PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences

Jun Wang[1*] Xiaolong Li[2*] Alan Sullivan[3] Lynn Abbott[2] Siheng Chen [4]
[1]University of Maryland [2]Virginia Tech [3]Mitsubishi Electric Research Labs (MERL)
[4]Shanghai Jiao Tong University
junwang@umiacs.umd.edu, {lxiaol9,abbott}@vt.edu,
sullivan@merl.com, sihengc@sjtu.edu.cn

Abstract

We propose a point-based spatiotemporal pyramid architecture, called PointMotionNet, to learn motion information from a sequence of large-scale 3D LiDAR point clouds. A core component of PointMotionNet is a novel technique for point-based spatiotemporal convolution, which finds the point correspondences across time by leveraging a time-invariant spatial neighboring space and extracts spatiotemporal features. To validate PointMotionNet, we consider two motion-related tasks: point-based motion prediction and multisweep semantic segmentation. For each task, we design an end-to-end system where PointMotionNet is the core module that learns motion information. We conduct extensive experiments and show that i) for point-based motion prediction, PointMotionNet achieves less than 0.5m mean squared error on Argoverse dataset, which is a significant improvement over existing methods; and ii) for multisweep semantic segmentation, PointMotionNet with a pretrained segmentation backbone outperforms previous SOTA by over 3.3 % mIoU on SemanticKITTI dataset with 25 classes including 6 moving objects.

1. Introduction

Capturing and interpreting motion information from the environment is critical for autonomous systems that must interact with the three-dimensional (3D) world. Although many researchers have emphasized visual sensing for this purpose (e.g., optical flow for 2D images [12]), the growing popularity of 3D sensing techniques has led to a need for improved motion analysis of 3D point clouds.

It is nontrivial to achieve effective motion learning on a sequence of 3D point clouds. A particular issue is that fine-grained point correspondences are often not available between consecutive sweeps due to spatial and temporal resolution of the sensors. Recent systems, such as FlowNet3D [16], HPLFlowNet [9] and PointPWC-Net [36], have addressed this problem by generalizing optical flow to the 3D domain. However, these techniques rely on learning point-to-point correspondences between consecutive 3D point clouds, and this assumption often does not extend to multiple sweeps. An alternative is MotionNet [35], which handles multiple sweeps by representing 3D point clouds in bird's-eye-view (BEV) maps. However, this approach results in inevitable discretization errors. MeteorNet [17] is a recent method that performs motion learning based on PointNet, which utilizes raw points and does not require voxelization. To handle temporal information, it gathers 3D points from multiple sweeps and encodes the time stamp in an additional feature channel. However, MeteorNet does not design specific operations to explicitly handle temporal information. Occupancy Flow [20] and CaSPR [24] explicitly consider the temporal ordering by leveraging tools like continuous normalizing flows and neural ordinary differential equations. However, these methods work well for 3D point sets that are densely sampled, which is often not the case for sparse, outdoor scenes.

To address these issues, we propose a novel method to achieve motion learning that can handle multiple frames,

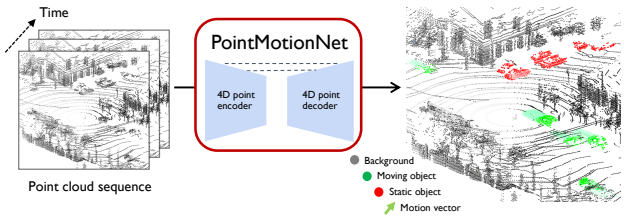

Figure 1. **PointMotionNet is a point-based spatiotemporal pyramid network to achieve motion learning from a sequence of 3D LiDAR point clouds.** It can handle multiple frames, large-scale open scenes, explicitly model the temporal ordering, and does not suffer from discretization error.

*The first two authors contributed equally. Part of this work was done when JW and XL were interns at Mitsubishi Electric Research Labs.

does not suffer from discretization (voxelization) error, explicitly models temporal ordering, and handles large-scale open scenes. Because this method is based on raw points and focuses explicitly on motion learning, we call it *PointMotionNet*; see Figure 1. It has a pyramid structure that enables spatiotemporal feature learning at multiple spatial scales. A core operation in PointMotionNet is a novel point spatiotemporal convolution (*Point-STC*) operation. Instead of looking for point-to-point correspondences, the proposed Point-STC finds the correspondences across time by leveraging the time-invariant neighboring space. It handles 4D points (3D coordinates + time index) in two steps. Step 1 considers spatial information: at each time stamp, the system finds the spatial neighbors for each 4D point and designs trainable multi-kernel functions to aggregate neighboring information to extract spatial features. Step 2 considers temporal information: It concatenates the spatial features sequentially based on their time stamps and then uses a multilayer perceptron to aggregate spatiotemporal features and output final motion features. Overall, PointMotionNet is a point-based architecture that avoids the voxelization issue, explicitly models the temporal ordering, and accommodates multiple sweeps and large-scale scenes.

To validate PointMotionNet, we apply it as a core module to handle two motion-related tasks: point-based motion prediction and multisweep semantic segmentation. For the task of point-based motion prediction, we aim to predict the 3D motion vector for each point in the current frame, which is the displacement from the current 3D position to a future 3D position. We validate the system's performance empirically on the Argoverse [4], and find that PointMotionNet significantly outperforms others. For the task of multisweep semantic segmentation, the system aims to estimate the semantic and motion labels for each point in the current frame, extending semantic segmentation from a single sweep to multiple sweeps. We validate the system's performance on the SemanticKITTI [2], and find that PointMotionNet achieves top performance on the leaderboard.

