


Sequential Multi-View Fusion Network for Fast LiDAR Point Motion Estimation

Gang Zhang², Xiaoyan Li¹, and Zhenhua Wang³

¹ Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology,
Beijing Artificial Intelligence Institute, Faculty of Information Technology, Beijing
University of Technology, Beijing 100124, China

² Damo Academy, Alibaba Group

³ Cenozoic Robot

{zhanggang11021136,hblix2,zhwang.me}@gmail.com

Abstract. The LiDAR point motion estimation, including motion state prediction and velocity estimation, is crucial for understanding a dynamic scene in autonomous driving. Recent 2D projection-based methods run in real-time by applying the well-optimized 2D convolution networks on either the bird’s-eye view (BEV) or the range view (RV) but suffer from lower accuracy due to information loss during the 2D projection. Thus, we propose a novel sequential multi-view fusion network (SMVF), composed of a BEV branch and an RV branch, in charge of encoding the motion information and spatial information, respectively. By looking from distinct views and integrating with the original LiDAR point features, the SMVF produces a comprehensive motion prediction, while keeping its efficiency. Moreover, to generalize the motion estimation well to the objects with fewer training samples, we propose a sequential instance copy-paste (SICP) for generating realistic LiDAR sequences for these objects. The experiments on the SemanticKITTI moving object segmentation (MOS) and Waymo scene flow benchmarks demonstrate that our SMVF outperforms all existing methods by a large margin.

Keywords: Motion State Prediction, Velocity Estimation, Multi-View Fusion, Generalization of Motion Estimation

1 Introduction

Based on the 3D point clouds captured at consecutive time steps, the motion estimation aims at describing which parts of the scene are moving and where they are moving to. It is a combination of moving object segmentation and scene flow estimation, which are highly related and can be given by a single inference. The motion estimation provides a low-level understanding of a dynamic scene that not only benefits downstream tasks, such as object trajectory prediction, object tracking, simultaneous localization and mapping (SLAM), *etc.*, but also remedies for undetected objects in the perception system. Therefore, an accurate, fast, and well-generalized motion estimation algorithm is crucial for real-world

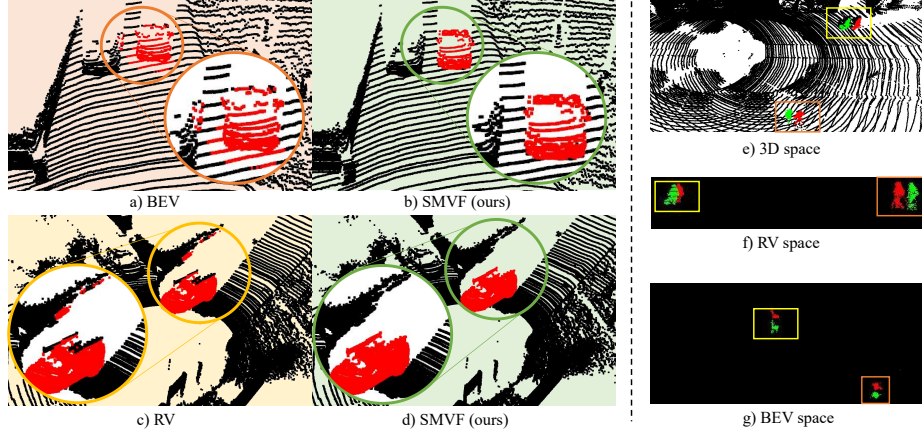


Fig. 1. The left part shows that the previous single-view methods frequently confuse points projected to the same or nearby 2D grids (a,c), while our SMVF solves the problem (b,d). The predicted moving points are shown in red, while the stationary ones are shown in black. The right part shows two moving pedestrians with almost the same velocity in the 3D space (e), while their trajectories are distorted in the RV space (f) but consistent in the BEV space (g). The red pedestrians are in the current scan and the green ones are in the history scan.

applications, *e.g.* autonomous driving. Previous methods are grouped as point-based methods, sparse voxel-based methods, and 2D projection-based methods.

The point-based methods, including FlowNet3D [14], FlowNet3D++ [24], MeteorNet [15], and HPLFlowNet [9], directly process the raw unordered 3D points. These methods extract motion information by searching the spatiotemporal neighborhoods, which is both memory and computation inefficient. Thus, these methods are rarely used in real-world autonomous driving systems. To reduce the memory and computation costs, the sparse voxel-based methods [4], [18] quantize a sequence of the LiDAR points into sparse voxels, and apply convolution operations on these non-empty voxels along spatial and temporal dimensions. However, they are still too computationally expensive to run in real-time.

The 2D projection-based methods first project the 3D points onto a 2D plane, such as bird’s-eye view (BEV) [10], [25] or range view (RV) [3], [6], to generate a sequence of 2D pseudo images, where existing optical flow methods [5], [19], [17] of the 2D image domain can be applied. Generally, these methods achieve real-time inference speed but have lower accuracy due to the 2D projection information loss as shown in Fig. 1(a,c).

For accurate and fast LiDAR point motion estimation, it is argued that the deficiencies of the 2D projection-based paradigms can be remedied by division and cooperation between different views. As shown in Fig. 1(b,d), our SMVF solves this problem by using the proposed multi-view features since the distinct 3D points that are confused in the BEV space are separated in the RV space. Moreover, in this work, the properties of different views are explored. Specifically,

the BEV branch consumes multiple LiDAR scans for motion features extraction, considering that objects primarily move on the BEV plane. The RV branch only uses the current scan to extract the spatial features to distinguish the 3D points that fall in the same BEV grid, since the object trajectories are distorted in the RV space as shown in Fig. 1(e,f,g). Moreover, the original point features are also fused to alleviate other information loss during the 2D projection.

