dataset without any annotations. Following, it is trained with the annotated KITTI 3D detection dataset using the proposed alternating training scheme. The baseline is trained using the same configuration and hyper-parameters. The only difference is that the baseline network weights are initialized randomly. Table 1 shows the 3D and BEV $AP_{R_{40}}$ scores of the baseline and self-supervised Point-GNN on KITTI validation set, where our method outperforms the baseline in all difficulty levels and especially with a large margin in hard difficulty level (2.5%). Similarly, our self-supervised Point-GNN outperforms its baseline on the KITTI test set on hard difficulty level with a 2% improvement. In supplementary material, we also include the same comparison with $AP_{R_{11}}$ metric using the reported Point-GNN results, where our method outperforms the original Point-GNN. These results show that motion-related point representations help distinguish even difficult objects that reflect only a small number of points.

Car (IoU=0.7)	3D $AP_{R_{40}}$			BEV $AP_{R_{40}}$		
Method	Easy	Mod	Hard	Easy	Mod	Hard
AVOD[20]	76.39	66.47	60.23	89.75	84.95	78.32
F-PointNet[33]	82.19	69.79	60.59	91.17	84.67	74.77
TANet[27]	84.39	75.94	68.82	91.58	86.54	81.19
Associate-3Ddet[11]	85.99	77.40	70.53	91.40	88.09	82.96
UBER-ATG-MMF[24]	88.40	77.43	70.22	93.67	88.21	81.99
CenterNet3D[46]	86.20	77.90	73.03	91.80	88.46	83.62
SECOND[55]	87.44	79.46	73.97	92.01	88.98	83.67
SERCNN[61]	87.74	78.96	74.30	94.11	88.10	83.43
PointPillars	80.51	68.57	61.79	**90.74**	84.98	79.63
Self-supervised PointPillars	**82.54**	**72.99**	**67.54**	88.92	**85.73**	**80.33**
Improvement	+2.03	+4.42	+5.75	-1.82	+0.75	+0.7

Table 2. Self-supervised PointPillars compared with the baseline on KITTI test set for car class using 3D $AP_{R_{40}}$ metric.

The Fig. 3 shows our qualitative results on KITTI 3D Object Detection scenes. The blue bounding boxes and green bounding boxes indicate results of our approach and the baseline, respectively. We show the bird's eye view and front-view lidar visualizations at the top and middle rows. At the bottom, we show the projected 3D bounding boxes on the image plane. Our approach can detect distant objects (left-most column) better as well as distant and moving objects (middle column). In addition, as seen in the right-most column of Fig. 3, our approach can provide better detection results in a denser scene.

5.2 CenterPoint

Our self-supervised CenterPoint also outperforms the CenterPoint baseline on nuScenes validation and test sets as the mAP and NDS results given in Tables 3 and 4, respectively. We obtained the baseline scores with the best-performing *mmdetection3d* CenterPoint checkpoint[8] for both evaluation sets.

5.3 PointPillars

We also report results of our self-supervised pre-training method using PointPillars [22] 3D detector on the nuScenes and KITTI datasets. We pre-train the PointPillars voxel encoder with the self-supervised scene flow task without annotations. Following, the entire PointPillars network is trained on the annotated 3D detection data using our alternating training strategy. We compare our self-supervised PointPillars with its baseline on the KITTI validation and test sets as given in Tables 1 and 2, respectively. Consistent with the previously-introduced results, our method improves the baseline results with a large margin for the 3D detection task. The increment is the most obvious for 3D AP moderate and hard difficulty levels with 2.4% and 6.6% for the validation set and with 4.4% and 5.8% for the test set.