In summary, the main contributions include:

- we design a novel point-based spatiotemporal convolution operation (Point-STC) that extracts spatiotemporal features from raw points;
- we implement two systems based on PointMotionNet to handle two motion-related tasks, point-based motion prediction and multisweep semantic segmentation; and
- we have conducted extensive experiments on point-based motion prediction and multisweep semantic segmentation. Experimental results show that the proposed approach consistently outperforms previous state-of-the-art methods.

2. Related Work

Deep learning on 3D point clouds. Current approaches can be divided into three categories, based on the input representations: projection-based, voxel-based, and point-based. The projection-based methods, including [14, 19, 25, 30, 33, 34], leverage well-established 2D convolutional networks by projecting 3D point clouds into 2D images. SqueezeSeg [33, 34] projects a 3D LiDAR point cloud onto a spherical surface. RangeNet++ [19] further proposes a nearest-neighbor search for point labels. However, projection-based approaches might suffer from discretization errors that result from the projections. The voxel-based methods, such as [6, 8, 11, 15, 26, 29, 32, 38], focus on mapping points into regular 3D voxels and then applying 2D/3D convolution operations to extract features. PolarNet [38] and Cylinder3D [39] discretize the points based on a polar coordinate system, which makes input points more evenly distributed compared with Cartesian coordinates. However, those methods are characterized by large memory and computational demands.

To directly process raw points, [22, 23] introduce PointNet/PointNet++ for point-cloud classification and segmentation. Many researchers [21, 27, 37] have extended those techniques to 3D object detection. Instead of using the multi-layer perceptron (MLP) from PointNet/PointNet++, KPConv [31] implements an explicit spatial kernel point convolution to directly operate on the input points without any intermediate representation. RandLA-Net [13] demonstrates a new random point sampling method to encode spatial information. In this work, instead of working with a single point cloud, we aim to learn motion information from a temporal sequence of 3D point clouds based on a novel point-based method. Although recent work [1, 7] deal with point cloud sequences, [7] is generic for recognition and segmentation, and [1] aims at panoptic segmentation.

Motion learning on 3D point cloud sequences. Modeling and learning motion information for a temporal sequence of 3D point clouds is crucial to scene understanding for dynamic environments. Recently, FlowNet3D [16], MeteorNet [17] and HPLFlowNet [10], and PointPWC-Net [36] leverage various strategies to generate 3D scene flow between two consecutive frames to represent previous motion displacements. Besides scene flow estimation, there are several studies of motion prediction jointly trained with object detection using deep neural networks [3, 5, 18], which typically require bounding-box annotations. MotionNet [35] predicts future motion behaviors from a sequence of point clouds without bounding box annotations. However, MotionNet works with bird's-eye-view representations, with limited precision by the voxelization. In this work, we aim to capture motion information for each point without any discretization.

3. PointMotionNet

Our goal is to propose a generic neural network module to extract motion information from a temporal sequence of

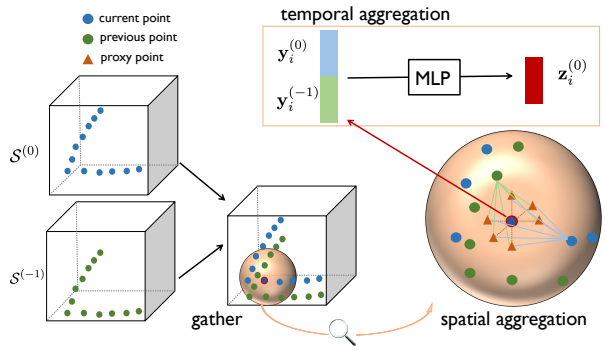

Figure 2. **4D point spatiotemporal convolution operation (Point-STC).** To extract spatiotemporal features, we gather all points across time to the same spatial coordinate system. For each query point (with red circle), we find its neighbors (in the orange ball) in each frame. With the set of proxy points (in red triangle) of the query point, we extract spatial features in each individual frame, which follows from (1a). We next concatenate the spatial features across all time stamps and use a MLP to obtain the final spatiotemporal features, which follows from (1b).

3D point clouds. The motion information could include the motion state (static/moving), the motion displacement and many others. Let $\mathcal{P} = \{\mathcal{S}^{(t)}\}_{t=0}^{-T}$ be a temporal sequence of 3D point clouds, where the set $\mathcal{S}^{(t)} = \{p_i^{(t)}\}_{i=1}^{N_t}$ is the point cloud collected at the tth frame with $p_i^{(t)}$ the ith point in the tth frame. Here $t = 0$ denotes the current frame and $t < 0$ denotes previous frames. Let $\mathbf{p}_i^{(t)} \in \mathbb{R}^3$ be the 3D coordinate of the point $p_i^{(t)}$. Since $\mathbf{p}_i^{(t)}$ is also indexed by time, together with 3D coordinates, we consider it a *4D point* and $\mathcal{P}$ a *4D point cloud*. We aim to design a neural network $f(\cdot)$ to produce motion information for each point in the current frame; that is, $\mathbf{m}_i^{(0)} = f(p_i^{(0)}, \mathcal{P})$, where $\mathbf{m}_i^{(0)}$ could be a scalar to indicate the moving probability, or a 3D vector to indicate the motion displacement.