In this work, we present a novel sequential multi-view fusion network (SMVF), which consists of two complementary branches, namely a BEV branch and an RV branch. The BEV branch is responsible for motion features extraction according to a sequence of consecutive LiDAR scans, while the RV branch extracts complementary spatial features only from the current scan. Finally, the motion and spatial features are fused with the 3D point features to acquire per-point motion estimation. Besides, the sequential instance copy-paste (SICP) is designed to generate and insert object trajectories into a sequence of LiDAR scans to mimic the real data and further improve the generalization ability of a motion estimator. Our contributions include:

- The SMVF is designed for accurate and fast LiDAR point motion estimation by fusing motion features from the BEV and spatial features from the RV.
- The SICP is proposed to generalize the motion estimation well to the objects with fewer training samples.
- Our SMVF currently ranks **1st** on the SemanticKITTI MOS leaderboard and Waymo scene flow benchmark, running 13 ms on NVIDIA RTX 2080Ti GPU with TensorRT [22] FP16 inference mode.

2 Related Work

Point motion estimation, including moving object segmentation (MOS) and scene flow estimation, can not only benefit the downstream tasks, such as tracking and SLAM, but also serve as a powerful basis for planning when the object detection and tracking systems fail in the presence of undetected objects.

Moving Object Segmentation (MOS). In this task, each 3D point is classified as moving or stationary. SpSequenceNet [18] divides the 3D space uniformly into structured voxels, and then a cross-frame global attention module and a cross-frame local interpolation module operating on these sparse voxels are proposed to capture spatial and temporal information. Recently, Chen *et al.* [3] project the 3D points to the RV, and generate the 2D residual distance images between the current frame and history frames to provide motion information. Different from the previous voxel-based methods, the 2D projection-based methods support real-time perception and are easily deployed on an autonomous vehicle.

Scene Flow Estimation. In this work, the scene flow estimation refers to predicting the velocity for each 3D point instead of each BEV grid. To achieve this

goal, a series of point-based methods are proposed. FlowNet3D [14] adopts the time-consuming farthest point sampling (FPS) for downsampling points, and ball query for spatiotemporal neighbor searching. MeteorNet [15] proposes a direct grouping and a chained-flow grouping to determine the spatiotemporal neighborhoods. Inspired by Bilateral Convolution Layers (BCL), HPLFlowNet [9] proposes DownBCL, UpBCL, and CorrBCL that restore the spatial relationship of points to accelerate the FlowNet3D. These point-based methods can preserve and exploit the information from raw point clouds, but they are computation and memory inefficient. Recently, the FastFlow3D [10] fuses the BEV motion features and the point features for real-time scene flow estimation. However, the point features cannot be fully digested by a shared MLP and fail to compensate for the aforementioned projection information loss. Therefore, the FastFlow3D may classify the stationary points as moving.

Other Spatiotemporal Encoding Methods Other methods that also exploit the spatiotemporal information are introduced for inspiration. For multi-scan LiDAR semantic segmentation, Minkowski CNN [4] projects 3D points into voxels and applies a novel 4D spatiotemporal convolution along both 3D spatial and 1D temporal dimensions. Duerr *et al.* [6] project the input point clouds onto the RV and propose a novel temporal memory alignment strategy to align features between adjacent frames. For object trajectory prediction, the RV-FuseNet [11] consumes multiple LiDAR scans in the RV space and predicts object trajectory on the BEV plane. MVFuseNet [12] fuses the features from the multi-scan BEV and multi-scan RV branches to form the new BEV features for the latter prediction. In contrast to RV-FuseNet and MVFuseNet, our SMVF conducts motion estimation on 3D points instead of BEV grids. Moreover, our SMVF illustrates the complementary properties of the two views, which is not discussed before.

3 Approach

The sequential multi-view fusion network (SMVF) is proposed for LiDAR point motion estimation, as shown in Fig. 2. The SMVF encodes the features in a multi-view manner: the bird’s-eye view (BEV) branch takes a sequence of consecutive LiDAR scans as input for motion information extraction; the range view (RV) branch only uses the current LiDAR scan to extract spatial information for distinguishing nearby points; finally, the features of the point clouds and the two views are fused for per-point motion prediction. Moreover, the sequential instance copy-paste (SICP) is designed to augment more training samples.

The problem definition, multi-view feature encoding paradigm, sequential instance copy-paste, and the overall optimization objectives are illustrated in Sec. 3.1, 3.2, 3.3, and 3.4, respectively.

3.1 Problem Definition

Point motion estimation is an integration of scene flow estimation and moving object segmentation. A motion estimator \mathcal{M} takes a sequence of 3D point clouds

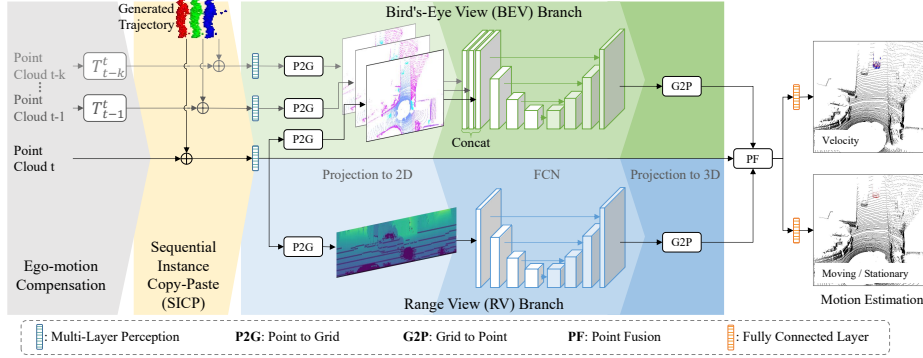


Fig. 2. Overview of the SMVF. First, the input sequential point clouds are adjusted by ego-motion compensation and augmented by the sequential instance copy-paste. Then, efficient and effective feature extraction is achieved by looking from the BEV and RV with 2D FCNs. Finally, features from the original points and the two branches are fused in 3D space to produce per-point motion predictions. Note that the multi-layer perception (MLP) of each LiDAR frame has shared parameters.