Method	mAP	NDS	Car	Ped	Bus	Barrier	T. C.	Truck	Trailer	Moto.
SECOND[55]	27.12	-	75.53	59.86	29.04	32.21	22.49	21.88	12.96	16.89
PointPillars*[22]	40.02	53.29	80.60	72.40	46.30	52.60	33.60	35.10	26.20	38.40
Self-supervised PointPillars	**42.06**	**55.02**	**81.10**	**74.50**	**49.50**	**54.70**	**34.70**	**38.40**	**29.70**	**38.80**
CenterPoint*[59]	49.13	59.73	83.70	77.40	**61.90**	59.40	**52.90**	50.20	35.00	**44.40**
Self-supervised CenterPoint	**49.94**	**60.06**	**84.10**	**77.90**	61.50	**61.00**	52.50	**51.00**	**35.20**	44.10

Table 3. Self-supervised PointPillars and CenterPoint results on nuScenes validation set.(* mmdetection3d PointPillars checkpoint results, on which we built our work.)

In Table 3, we compare our self-supervised PointPillars with the baseline on nuScenes validation set. The baseline results are obtained from the best checkpoint given in the well-known mmdetection3d repository [8]. Our self-supervised PointPillars outperforms the baseline with a large increment on mAP and NDS metrics (2%) as well as for all class scores. Moreover, we provide results of our self-supervised PointPillars on nuScenes test set in Table 4 comparing to the previously-submitted PointPillars versions from the nuScenes leaderboard.

Method	mAP	NDS	Car	Ped	Bus	Barrier	T. C.	Truck	Trailer	Moto.
PointPillars[22]	30.50	45.30	68.40	59.70	28.20	38.90	30.80	23.00	23.40	27.40
InfoFocus[47]	39.50	39.50	77.90	63.40	**44.80**	47.80	46.50	31.40	37.30	29.00
PointPillars+[44]	40.10	55.00	76.00	64.00	32.10	56.40	45.60	31.00	36.60	34.20
Self-supervised PointPillars	**43.63**	**56.28**	**81.00**	**73.10**	37.10	**58.20**	**47.80**	**36.10**	**41.80**	**35.40**
CenterPoint[44]	49.54	59.64	83.40	76.10	54.20	62.40	62.40	44.40	**48.90**	37.80
Self-supervised CenterPoint	**51.42**	**60.92**	**83.80**	**77.00**	**56.80**	**65.10**	**63.90**	**46.30**	48.50	**41.10**

Table 4. Self-supervised PointPillars and CenterPoint results on nuScenes test set comparing with other PointPillars-based detector and CenterPoint baseline submissions from the nuScenes leaderboard. The CenterPoint baseline is from mmdetection3d.

5.4 Ablation Study

We conduct two types of ablation studies to further justify the effectiveness of our self-supervised pre-training approach: (i) performance after training with limited annotated data and (ii) performance with and without alternating training strategy. Datasets with 3D annotations are mostly limited for the real-world scenarios due to expense and difficulty of requiring expert knowledge for the annotation process. To show our method's enhancement over the baseline using the self-supervised pre-training, we train our self-supervised 3D detectors and baselines with a percentage of the annotated datasets.

Training Data Size	1%			5%			20%		
Car AP (IoU=0.7)	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Point-GNN	63.34	50.92	44.05	81.26	71.27	65.05	88.47	77.20	74.20
SSL Point-GNN	**66.47**	**51.42**	**44.63**	**84.04**	**72.69**	**65.93**	**88.65**	**79.52**	**74.87**
Improvement	+3.13	+0.50	+0.58	+2.78	+1.42	+0.88	+0.18	+2.32	+0.67

Table 5. Self-supervised (SSL) Point-GNN trained with a percentage (1%, 5%, and 20%) of labelled 3D detection data. 3D $AP_{R_{40}}$ results for car class on KITTI val. set.