3.1. 4D Point Spatiotemporal Convolution

To extract motion features for 4D points, we consider a spatiotemporal convolution operation. Let $\mathbf{x}_i^{(t)}$ be the input features of the point $p_i^{(t)}$. For $p_i^{(t)}$, the 4D point spatiotemporal convolution (Point-STC) operates as,

$$\mathbf{y}_i^{(t+\tau)} = \sum_{\mathbf{p}_j^{(t+\tau)} \in \mathcal{S}^{(t+\tau)} \bigcup \mathcal{N}(\mathbf{p}_i^{(t)})} \Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)}\right), \tag{1a}$$

$$\mathbf{z}_i^{(t)} = \text{MLP}\left(\text{concat}\left(\{\mathbf{y}_i^{(t+\tau)}\}\right)_{\tau=0}^{-T}\right), \tag{1b}$$

where $\text{MLP}(\cdot)$ denotes the multilayer perceptron network shared by all the points and $\text{concat}(\cdot)$ denotes the concatenation operation that combines several short vectors to output a long vector, a kernel function $\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)}\right)$ that evaluates the effect from point $p_j^{(t+\tau)}$ to point $p_i^{(t)}$ with τ being time difference and $\mathcal{N}(\mathbf{p}) = \{\mathbf{q} \in \mathbb{R}^3 | \|\mathbf{q} - \mathbf{p}\|_2 \leq r\}$ defines a neighboring set of points for point $\mathbf{p}$ with $r \in \mathbb{R}$ a pre-defined radius, reflecting a fixed 3D-ball space. Step (1a) is a spatial convolution that aggregates the neighboring information at each single frame and $\mathbf{y}_i^{(t+\tau)}$ denotes the intermediate feature extracted at frame $t+\tau$; and step (1b) aggregates temporal features sequentially based on time stamps and $\mathbf{z}_i^{(t)}$ denotes the output feature for $p_i^{(t)}$. Since the sequence length T is usually small in practice, MLP works effectively. Combining (1a) and (1b), we achieve information aggregation over both the spatial and temporal dimensions; see in Figure 2. Note that instead of looking for point-to-point correspondences, Point-STC finds correspondences across time by leveraging the time-invariant neighboring space. Because for each point, the neighboring space is fixed given a radius, we can evaluate the spatial features within this space across time to extract motion features.

To define the kernel function $\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)}\right)$, instead of directly considering the pairwise effect from $p_j^{(t+\tau)}$ to $p_i^{(t)}$, inspired by [31], we first introduce a few proxy points to represent each 4D point and then consider the effect from $p_j^{(t+\tau)}$ to each of the proxy points. The benefit is to leverage multiple kernels to make the training more stable and effective. The set of proxy points for a 4D point $\mathbf{p}_i^{(t)}$ is $\mathcal{K}(\mathbf{p}_i^{(t)}) = \{\mathbf{p}_i^{(t)} - \mathbf{b}_k \in \mathbb{R}^3\}_{k=1}^K$, where the offsets $\mathbf{b}_k \in \mathbb{R}^3$ are pre-defined so that each proxy point belongs to the vertices or center of a regular polyhedron in 3D space; see the definition in supplementary. With this, we consider a multi-kernel function as

$$\Psi_{\mathbf{p}_i^{(t)}, \mathbf{p}_j^{(t+\tau)}} \left(\mathbf{x}_j^{(t+\tau)}\right) = \sum_{\delta_k \in \mathcal{K}(\mathbf{p}_i^{(t)})} w_k h(\delta_k, \mathbf{p}_j^{(t+\tau)}) \mathbf{x}_j^{(t+\tau)}, \tag{2}$$

where w_k are trainable parameters shared by all the points and the kernel associated with each proxy point is

$$h(\delta_k, \mathbf{p}_j^{(t+\tau)}) = \max\left(0, 1 - \frac{\left\|\delta_k - \mathbf{p}_j^{(t+\tau)}\right\|_2}{\sigma}\right) \in \mathbb{R}, \tag{3}$$

where $\sigma = r/2$ is a hyperparameter related to the neighboring radius. In (2), $h(\delta_k, \mathbf{p}_j^{(t+\tau)})$ reflects the spatial proximity prior between $\mathbf{p}_j^{(t+\tau)}$ and the kth proxy point of $\mathbf{p}_i^{(t)}$ and w_k can further adaptively change the kernel weight. Note that the set of proxy points is similar to rigid kernel points in [31]. The difference is that KP-Conv considers a pair of points at the same time stamp, while (2) evaluates a pair of points across time. This is also related to grid points in [28], where they use a fixed global grid; while we recenter the proxy points to each query point, where the

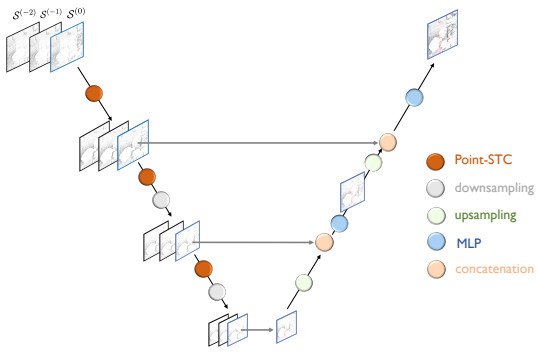

Figure 3. **PointMotionNet is a point-based spatiotemporal pyramid network.** Point-STC layers enable spatiotemporal feature learning. Downsampling and upsampling layers enable multiscale feature learning. With the skip connections, the aggregation layers fuse features from both encoder and decoder.

relative displacements are shift-invariant.