$\{\mathbf{p}_t^j\}, \{\mathbf{p}_{t-1}^j\}, \dots, \{\mathbf{p}_{t-k}^j\}$ as inputs (where t is a constant denoting the time stamp, and $k+1$ is the size of the time window, and $j = 1, \dots, N$ denotes the index of the point in a scan with N points), and outputs a set of motion vectors $\{\mathbf{m}_t^j\}$ at time t , as the following,

$$\mathcal{M}(\{\mathbf{p}_t^j\}, \{\mathbf{p}_{t-1}^j\}, \dots, \{\mathbf{p}_{t-k}^j\}) = \{\mathbf{m}_t^j\}, \quad (1)$$

where the motion vector $\mathbf{m}_t^j = (v_x, v_y, v_z, m)$ corresponding to the point \mathbf{p}_t^j includes the velocities (v_x, v_y, v_z) of the point \mathbf{p}_t^j in each axis, and a binary variable m , indicating whether the point is moving ($m = 1$) or not ($m = 0$).

3.2 Multi-View Feature Encoding

The proposed SMVF exploits the multi-view projection-based paradigm for efficient and effective feature encoding. Since object movements in autonomous driving are mostly present on the x - y plane, the BEV that squeezes the z -axis is adequate for encoding temporal motion information. However, there are spatial confusions between points that are projected to the same or nearby BEV grid. Thus, the RV encoding of the current LiDAR scan is required for complementing the spatial information. Both BEV and RV branches have a similar three-step process, including 1) projection to the 2D space (P2G), 2) feature extraction with a fully convolutional network (FCN), and 3) projection back to the 3D space (G2P). For the BEV branch, each LiDAR frame is projected to the BEV grid separately.

Ego-Motion Compensation. Each LiDAR scan is described by its local coordinate system. The proposed method transforms the history LiDAR scans

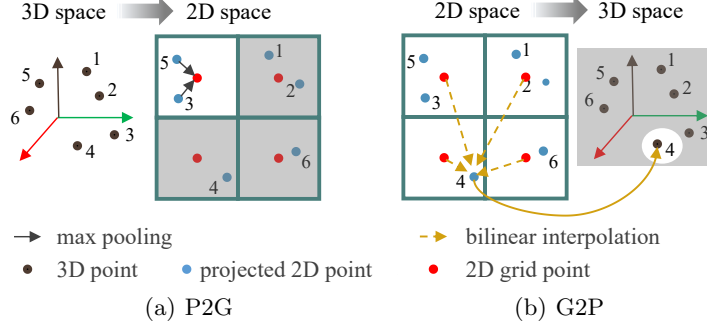


Fig. 3. An illustration of Point to Grid (a) and Grid to Point (b) operations. The white area shows the targets of interest in the output space.

$(t-1, \dots, t-k)$ to the coordinates of the current scan (t), to avoid specious motion estimation caused by ego-motion. The k consecutive relative transformations are represented as transformation matrices $T_{t-1}^t, \dots, T_{t-k}^t$ ($T_i^t \in \mathbb{R}^{4 \times 4}$), and assumed to be known following the common practice.

Point to Grid (P2G). The Point to Grid (P2G) operation transforms the 3D point features to the 2D grid feature maps (*e.g.* BEV, RV), as shown in Fig. 3(a). Each frame of consecutive LiDAR scans applies the same P2G operation separately. Thus, we only illustrate the P2G of the current scan t . The j^{th} 3D point of the current scan t is $\mathbf{p}_t^j = (x_j, y_j, z_j)$, which is first projected onto the 2D grid to acquire the corresponding 2D coordinates (u_j, v_j) . Then the features $\mathcal{F}_{j,c}^{3D}$ of 3D points that fall in the same 2D grid (h, w) , namely $\lfloor u_j \rfloor = h$ and $\lfloor v_j \rfloor = w$, are aggregated by max-pooling to form the 2D grid features $\mathcal{F}_{h,w,c}^{2D}$,

$$\mathcal{F}_{h,w,c}^{2D} = \max_{\forall j \text{ s.t. } \lfloor u_j \rfloor = h, \lfloor v_j \rfloor = w} \mathcal{F}_{j,c}^{3D}. \quad (2)$$

Both BEV and RV are 2D representations. They use a similar P2G operation and just differ in the way of 2D projection. For the BEV projection, it projects the 3D point onto the x - y plane that is discretized by using a rectangular 2D grid $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$ with the predefined width W_{bev} and height H_{bev} . The corresponding 2D coordinates on the BEV are formulated as

$$\begin{pmatrix} u_j \\ v_j \end{pmatrix} = \begin{pmatrix} \frac{x_j - x_{\min}}{x_{\max} - x_{\min}} \times W_{bev} \\ \frac{y_j - y_{\min}}{y_{\max} - y_{\min}} \times H_{bev} \end{pmatrix}. \quad (3)$$

For the RV projection, the 3D point is first mapped to the spherical space (r_j, θ_j, ϕ_j) ,

$$\begin{pmatrix} r_j \\ \theta_j \\ \phi_j \end{pmatrix} = \begin{pmatrix} \sqrt{x_j^2 + y_j^2 + z_j^2} \\ \arcsin\left(\frac{z_j}{\sqrt{x_j^2 + y_j^2 + z_j^2}}\right) \\ \arctan2(y_j, x_j) \end{pmatrix}, \quad (4)$$

where r_j , θ_j , ϕ_j denote the distance, zenith, and azimuth angle, respectively. Then, its corresponding 2D coordinates (u_j, v_j) on the RV are given by quantizing θ_j and ϕ_j but ignoring r_j as the following,

$$\begin{pmatrix} u_j \\ v_j \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(1 - \phi_j \pi^{-1}) W_{rv} \\ [1 - (\theta_j + f_{up}) f^{-1}] H_{rv} \end{pmatrix}, \quad (5)$$

where $f = f_{up} + f_{down}$ is the LiDAR vertical field-of-view, and W_{rv} and H_{rv} are the width and height of the RV.