In Table 5, we show the performance of self-supervised Point-GNN and the baseline trained with 1%, 5%, and 20% of the KITTI train split. Our method consistently outperforms the baseline for all difficulty levels on KITTI validation set. We note that all self-supervised 3D detector ablation results are obtained without alternating training except the alternating training ablation. We conduct the same experiment for PointPillars, CenterPoint, and SSN 3D detectors on nuScenes validation set and report the results in the supplementary material. Similarly, self-supervised 3D detectors outperform their baselines with large margins. Overall, our results suggest that the self-supervised scene flow pre-training can help learn more representative point-wise features in the lack of labelled training data.

In addition, we conduct an ablation study to justify our alternating training strategy. The alternating training enhances the hard difficulty 3D AP with 2.32% increment for Point-GNN. We think that this improvement is due to the repeated motion-awareness of the backbone brought by the first 3D detection fine-tuning. The detailed 3D and BEV AP results for Point-GNN are provided in the supplementary material. Similarly, the alternating training results for PointPillars, CenterPoint, and SSN 3D detectors reported in the supplementary material support our argument.

5.5 Comparison with Other Self-supervised Learning Methods

Our method is the first study that shows the relation between the self-supervised scene flow and 3D detection representations. Our experiments show that the self-supervised scene flow pre-training provides useful point representations for the supervised 3D detection training. In addition, we compare our CenterPoint-based self-supervised scene flow pre-training against other state-of-the-art self-supervised learning methods in Table 6. Our method performs better than other CenterPoint-based self-supervised methods in the low-data regime on the nuScenes validation set.

5.6 Sparse Scene Flow Estimations

Fig. 4 shows visualized sparse scene flow estimations on the sampled KITTI lidar point clouds obtained using Point-GNN backbone. Red points are the sparsely-sampled points at frame t, which are propagated to the frame $t + 1$ using the

Approach	Model	5%		10%	
		mAP	NDS	mAP	NDS
PointContrast[54]	CenterPoint[59]	30.79	41.57	38.25	50.1
GCC3D[23]		32.75	44.2	39.14	50.48
Ours		**36.04**	**48.28**	**41.29**	**51.35**

Table 6. Comparison against other self-supervised learning methods on nuScenes validation set. GCC3D and PointContrast results are taken from [23].

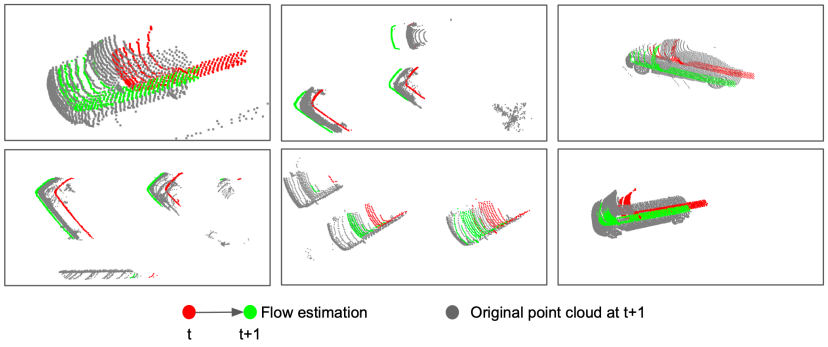

Fig. 4. Sparse scene flow estimation on the sampled KITTI lidar points. Gray points are from the full point clouds at frame $t+1$, red points are sampled points at frame t, and green ones are the propagated points to the frame $t+1$ using scene flow estimation.

estimated flow vectors as shown with the green points. The green points closely match the gray points, which are the original point cloud at frame $t+1$. The network is trained with the cycle consistency loss followed by a 100 epoch fine-tuning on the KITTI Scene Flow Dataset following [30]. This suggests that our scene flow network learns useful point features and therefore the point cloud motion patterns, which improves the 3D object detection accuracy.

6 Conclusion

In this study, we propose a self-supervised backbone training approach for 3D object detection. We utilize large unlabelled datasets for self-supervised training of the 3D detection backbone. The scene flow task is used for the self-supervision using the cycle consistency, which helps the backbone learning the point cloud data structure. We show that our approach can improve the detection results of different 3D detectors comparing to their baselines on KITTI and nuScenes datasets. We also show that self-supervised pre-training is especially helpful with the lack of data. Our approach is flexible and can be combined with different point- and voxel-based 3D detectors.