The proposed Point-STC (1) directly handles raw 4D points without any discretization. Since the operation only takes the neighboring points, we do not need to handle the entire point cloud at the same time; instead, we can split a large-scale point cloud into multiple patches, which are small-scale point clouds, and handle each patch locally. This can significantly ease the computation and make it easier to handle large-scale 3D point clouds.

Compared to recent point-based methods, the proposed Point-STC can effectively operate on a sequence of large-scale 3D point clouds. Specifically, FlowNet3D [16] and HPLFlowNet [10] generate scene flow by learning the point-to-point correspondence between two frames. However, they could only operate on two frames; while the proposed Point-STC can operate on multiple frames by concatenating spatial features from all frames. Although MeteorNet [17] can take multiple frames, it gathers the 3D points from all frames together and encodes the time stamp as a separate feature channel, which is not dedicated to extracting the temporal information; while the proposed Point-STC preserves the temporal ordering by sequentially concatenating spatial features from each frame and uses MLP to fully extract temporal information.

3.2. 4D Point Pyramid Network Architecture

Here we propose a network architecture to systematically handle a sequence of 3D point clouds. This architecture has a pyramid structure; see Figure 3. The left part acts like an encoder that extracts spatiotemporal features at multiple scales; and the right part acts like a decoder that only focuses on aggregating features at the current frame from both the previous layer and the encoder.

Point-STC layer. As the main compontent to extract spatiotemporal features from points, a Point-STC layer implements the 4D point spatiotemporal convolution (1). The first layer takes raw points from multiple frames as the input, where the point features are their 3D coordinates. For the deeper layers, the point features are hidden features from the previous layer. For each point, the spatiotemporal neighborhood is implemented by collecting certain number of the 3D points within a pre-defined spatial distance to this query point at each time stamp. The neighborhood radius is defined as $r_{\ell+1} = 2r_\ell$, where ℓ is the network layer index. In a deeper layer, we need a larger reception field for a Point-STC to extract more global features.

Downsampling layer. Since we increase the reception field in a deep layer, the corresponding Point-STC needs to handle many more points. In a large-scale scene, a point cloud is usually irregularly sampled, where nearby points are relatively denser and far-away points are relatively sparser. Uniformly downsampling would make far-away points too sparse. To resolve this issue, we adopt downsampling based on grids following [31]. We first partition a 3D scene into a series of nonoverlapping 3D grids. For a non-empty grid cell with multiple 3D points, we only take their barycenter as their proxy. At each time stamp, we collect the sampled points from all the grids to be a *downsampled 3D point cloud*. In this manner, we would downsample more points nearby and less points far-away, balancing the spatial distribution of point density. The grid size is defined as $d_{\ell+1} = 2d_\ell$, where ℓ is the network layer index. Note that the downsampled 3D points are still irregularly scattered in the 3D space, which is clearly different from the regular voxelization.

Upsampling layer. Each upsampling layer aims to populate point features from a downsampled point cloud to an original point cloud (previous layer before the downsampling layer). For each 3D point in the original point cloud, we find its closest point in the downsampled point cloud and take its associated point features.

Aggregation layer. Each aggregation layer aims to aggregate point features from both the encoder and the decoder at the same scale. In our pyramid architecture, we use a skip connection to introduce the point features at the current time stamp from the encoder side and concatenate them with the corresponding point features from the upsampled point cloud. We then use a multilayer perceptron to aggregate the concatenated features.

Compared with previous works, the proposed pyramid architecture explicitly preserves the temporal ordering. MeteorNet [17] merges the entire input sequence of 3D point clouds directly to a denser 3D point cloud without ordering them among the temporal axis; while the proposed architecture preserves the complete temporal information and our design decouples the spatio-temporal convolution along spatial and temporal dimensions. In addition, MeteorNet [17] uses FPS sampling as in PointNet++ [23], while we adopt grid-based sampling, which balances the spatial

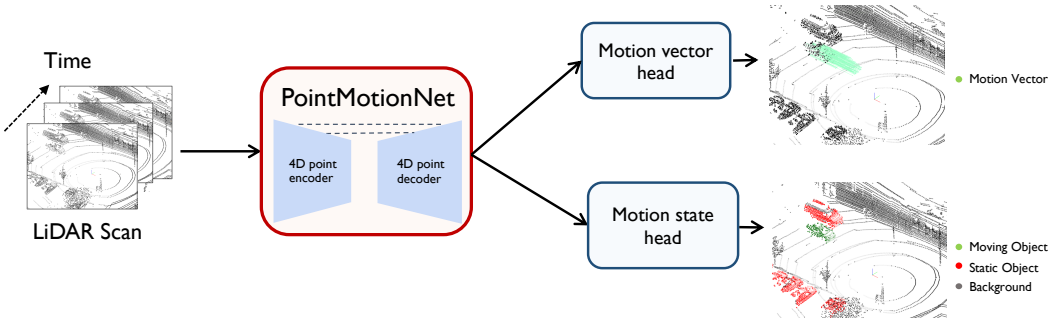

Figure 4. **Point-based motion prediction system.** With PointMotionNet as a backbone, the system takes a sequence of 3D point clouds as the input and generates the predicted motion vector and the estimated motion state for each single 3D point in the current frame through the motion vector head and the motion state head. For visualization, we sample the moving points for the motion vector output.