Fully Convolutional Network (FCN). For efficiency, both BEV and RV branches adopt a similar 2D fully convolutional network (FCN) to extract meaningful features. The RV branch only takes the current LiDAR scan as input to provide spatial information and the input size of RV is $C \times H_{rv} \times W_{rv}$, where C denotes the number of feature channels. For the BEV branch, the features from consecutive LiDAR scans are concatenated to form a $(kC + C) \times H_{bev} \times W_{bev}$ input tensor, where k denotes the number of history LiDAR scans. The detailed architecture of the FCN can be seen in the supplementary material.

Grid to Point (G2P). On the contrary to the Point to Grid (P2G) operation, the Grid to Point (G2P) operation transfers the features from the 2D grid to the 3D point for the latter point-level prediction, as shown in Fig. 3(b). The features of the j^{th} 3D point \mathbf{p}_t^j can be obtained by bilinear interpolation within the four-neighbor grids of its corresponding 2D position (u_j, v_j) as follows,

$$\mathcal{F}_{j,c}^{3D} = \sum_{p=0}^1 \sum_{q=0}^1 \omega_{p,q,j} \mathcal{F}_{\lfloor u_j \rfloor + p, \lfloor v_j \rfloor + q, c}^{2D}, \quad (6)$$

where $\omega_{p,q,j} = (1 - |u_j - (\lfloor u_j \rfloor + p)|)(1 - |v_j - (\lfloor v_j \rfloor + q)|)$ denotes the bilinear interpolation weight. The neighbor grids beyond the 2D grid range are regarded as all zeros.

Point Fusion. The point fusion (PF) module fuses the features from the 3D points and the projected features from the BEV and RV to form a final per-point estimation. The PF serves as a mid-fusion module and allows end-to-end training of the proposed SMVF. For efficiency, it only adopts a feature concatenation operation and two MLP layers for feature fusion. Finally, two additional fully connected (FC) layers predict the velocity and segmentation results, respectively. By using this simple fusion module, the network automatically learns motion information from the BEV branch, spatial information from the RV branch, and other complementary information from the original point clouds.

3.3 Sequential Instance Copy-Paste

The motion estimator usually presents lower confidence on the objects with fewer or even no training samples (*e.g. cyclist, dog, toy car*). Inspired by the previous 3D detection methods [26], [13] that adopt the instance copy-paste strategy to improve the performance of rare classes (*e.g. pedestrian and cyclist*), we extend the instance copy-paste from a single frame to multiple frames as shown in Fig. 4.

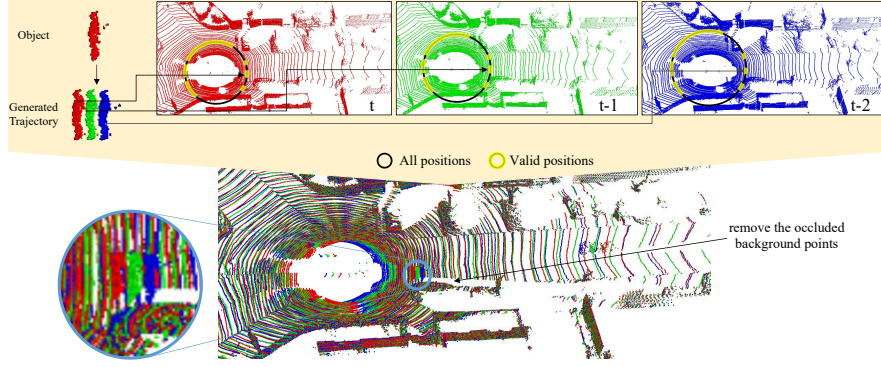


Fig. 4. Overview of the proposed sequential instance copy-paste (SICP). The augmented LiDAR sequence is shown with all scans overlapped. Note that all LiDAR scans have been compensated with ego-motion.

First, it constructs an object bank, consisting of the objects cropped from their original LiDAR scans according to the annotated 3D bounding boxes. Then, it uniformly samples a category and an object from this category. Finally, an object trajectory is generated by inserting this object to the LiDAR sequence.

Specifically, the object is assumed to be moving along its 3D bounding box heading yaw with a random sampled velocity \tilde{v} on the x - y plane. The velocities \tilde{v}_x , \tilde{v}_y of an inserted object are formulated by

$$\begin{pmatrix} \tilde{v}_x \\ \tilde{v}_y \end{pmatrix} = \begin{pmatrix} \tilde{v} \cos(yaw) \\ \tilde{v} \sin(yaw) \end{pmatrix}. \quad (7)$$

\tilde{v} is sampled uniformly within a predefined range of each category and the object is moving backward when \tilde{v} is negative. Its speed \tilde{v}_z along the z -axis is calculated by ensuring that the object is moving on the ground.

Then, the position of an inserted object trajectory is determined by first filtering out the infeasible positions and then randomly sampling from the remaining candidates. The inserted object has a distance \tilde{r}_k to the sensor in its original LiDAR scan. Given that the LiDAR point density is changed across different distances, the candidate positions in the t^{th} scan are kept, only when their distance to the sensor are the same as \tilde{r}_k (marked as a black circle in Fig. 4). Then, the corresponding positions in the history LiDAR scans can be derived from the above velocity $(\tilde{v}_x, \tilde{v}_y, \tilde{v}_z)$. A candidate position of the trajectory is filtered out if the inserted object cannot be placed on the ground or there is occlusion between the inserted object and the existing objects in any time step. Finally, the inserted position is randomly selected from the remaining candidates (marked as transparent yellow in Fig. 4) and the occluded background points are removed. Different from the previous methods [26], [13], the SICP specially considers the inserted position and the ray occlusion to ensure the reality of the augmented LiDAR sequence.