References

1. Asano, Y.M., Rupprecht, C., Vedaldi, A.: Self-labelling via simultaneous clustering and representation learning. arXiv preprint arXiv:1911.05371 (2019)
2. Basha, T., Moses, Y., Kiryati, N.: Multi-view scene flow estimation: A view centered variational approach. International journal of computer vision **101**(1), 6–21 (2013)
3. Baur, S.A., Emmerichs, D.J., Moosmann, F., Pinggera, P., Ommer, B., Geiger, A.: Slim: Self-supervised lidar scene flow and motion segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13126–13136 (2021)
4. Beker, D., Kato, H., Morariu, M.A., Ando, T., Matsuoka, T., Kehl, W., Gaidon, A.: Monocular differentiable rendering for self-supervised 3d object detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16. pp. 514–529. Springer (2020)
5. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)
6. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. arXiv preprint arXiv:2006.09882 (2020)
7. Chen, Y., Schmid, C., Sminchisescu, C.: Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7063–7072 (2019)
8. Contributors, M.: MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d (2020)
9. Creß, C., Zimmer, W., Strand, L., Lakshminarasimhan, V., Fortkord, M., Dai, S., Knoll, A.: A9-dataset: Multi-sensor infrastructure-based dataset for mobility research. arXiv preprint arXiv:2204.06527 (2022)
10. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1201–1209 (2021)
11. Du, L., Ye, X., Tan, X., Feng, J., Xu, Z., Ding, E., Wen, S.: Associate-3ddet: Perceptual-to-conceptual association for 3d point cloud object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13329–13338 (2020)
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012)
13. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3828–3838 (2019)
14. Gu, X., Wang, Y., Wu, C., Lee, Y.J., Wang, P.: Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3254–3263 (2019)

15. He, C., Zeng, H., Huang, J., Hua, X.S., Zhang, L.: Structure aware single-stage 3d object detection from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11873–11882 (2020)
16. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020)
17. Hou, J., Graham, B., Nießner, M., Xie, S.: Exploring data-efficient 3d scene understanding with contrastive scene contexts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15587–15597 (2021)
18. Huguet, F., Devernay, F.: A variational method for scene flow estimation from stereo sequences. In: 2007 IEEE 11th International Conference on Computer Vision. pp. 1–7. IEEE (2007)
19. Hur, J., Roth, S.: Self-supervised monocular scene flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7396–7405 (2020)
20. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3d proposal generation and object detection from view aggregation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–8. IEEE (2018)
21. Lai, H.Y., Tsai, Y.H., Chiu, W.C.: Bridging stereo matching and optical flow via spatiotemporal correspondence. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1890–1899 (2019)
22. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
23. Liang, H., Jiang, C., Feng, D., Chen, X., Xu, H., Liang, X., Zhang, W., Li, Z., Van Gool, L.: Exploring geometry-aware contrast and clustering harmonization for self-supervised 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3293–3302 (2021)
24. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7345–7353 (2019)
25. Liu, X., Qi, C.R., Guibas, L.J.: Flownet3d: Learning scene flow in 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 529–537 (2019)
26. Liu, X., Yan, M., Bohg, J.: Meteornet: Deep learning on dynamic 3d point cloud sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9246–9255 (2019)
27. Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., Bai, X.: Tanet: Robust 3d object detection from point clouds with triple attention. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11677–11684 (2020)
28. Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., Yuille, A.: Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. IEEE transactions on pattern analysis and machine intelligence **42**(10), 2624–2641 (2019)
29. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4040–4048 (2016)