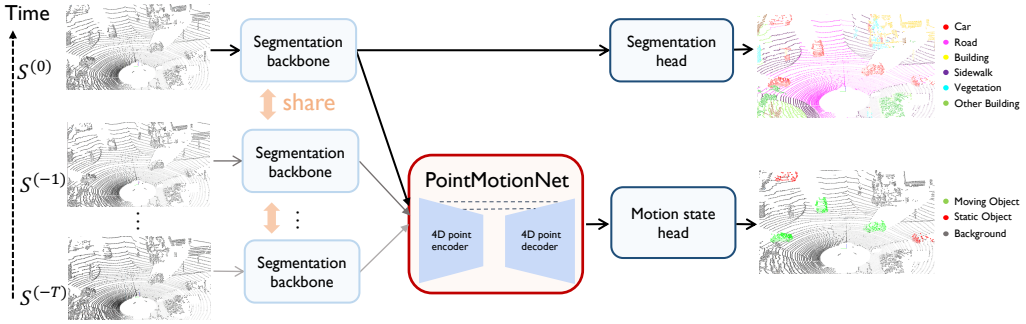

Figure 5. **Multisweep semantic segmentation system.** It consists of four parts: the segmentation backbone, extracting semantic features from each individual 3D point cloud, the segmentation head, producing the estimated semantic labels, PointMotionNet, extracting motion features based on semantic features across time stamps, and the motion state head, producing the estimated motion states.

distribution of point density. [6] relies on high-dimensional sparse convolutions to extract spatiotemporal information from discretized 4D data; while PointMotionNet directly operates on raw points instead of voxels. The proposed Point-STC leverages a fixed set of proxy points, and it covers a general polyhedron compared to regular voxels.

4. Motion-Related Systems

In this section, we apply PointMotionNet as a core module to two motion-related systems.

4.1. Point-based Motion Prediction

Our aim is to predict a motion vector for each point in the current frame given a sequence of 3D point clouds. Each motion vector of a point is the displacement from the 3D coordinate at the current time stamp to the 3D coordinate at some future time stamp. Here we consider point-level motion, instead of an instance level or a cell level.

Architecture. Figure 4 illustrates a point-based motion prediction system. The proposed PointMotionNet takes a sequence of 3D point clouds as the input and produces the intermediate motion features to two output heads: the motion vector head, which predicts the motion vector for each 3D point in the current frame, and the motion state head, which estimates the binary motion state for each 3D point in the current frame. As the backbone network, the PointMotionNet has 4 downsampling layers in total. We also set the neighborhood radius in the first Point-STC layer be $r_\ell = 0.15m$, the grid size in the first downsampling layer be $d_\ell = 0.06m$, the number of the proxy points be 15. The motion vector head is implemented by three fully-connected layers and the motion state head is implemented by one fully-connected layer followed by a softmax layer.

Loss function. We apply the smooth L1 loss L_{vector} to supervise motion vector prediction and use the cross-entropy loss L_{state} for motion state. We jointly train motion state estimation and motion vector prediction with the overall loss $L = L_{\text{vector}} + 0.1L_{\text{state}}$. In the testing phase, the estimated binary motion states are used to refine the predicted motion vectors: when the estimated motion state is static, the corresponding motion vector is set to zero.

4.2. Multisweep Semantic Segmentation

The task of multisweep semantic segmentation aims to estimate the semantic label and motion state for each 3D point in the current sweep given a sequence of point clouds.

Architecture. A straightforward design for multisweep semantic segmentation is to take multiple sweeps as the

Method	View	$\text{mIoU}_{\text{total}} \uparrow (\%)$	$\text{mIoU}_{\text{static}} \uparrow (\%)$	$\text{mIoU}_{\text{moving}} \uparrow (\%)$	Static $\downarrow$	Speed $\leq$ 5m/s $\downarrow$	Speed $>$ 5m/s $\downarrow$
MotionNet [35]	BEV	94.8	97.8	91.7	0.007	0.755	0.503
FlowNet3D [16]	Point	75.2	90.5	59.9	0.003	1.535	1.834
MeteorNet [17]	Point	93.8	97.4	90.2	0.005	1.014	0.749
PointMotionNet	Point	**98.5**	**99.4**	**97.6**	**0.001**	**0.416**	**0.445**

Table 1. **PointMotionNet outperforms in the motion prediction task on Argoverse Validation Set**.