3.4 Optimization Objectives

For the moving object segmentation (MOS) task, the 3D points are classified into two categories, namely moving or stationary. Therefore, we apply the commonly-used cross-entropy (CE) loss, which can be formulated as,

$$\mathcal{L}_{CE} = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_n^c \log(\hat{y}_n^c), \quad (8)$$

where y_n^c ($y_n^c \in \{0, 1\}$) and \hat{y}_n^c ($\hat{y}_n^c \in [0, 1]$) are the ground-truth label and the predicted probability of the c^{th} class on the n^{th} point. To facilitate more accurate classification on hard samples, another loss term $\mathcal{L}_{CE}^{20\%}$ only considers the top 20% points with higher losses. In addition, the Lovász-Softmax loss [2] \mathcal{L}_{LS} is also adopted to directly optimize the Intersection over Union (IoU) metric. The total loss function for the MOS task is defined as

$$\mathcal{L}_{mos} = \mathcal{L}_{CE} + 4\mathcal{L}_{CE}^{20\%} + 3\mathcal{L}_{LS}. \quad (9)$$

For the scene flow estimation task, the L_2 error between the ground-truth velocities \tilde{v}_n and the predicted velocities v_n is used as the following,

$$\mathcal{L}_{sf} = \frac{1}{N} \sum_{n=1}^N \|v_n - \tilde{v}_n\|_2. \quad (10)$$

If a dataset simultaneously provides benchmarks of the two tasks, \mathcal{L}_{mos} and \mathcal{L}_{sf} are both adopted for the guidance of corresponding predictions.

4 Experiments

We conduct ablation studies and evaluate the performance of the proposed SMVF on the SemanticKITTI [1] benchmark for moving object segmentation and the Waymo Open Dataset [10] for scene flow estimation.

4.1 Datasets and Evaluation Metrics

SemanticKITTI. The SemanticKITTI [1] derives from the odometry dataset of the KITTI Vision Benchmark [8]. It contains 43,552 360° LiDAR scans from 22 sequences collected in a city of Germany. The training set (19,130 scans) consists of sequences from 00 to 10 except 08, and the sequence 08 (4,071 scans) is used for validation. The rest sequences (20,351 scans) from 11 to 21 are only provided with LiDAR point clouds and are used for the online leaderboard. This dataset is labeled with 28 classes. For the moving object segmentation (MOS) task, the same splits for training, validation, and test sets are used, while all classes are reorganized into only two types: moving and non-moving/static [3].

As the official guidance [3] suggests, we adopt the Intersection over Union (IoU) [7] on the **moving** class to evaluate the proposed SMVF and its competitors. It can be formulated as

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \quad (11)$$

where TP, FP, and FN correspond to the true positive, false positive, and false negative of the **moving** points.

Waymo Open Dataset. The Waymo Open Dataset is a large-scale dataset for autonomous driving perception across a diverse and rich domain [10], [20]. The scene flow labels are generated by leveraging the human-annotated tracked 3D objects in this dataset [20]. The training split consists of 800 run segments containing 158,081 frames in total, while the validation split contains 200 run segments with 39,987 frames.

The commonly-used metrics for scene flow estimation are the mean L_2 error of point-wise flow and the percentage of predictions with L_2 error below a given threshold [23], [14]. Considering that objects in autonomous driving have different speed distributions dictated by the object category (*e.g. vehicle, pedestrian*), prediction performances delineated by the object category are also reported.

4.2 Experimental Setup

Network Setup. The input point features contain $x, y, z, intensity, r, \Delta x$ and Δy , where Δx and Δy denote the offsets to the corresponding BEV grid center. For the Waymo Open Dataset, *elongation* is used as an additional feature. As shown in Fig. 2, the first MLP layer of each LiDAR frame has shared parameters and outputs 64 feature channels, which the following P2G operation transforms to the BEV and RV feature maps, respectively. Both BEV and RV branches utilize a similar 2D FCN network with three down-sampling and two up-sampling stages, but the RV does not apply down-sampling along the height dimension. The PF module with only two MLP layers takes features from the three sources as inputs and outputs 64 feature channels. Finally, two FC layers are used for motion state prediction and scene flow estimation, respectively.

Data Augmentation. During training, we apply the widely-used data augmentation strategies, including random rotation around the z -axis, random global scale sampled from $[0.95, 1.05]$, random flipping along x and y axes, and random Gaussian noise $\mathcal{N}(0, 0.02)$. Besides, the proposed SICP is adopted.

4.3 Results on the SemanticKITTI

Moving object segmentation is evaluated on the SemanticKITTI. The 3D space is set to be within $[-50, 50]$ m for the x and y axes and $[-2, 4]$ m for the z -axis. For the BEV branch, the input resolutions are $W_{bev} = 512$ and $H_{bev} = 512$. For the RV branch, the input resolutions are $W_{rv} = 2048$ and $H_{rv} = 64$. Our SMVF is trained from scratch for 48 epochs with a batch size of 24, taking

Table 1. Moving object segmentation results on the SemanticKITTI validation and test set. k denotes the number of history LiDAR scans. * means the re-implementation based on its official code. “TRT16” means the TensorRT FP16 inference.