30. Mittal, H., Okorn, B., Held, D.: Just go with the flow: Self-supervised scene flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11177–11185 (2020)
31. Purushwalkam Shiva Prakash, S., Gupta, A.: Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. Advances in Neural Information Processing Systems **33** (2020)
32. Puy, G., Boulch, A., Marlet, R.: Flot: Scene flow on point clouds guided by optimal transport. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16. pp. 527–544. Springer (2020)
33. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 918–927 (2018)
34. Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., Black, M.J.: Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12240–12249 (2019)
35. Rao, Y., Liu, B., Wei, Y., Lu, J., Hsieh, C.J., Zhou, J.: Randomrooms: Unsupervised pre-training from synthetic shapes and randomized layouts for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3283–3292 (2021)
36. Shao, L., Shah, P., Dwaracherla, V., Bohg, J.: Motion-based object segmentation based on dense rgb-d scene flow. IEEE Robotics and Automation Letters **3**(4), 3797–3804 (2018)
37. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10529–10538 (2020)
38. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 770–779 (2019)
39. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1711–1719 (2020)
40. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)
41. Tschannen, M., Djolonga, J., Ritter, M., Mahendran, A., Houlsby, N., Gelly, S., Lucic, M.: Self-supervised learning of video-induced visual invariances. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13806–13815 (2020)
42. Vedula, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. IEEE transactions on pattern analysis and machine intelligence **27**(3), 475–480 (2005)
43. Vogel, C., Schindler, K., Roth, S.: 3d scene flow estimation with a piecewise rigid scene model. International Journal of Computer Vision **115**(1), 1–28 (2015)
44. Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4604–4612 (2020)
45. Wang, G., Wu, X., Liu, Z., Wang, H.: Hierarchical attention learning of scene flow in 3d point clouds. IEEE Transactions on Image Processing **30**, 5168–5181 (2021)

46. Wang, G., Tian, B., Ai, Y., Xu, T., Chen, L., Cao, D.: Centernet3d: An anchor free object detector for autonomous driving. arXiv preprint arXiv:2007.07214 (2020)
47. Wang, J., Lan, S., Gao, M., Davis, L.S.: Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In: European Conference on Computer Vision. pp. 405–420. Springer (2020)
48. Wang, R., Yang, N., Stückler, J., Cremers, D.: Directshape: Direct photometric alignment of shape priors for visual vehicle pose and shape estimation. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 11067–11073. IEEE (2020)
49. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proceedings of the IEEE international conference on computer vision. pp. 2794–2802 (2015)
50. Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2566–2576 (2019)
51. Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., Chen, M.: Flownet3d++: Geometric losses for deep scene flow estimation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 91–98 (2020)
52. Wei, Y., Wang, Z., Rao, Y., Lu, J., Zhou, J.: Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6954–6963 (2021)
53. Wu, W., Wang, Z., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. arXiv preprint arXiv:1911.12408 (2019)
54. Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L., Litany, O.: Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In: European conference on computer vision. pp. 574–591. Springer (2020)
55. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
56. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3dssd: Point-based 3d single stage object detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11040–11048 (2020)
57. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: Sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1951–1960 (2019)
58. Ye, M., Xu, S., Cao, T.: Hvnet: Hybrid voxel network for lidar based 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1631–1640 (2020)
59. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11784–11793 (2021)
60. Yin, Z., Shi, J.: Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1983–1992 (2018)
61. Zhou, D., Fang, J., Song, X., Liu, L., Yin, J., Dai, Y., Li, H., Yang, R.: Joint 3d instance segmentation and object detection for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1839–1849 (2020)
62. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018)

63. Zhu, X., Ma, Y., Wang, T., Xu, Y., Shi, J., Lin, D.: Ssn: Shape signature networks for multi-class object detection from point clouds. In: European Conference on Computer Vision. pp. 581–597. Springer (2020)
64. Zou, Y., Luo, Z., Huang, J.B.: Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In: Proceedings of the European conference on computer vision (ECCV). pp. 36–53 (2018)