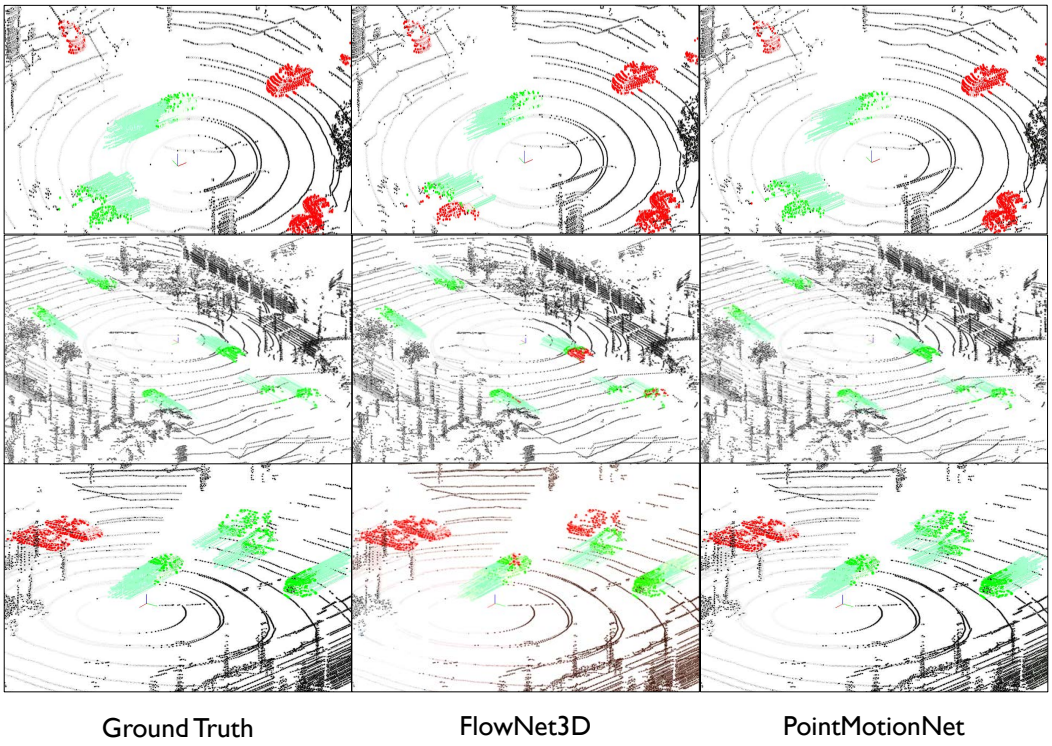

Figure 6. **PointMotionNet (right) qualitatively outperforms FlowNet3D (middle) in point-based motion prediction on Argoverse**. Moving points are associated with green motion vectors and the static points are in red; the remaining points are background, marked in grey.

Models	$\text{mIoU}_{\text{total}} \uparrow (\%)$	$\text{mIoU}_{\text{static}} \uparrow (\%)$	$\text{mIoU}_{\text{moving}} \uparrow (\%)$
RangeNet++ [19]	72.0	97.6	46.5
MotionNet [35]	81.4	98.6	64.2
FlowNet3D [16]	59.0	97.1	20.8
MeteorNet [17]	72.0	97.9	46.1
4D-PointNet2 [23]	68.3	97.1	39.5
PointMotionNet	**81.8**	**98.7**	**64.9**

Table 2. **PointMotionNet outperforms in motion state estimation on SemanticKITTI**. RangeNet++ and MotionNet are in the range view and BEV, respectively, while the rest are point-based.

input and directly estimate all the semantic and motion labels. However, we observe that for foreground categories, such as car, person, and bicyclist, semantic information and motion information are mostly independent. Jointly training the semantic and motion information would potentially harm the performances on both tasks. Moreover, previous sweeps might not significantly benefit the point-wise semantic labeling of the current sweep because the points across sweeps do not have a clear point-to-point correspondence. Therefore, our design rationale is to split multisweep semantic segmentation into two separated tasks: semantic segmentation based on a single sweep and motion state estimation based on multiple sweeps.

Figure 5 illustrates a multisweep semantic segmentation system. The overall architecture consists of four parts: the segmentation backbone, the segmentation head, PointMotionNet and the motion state head. The segmentation backbone consumes each individual sweep and learns per-point semantic features. All sweeps share the same backbone network. The segmentation head takes semantic features of all the points in the current sweep to generate the per-point semantic labels. It is implemented by a

Approach	25-class mIoU	12-class mIoU	car (S)	car(M)	bicyc. (S)	bicyc.(M)	person (S)	person(M)	motor. (S)	motor.(M)	other-veh. (S)	other-veh.(M)	truck (S)	truck(M)
TangentConv [30]	34.1	20.7	84.9	40.3	0.0	30.1	1.6	1.9	0.0	42.2	18.5	6.4	21.1	1.1
DarkNet53 [2]	41.6	24.2	84.1	61.5	0.0	28.9	7.5	0.2	0.0	37.8	20.7	15.2	20.0	14.1
SpSeqnet [26]	43.1	25.5	88.5	53.2	0.0	2.3	6.3	36.2	0.0	0.1	22.7	26.2	29.2	**41.2**
LatticeNet [25]	45.2	33.7	91.1	54.8	0.0	44.6	6.8	49.9	0.0	**64.3**	23.1	0.6	**65.4**	3.5
KP-Conv [31]	51.2	**40.5**	93.7	69.4	0.0	**67.4**	**21.6**	**67.5**	0.0	47.2	**38.6**	0.5	42.5	0.5
Cylider3D [39]	51.5	31.4	**93.8**	68.1	0.0	60.0	12.9	63.1	0.1	0.4	37.6	0.1	41.2	0.0
PointMotionNet	**54.8**	39.2	**93.8**	**71.4**	**6.2**	62.2	13.7	61.8	**23.4**	15.1	38.1	**29.9**	44.2	10.5

Table 3. PointMotionNet outperforms in multisweep semantic segmentation on SemanticKITTI test set. Note that S represents the static foreground category, and M denotes the moving foreground category. Please refer to the supplementary for the detailed 25 classes results.