Method	k	val IoU	test IoU	Runtime
KPConv [21]	1	–	60.9	168 ms
SpSequenceNet [18]	1	–	43.2	450 ms
Chen <i>et al.</i> [3]	1	59.9	52.0	24.3 ms
SMVF [ours]	1	75.1	72.1	21.8 ms
Chen <i>et al.</i> [3]	8	66.5	62.5	24.8 ms
MotionNet* [25]	4	67.8	62.6	78.9 ms
SMVF [ours]	2	76.9	75.9	24.3 ms
SMVF [ours]; TRT16	2	77.0	75.9	13 ms

around 15 hours on 8 NVIDIA RTX 2080Ti GPUs. Stochastic gradient descent (SGD) serves as the optimizer with a weight decay of 0.001 and a learning rate initialized to 0.02, which is decayed by 0.1 every 10 epochs.

Comparison With the State of the Art. All methods are evaluated on the SemanticKITTI validation and test set with the official evaluation code. In Table 1, our SMVF outperforms the previous methods by remarkable margins. By utilizing the current LiDAR scan and one history LiDAR scan ($k = 1$), the point-based KPConv [21] and the sparse voxel-based SpSequenceNet [18] are $\times 7.7$ and $\times 20.6$ slower than our SMVF, respectively, and they also perform worse with -11.2 and -28.9 IoU drops on the test set, respectively. Chen *et al.* [3] ($k = 8$) based on the RV representation, runs as fast as our SMVF ($k = 2$), but it performs much worse with -10.4 and -13.4 IoU drops on the validation and test set, respectively. We re-implement the BEV-based MotionNet [25] on the SemanticKITTI based on its official code. For a fair comparison, the scope and partition of the 3D space are the same as our SMVF, while it divides 20 bins along the z -axis. The point-level prediction is acquired by its corresponding BEV grid prediction. Our SMVF ($k = 2$) outperforms the MotionNet on both IoU metric and inference speed. There are three primary reasons: 1) the MotionNet is a BEV-based method that cannot distinguish the 3D points on the same BEV grid; 2) the MotionNet encodes the BEV features by the binary voxels that lead to information loss; 3) the MotionNet adopts a expensive spatiotemporal network to extract motion information. Besides, our SMVF can be easily deployed with TensorRT FP16 inference mode and runs only 13 ms, supporting real-time perception tasks in autonomous driving.

Ablation Studies. The settings for the multi-view framework and the number of history LiDAR scans are evaluated on the SemanticKITTI validation set. As shown in Table 2, we can discover that: 1) the BEV branch is more suitable for extracting motion information than the RV branch (a,c and b,d); 2) the multi-view framework performs better than the single-view one (a,b and c,d); 3) based on our SMVF, the RV branch does not need to extract temporal information

Table 2. Ablative analysis on the SemanticKITTI validation set. “-”: the branch is not applied. “multi”: the branch takes multiple LiDAR scans as input. “single”: the branch only uses the current LiDAR scan. k : the number of history LiDAR scans.

(a) Multi-view framework.						(b) Number of history LiDAR scans.					
	BEV	RV	k	IoU	Runtime		BEV	RV	k	IoU	Runtime
a)	-	multi	2	63.8	14.4 ms	f)	multi	single	1	75.1	21.8 ms
b)	single	multi	2	66.5	22.3 ms	d)	multi	single	2	76.9	24.3 ms
c)	multi	-	2	73.5	15.9 ms	g)	multi	single	3	76.8	26.5 ms
d)	multi	single	2	76.9	24.3 ms	h)	multi	single	4	76.9	29.0 ms
e)	multi	multi	2	76.7	28.1 ms						

(d,e); 4) the segmentation performance rises up when more history LiDAR scans ($k \leq 2$) are incorporated, but the performance saturates when $k > 2$ (f,d,g,h).

4.4 Results on the Waymo Open Dataset

The scene flow estimation task is conducted on the Waymo Open Dataset. The same configuration as FastFlow3D [10] is adopted. The 3D space is limited to $[-85, 85]$ m for the x and y axes and $[-3, 3]$ m for the z -axis. For the BEV branch, the input resolutions are $W_{bev} = 512$ and $H_{bev} = 512$. For the RV branch, the input resolutions are $W_{rv} = 2560$ and $H_{rv} = 64$. The proposed SMVF is trained from scratch for 24 epochs with a batch size of 24, taking around 40 hours on 8 NVIDIA RTX 2080Ti GPUs. The optimizer is AdamW [16] with a weight decay of 0.001 and an initial learning rate of 0.002, which is decayed by 0.1 every 10 epochs. The SICP constructs the object bank on the training split by the annotated 3D bounding boxes and it inserts 10 object trajectories into each training sample consisting of $k + 1$ LiDAR scans.

Comparison With the State of the Art. As shown in Table 3, the point-based FlowNet3D [14] over-fits on the stationary class, and our SMVF surpasses it by a large margin for all the other metrics, while our SMVF runs $\times 5.7$ faster. Compared with the FastFlow3D [10], our SMVF achieves higher performance especially for the mean L_2 error and the percentage of predictions with L_2 error ≤ 1.0 m/s. Besides, our SMVF shows obvious superiority for *Background* and the class with fewer training samples (*e.g. Cyclist*), demonstrating its generalization ability for various classes. We also re-implement the BEV-based MotionNet [25] based on its official code and assign the predicted BEV grid velocities to the corresponding LiDAR points. The scope and partition of the 3D space are the same as our SMVF, while it divides 20 bins along the z -axis. Our SMVF outperforms the MotionNet on most metrics and inference speed for the same reason in Section 4.3. Moreover, we integrate the proposed SICP into the MotionNet and SMVF, respectively, and it improves the performance across all categories, especially *Cyclist* that has fewer training samples.

Ablation Studies. The ablative analysis of the multi-view framework on the Waymo Open Dataset is shown in Table 4. All methods here do not adopt the

Table 3. Scene flow performance comparison on the validation split of the Waymo Open Dataset. k denotes the number of history LiDAR scans. * means the re-implementation based on its official code. A: All. M: Moving. S: Stationary.