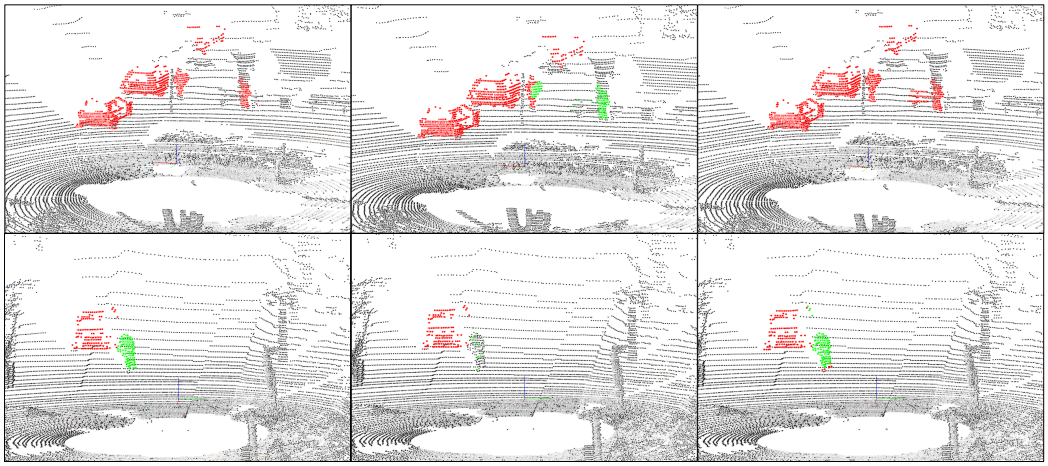

Figure 7. PointMotionNet (right) visually outperforms MeteorNet (middle) in motion state estimation on SemanticKITTI. For six foreground classes, moving points are in green and static ones are in red; for the remaining background classes, points are in grey.

fully-connected layer followed by a softmax layer. The third component, PointMotionNet, learns motion features based on semantic features from the segmentation backbones across all sweeps. The motion state head takes the motion features to estimate the motion state for each point in the current sweep. It is implemented by a fully-connected layer followed by a softmax layer.

Loss function. We adopt the cross-entropy losses for both multi-category semantic segmentation and the binary motion state estimation. Based on the groundtruth of the semantic category, we apply a foreground mask to the motion state estimation, so that only the losses of foreground points are counted. The overall loss function combines the semantic segmentation loss and the motion state loss. In the training phase, we could also use a pre-trained segmentation backbone to focus on motion training and accelerate the training time. In the inference phase, for estimated background categories, the system would set the corresponding estimated motion states to be static.

5. Experimental Results

In this section, we validate PointMotionNet both quantitatively and qualitatively on Argoverse and SemanticKITTI.

5.1. Point-based Motion Prediction

Dataset. Argoverse [4] is an autonomous driving dataset for 3D tracking and forecasting. We follow the official split to generate 65 logs for training and 24 logs for validation. Since there is no direct labeling for motion information, we generate the groundtruth based on 3D bounding boxes; see more ablation studies in supplementary.

Baselines. We compare PointMotionNet with several state-of-the-art methods, including MotionNet [35], which

proxy points(k)	d_1	temporal op.	# frames (n)	$\text{mIoU}_{\text{total}}$	$\text{mIoU}_{\text{static}}$	$\text{mIoU}_{\text{moving}}$	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s
15	0.06	MLP	3	**0.985**	0.994	**0.976**	**0.001**	0.416	0.445
6	0.06	MLP	3	0.981	0.992	0.970	0.002	0.461	0.543
15	*0.24*	MLP	3	0.983	0.993	0.974	0.002	0.398	0.461
15	0.06	*Conv.*	3	0.976	0.990	0.962	0.002	0.558	0.782
15	0.06	MLP	*4*	0.982	**0.996**	0.969	**0.001**	**0.341**	**0.422**

Table 4. Ablation Study of PointMotionNet on Argoverse Validation Set. We vary several hyper-parameters: number of proxy points k, initial sampling resolution d_1, temporal operation, and input frames number n.

is a BEV-based method, and MeteorNet [17], which is a point-based method. We adjust the output head of MeteorNet for future motion vector prediction and motion state estimation. We also consider a 3D scene flow method, FlowNet3D [16], which only takes 2 frames. We train it to estimate the 3D scene flow between one past frame and current frame, and then lift the predictions to motion displacements in future frames using a linear motion model.

Experimental setup. The inputs are 3 sequential frames of point clouds $\mathcal{S}^{(t-0.2s)}, \mathcal{S}^{(t-0.1s)}, \mathcal{S}^{(t)}$, which are synchronized to the same coordinate system. The final motion prediction contains a binary motion state, i.e. static or moving, together with the displacements from current time stamp to the next 5 future time stamps $\{t+\tau\}_{\tau=0.1s}^{0.5s}$.