Method	k	SICP	Metric	Vehicle			Pedestrian			Cyclist			Background	Runtime
				A	M	S	A	M	S	A	M	S		
FlowNet3D* [14]	1		mean (m/s) ↓	1.90	7.25	0.04	0.92	1.34	0.09	3.49	3.94	0.09	0.00	172 ms
			≤ 0.1 m/s ↑	67.4%	0.0%	90.9%	24.3%	0.0%	72.9%	9.2%	0.0%	79.3%	99.9%	
			≤ 1.0 m/s ↑	75.9%	7.0%	99.9%	44.9%	17.3%	99.9%	12.6%	1.1%	99.9%	99.9%	
FastFlow3D [10]	1		mean (m/s) ↓	0.18	0.54	0.05	0.25	0.32	0.10	0.51	0.57	0.10	0.07	32.5 ms
			≤ 0.1 m/s ↑	70.0%	11.6%	90.2%	33.0%	14.0%	71.4%	13.4%	4.8%	78.0%	95.7%	
			≤ 1.0 m/s ↑	97.7%	92.8%	99.4%	96.7%	95.4%	99.4%	89.5%	88.2%	99.6%	96.7%	
SMVF [ours]	1		mean (m/s) ↓	0.17	0.51	0.05	0.23	0.29	0.11	0.38	0.42	0.10	0.028	30 ms
			≤ 0.1 m/s ↑	67.7%	10.6%	90.4%	30.1%	13.3%	68.6%	14.6%	6.7%	75.1%	98.9%	
			≤ 1.0 m/s ↑	97.9%	92.9%	99.6%	98.2%	97.5%	99.6%	95.6%	95.1%	99.7%	99.5%	
SMVF [ours]	1	✓	mean (m/s) ↓	0.16	0.46	0.05	0.23	0.28	0.11	0.35	0.38	0.11	0.028	30 ms
			≤ 0.1 m/s ↑	67.8%	10.2%	89.8%	29.4%	11.7%	67.9%	15.3%	7.7%	73.2%	99.0%	
			≤ 1.0 m/s ↑	97.9%	92.8%	99.7%	98.1%	97.4%	99.5%	97.0%	96.8%	99.3%	99.5%	
MotionNet* [25]	4		mean (m/s) ↓	0.20	0.64	0.04	0.20	0.25	0.10	0.35	0.38	0.09	0.05	79.1 ms
			≤ 0.1 m/s ↑	68.8%	8.9%	89.7%	34.8%	17.6%	69.4%	15.6%	7.3%	77.2%	97.2%	
			≤ 1.0 m/s ↑	97.5%	90.5%	99.8%	98.8%	98.3%	99.8%	96.4%	95.9%	99.9%	97.9%	
MotionNet* [25]	4	✓	mean (m/s) ↓	0.19	0.61	0.04	0.20	0.24	0.10	0.31	0.34	0.09	0.05	79.1 ms
			≤ 0.1 m/s ↑	69.0%	9.0%	89.9%	34.4%	16.7%	69.7%	16.5%	8.2%	78.0%	96.9%	
			≤ 1.0 m/s ↑	97.3%	90.1%	99.8%	98.8%	98.3%	99.8%	97.8%	97.5%	99.9%	97.6%	
SMVF [ours]	4	✓	mean (m/s) ↓	0.14	0.41	0.04	0.19	0.24	0.10	0.29	0.31	0.09	0.027	38.4 ms
			≤ 0.1 m/s ↑	69.6%	12.1%	89.6%	35.5%	18.2%	70.0%	19.1%	11.2%	78.2%	99.2%	
			≤ 1.0 m/s ↑	98.5%	94.7%	99.8%	98.9%	98.4%	99.7%	97.9%	97.6%	99.8%	99.6%	

Table 4. Multi-view framework analysis on the validation split of the Waymo Open Dataset with $k = 1$. A: All. M: Moving. S: Stationary.

	BEV	RV	Metric	Vehicle			Pedestrian			Cyclist			Background	Runtime
				A	M	S	A	M	S	A	M	S		
a)	multi	-	mean (m/s) ↓	0.18	0.53	0.05	0.24	0.30	0.12	0.40	0.44	0.10	0.04	17.0 ms
			≤ 0.1 m/s ↑	67.6%	9.7%	87.9%	30.5%	12.3%	66.9%	14.7%	6.7%	75.1%	97.5%	
			≤ 1.0 m/s ↑	97.7%	92.2%	99.7%	98.2%	97.3%	99.6%	94.6%	94.0%	99.6%	98.3%	
b)	-	multi	mean (m/s) ↓	0.70	2.48	0.08	0.58	0.77	0.19	1.45	1.62	0.18	0.06	16.5 ms
			≤ 0.1 m/s ↑	64.8%	2.5%	86.6%	22.6%	3.48%	60.7%	8.29%	1.19%	62.2%	97.9%	
			≤ 1.0 m/s ↑	86.9%	52.3%	99.0%	77.6%	67.7%	97.3%	52.6%	46.6%	98.3%	98.9%	
c)	multi	single	mean (m/s) ↓	0.17	0.51	0.05	0.23	0.29	0.11	0.38	0.42	0.10	0.028	30 ms
			≤ 0.1 m/s ↑	67.7%	10.6%	90.4%	30.1%	13.3%	68.6%	14.6%	6.7%	75.1%	98.9%	
			≤ 1.0 m/s ↑	97.9%	92.9%	99.6%	98.2%	97.5%	99.6%	95.6%	95.1%	99.7%	99.5%	
d)	multi	multi	mean (m/s) ↓	0.17	0.52	0.05	0.24	0.30	0.12	0.39	0.42	0.11	0.028	34.8 ms
			≤ 0.1 m/s ↑	67.8%	9.81%	88.1%	29.9%	12.0%	65.6%	15.1%	7.24%	74.4%	99.0%	
			≤ 1.0 m/s ↑	97.9%	92.8%	99.7%	98.2%	97.5%	99.6%	94.8%	94.3%	99.1%	99.5%	