Results. We adopt the mean intersection-of-union (mIoU) [2] to evaluate motion state estimation for static and moving points. For the motion vector prediction, we evaluate on two speed intervals, which shows performances on both slow and fast motions. Since MotionNet will only predict 2D motion vector at each cell, we reproject its predictions back to each point, and quantitatively evaluate all methods on xy plane only. Table 1 shows that as a point-based method, PointMotionNet outperforms its competitors. Figure 6 shows that PointMotionNet visually outperforms FlowNet3D. Due to the limitation of fusing only two consecutive frames, FlowNet3D tends to easily fail to predict the motion state of objects either too closer or the too far away from the LiDAR sensor.

Ablation Studies. Table 4 considers several variants of PointMotionNet. We see that i) Less proxy points results in slightly worse results; ii) increasing the grid resolution up to 0.24m in downsampling makes minor effects; iii) MLP works significantly better than convolution in the temporal aggregation (1b); and iv) increasing the number of sweeps improves the performance; see more in supplementary.

5.2. Multisweep 3D Semantic Segmentation

Dataset. SemanticKITTI [2] is another one of the largest datasets for LiDAR-based semantic segmentation. There are 22 labeled sequences consisting of over $43,000$ input laser scans. The experiments follow the standard protocol on the training, validation and test sequences. Note that we manually remove the first 130 corrupted validation scans with inconsistent annotation when we evaluate PointMotionNet and previous architectures. The multiple-scan benchmark evaluates mIoUs on the 25 classes, including six moving and non-moving foreground classes, car, bicyclist, person, motorcyclist, other-vehicle and truck.

Baselines. We compare PointMotionNet with several state-of-the-art methods, including KP-Conv [31], and recent cylinder coordinates-based Cylider3D [39].

Experimental setup. We use the SGD optimizer with momentum 0.98 and weight decay of 10^{-3} and an initial learning rate of 0.01. We train the PointMotionNet and the heads for 500 epochs. Following MotionNet [35], we use a consistency loss to enforce that the points belonging to the same object have a same motion state.

Results. First, we validate PointMotionNet in the point motion state estimation task on SemanticKITTI validation set. We compute the overall $\text{mIoU}_{\text{total}}$, $\text{mIoU}_{\text{static}}$ for static points and $\text{mIoU}_{\text{moving}}$ for moving points. Table. 2 shows that PointMotionNet outperforms its competitors. Second, we visualize the motion state estimation results for the foreground classes on SemanticKITTI validation set. Figure 7 shows that with the same semantic segmentation backbone, PointMotionNet outperforms MeteorNet. Third, we evaluate the overall multisweep semantic segmentation performance on the SemanticKITTI test server. Table. 3 shows that the proposed PointMotionNet could improve the overall performance on 25 classes semantic segmentation using a simple single-frame semantic segmentation backbone.

6. Conclusion

The proposed PointMotionNet is a novel point-based method to achieve motion learning. It handles multiple frames and large-scale scenes, avoid discretization and explicitly learn from the temporal ordering. We further integrate PointMotionNet into two systems to handle point-based motion prediction and multisweep semantic segmentation. Experimental results show that PointMotionNet outperforms baselines on Argoverse and SemanticKITTI.

Acknowledgement: We thank the computing resources provided by ARC super-computing center at Virginia Tech. We are grateful for the hardware support from MERL. We also appreciate the helpful discussions from Pengxiang Wu.

References

[1] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic lidar segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5527–5537, 2021. 2

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 2, 7, 8

[3] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 2

[4] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 7

[5] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving. *arXiv preprint arXiv:2003.00601*, 2020. 2

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2, 5

[7] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. Pstnet: Point spatio-temporal convolution on point cloud sequences. In *International conference on learning representations*, 2020. 2

[8] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 2

[9] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3254–3263. Computer Vision Foundation / IEEE, 2019. 1

[10] 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*, pages 3254–3263, 2019. 2, 4

[11] Tianrui Guan, Jun Wang, Shiyi Lan, Rohan Chandra, Zuxuan Wu, Larry Davis, and Dinesh Manocha. M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 772–782, 2022. 2

[12] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981. 1

[13] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019. 2

[14] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 2

[15] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2

[16] 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*, pages 529–537, 2019. 1, 2, 4, 6, 8

[17] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9246–9255, 2019. 1, 2, 4, 6, 8

[18] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 2

[19] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. 2, 6

[20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019. 1

[21] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[22] 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 (CVPR)*, July 2017. 2

[23] 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. 2, 4, 6

[24] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J Guibas. Caspr:

Learning canonical spatiotemporal point cloud representations. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[25] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: fast spatio-temporal point cloud segmentation using permutohedral lattices. *Autonomous Robots*, pages 1–16, 2021. 2, 7

[26] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020. 2, 7

[27] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[28] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 3

[29] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, 2020. 2

[30] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2, 7

[31] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Franccois Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 2, 3, 4, 7, 8

[32] Jun Wang, Shiyi Lan, Mingfei Gao, and Larry S Davis. Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In *European Conference on Computer Vision*, pages 405–420. Springer, 2020. 2

[33] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 2

[34] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019. 2

[35] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020. 1, 2, 6, 7, 8

[36] 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. *arXiv preprint arXiv:1911.12408*, 2019. 1, 2

[37] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[38] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020. 2

[39] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar-based perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 7, 8