proposed SICP. Compared with the BEV-only model, the performance of the RV-only model drops drastically on the moving points (a,b). Moreover, when the multi-view fusion framework is adopted, the performance drops, if the RV branch uses multiple LiDAR scans instead of a single scan (c,d). These observations are similar to those of the SemanticKITTI in Table 2. It proves that the distorted object trajectories in the RV space are not helpful for extracting motion information, while a single LiDAR scan in the RV space is enough to compensate for the spatial information loss in the BEV space. More ablation studies can be found in the supplementary material.

Generalization of Scene Flow. As illustrated in FastFlow3D [10], the point-level ground-truth velocities are obtained by the tracked 3D bounding boxes.

Table 5. Generalization of scene flow on the validation split of the Waymo Open Dataset with $k = 1$.

Method	SICP		<i>Cyclist</i>			<i>Pedestrian</i>		
			mean error (m/s)↓			mean error (m/s)↓		
			A	M	S	A	M	S
FastFlow3D [10]		supervised	0.51	0.57	0.10	0.25	0.32	0.10
		stationary	1.13	1.24	0.06	0.90	1.30	0.10
		ignored	0.83	0.93	0.06	0.88	1.25	0.10
SMVF		supervised	0.38	0.42	0.10	0.23	0.29	0.11
		stationary	0.49	0.54	0.10	0.79	1.15	0.09
		ignored	0.47	0.53	0.10	0.57	0.80	0.09
SMVF	✓	supervised	0.35	0.38	0.11	0.23	0.28	0.11
		stationary	0.35	0.37	0.11	0.70	1.01	0.09
		ignored	0.34	0.37	0.11	0.36	0.48	0.10

However, some objects (*e.g. dog, toy car*) do not have any annotated 3D bounding boxes. For a reliable autonomous driving system, the ability to predict accurate motion velocity should generalize well to these objects, even though they are treated to be “stationary” or “ignored” during training. To quantitatively evaluate the generalization ability, we selectively ablate the categories for *pedestrian* and *cyclist* during training in two ways: 1) “stationary” treats the ablated category to be stationary; 2) “ignored” treats the ablated category to be ignored. On the Waymo Open Dataset, the number of *cyclist* samples is much fewer than that of *pedestrian* samples. Generally, the objects (*e.g. dog, toy car*) that have no annotated 3D bounding boxes have a limited number of samples, similar to or even fewer than that of the *cyclist*. As shown in Table 5, we can discover that: 1) the performance gets better when the ablated category is labeled as “ignored” rather than “stationary”; 2) our SMVF shows a much stronger generalization ability than the FastFlow3D; 3) the proposed SICP significantly improves the performance for the two categories. It indicates that the SICP can be used to generate reliable training samples for these unlabeled objects and promotes the model to be comparable with that in the supervised manner.

5 Conclusion

A novel SMVF is proposed for fast LiDAR point motion estimation, where features from the 3D points, the BEV, and the RV are assigned complementary roles and fused for accurate prediction. The SICP is designed to augment the training LiDAR sequences and improve the generalization ability for the category with fewer training samples. Experimental results on the SemanticKITTI dataset and the Waymo Open Dataset demonstrate the effectiveness and superiority of the proposed components. It can be observed that the information loss during the 2D projection can be remedied by using the multi-view fusion framework and the BEV branch is more suitable for extracting motion features, while the RV can be used to provide the complementary spatial information.

References

1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9297–9307 (2019)
2. Berman, M., Triki, A.R., Blaschko, M.B.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4413–4421 (2018)
3. Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., Stachniss, C.: Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters* **6**(4), 6529–6536 (2021)
4. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019)
5. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2758–2766 (2015)
6. Duerr, F., Pfaller, M., Weigel, H., Beyerer, J.: Lidar-based recurrent 3d semantic segmentation with temporal memory alignment. In: 2020 International Conference on 3D Vision (3DV). pp. 781–790. IEEE (2020)
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* **88**(2), 303–338 (2010)
8. 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)
9. 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)
10. Jund, P., Sweeney, C., Abdo, N., Chen, Z., Shlens, J.: Scalable scene flow from point clouds in the real world. *IEEE Robotics and Automation Letters* (2021)
11. Laddha, A., Gautam, S., Meyer, G.P., Vallespi-Gonzalez, C., Wellington, C.K.: Rv-fuseNet: Range view based fusion of time-series lidar data for joint 3d object detection and motion forecasting. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 7060–7066. IEEE (2020)
12. Laddha, A., Gautam, S., Palombo, S., Pandey, S., Vallespi-Gonzalez, C.: Mv-fuseNet: Improving end-to-end object detection and motion forecasting through multi-view fusion of lidar data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2865–2874 (2021)
13. 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)
14. 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)

15. 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)
16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017)
17. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4161–4170 (2017)
18. Shi, H., Lin, G., Wang, H., Hung, T.Y., Wang, Z.: Spsequencenet: Semantic segmentation network on 4d point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 4574–4583 (2020)
19. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 8934–8943 (2018)
20. 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)
21. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 6411–6420 (2019)
22. Vanholder, H.: Efficient inference with tensorrt. In: *GPU Technology Conference*. vol. 1, p. 2 (2016)
23. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2589–2597 (2018)
24. 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)
25. Wu, P., Chen, S., Metaxas, D.N.: 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*. pp. 11385–11395 (2020)
26. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. *Sensors* **18**(10), 3337 (2018)