2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

InsMOS: Instance-Aware Moving Object Segmentation in LiDAR Data

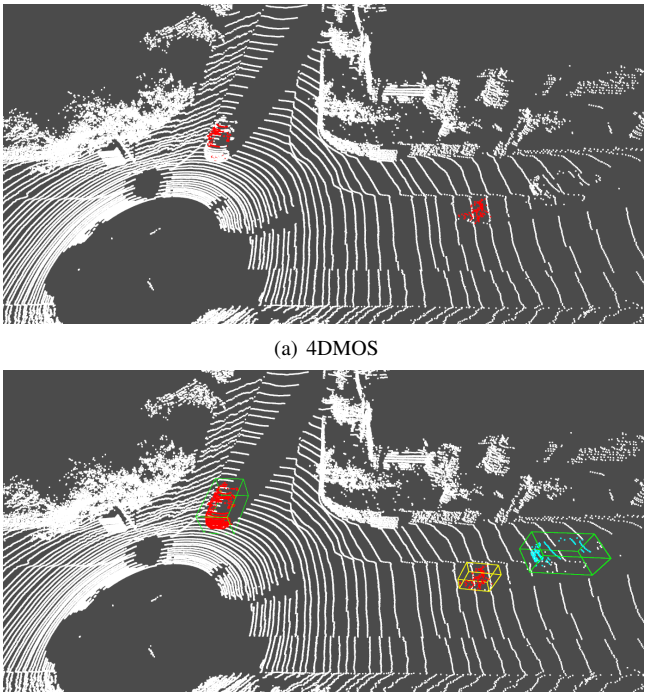
Neng Wang Chenghao Shi Ruibin Guo Huimin Lu* Zhiqiang Zheng Xieyuanli Chen*

***Abstract*— Identifying moving objects is a crucial capability for autonomous navigation, consistent map generation, and future trajectory prediction of objects. In this paper, we propose a novel network that addresses the challenge of segmenting moving objects in 3D LiDAR scans. Our approach not only predicts point-wise moving labels but also detects instance information of main traffic participants. Such a design helps determine which instances are actually moving and which ones are temporarily static in the current scene. Our method exploits a sequence of point clouds as input and quantifies them into 4D voxels. We use 4D sparse convolutions to extract motion features from the 4D voxels and inject them into the current scan. Then, we extract spatio-temporal features from the current scan for instance detection and feature fusion. Finally, we design an upsample fusion module to output point-wise labels by fusing the spatio-temporal features and predicted instance information. We evaluated our approach on the LiDAR-MOS benchmark based on SemanticKITTI and achieved better moving object segmentation performance compared to state-of-the-art methods, demonstrating the effectiveness of our approach in integrating instance information for moving object segmentation. Furthermore, our method shows superior performance on the Apollo dataset with a pre-trained model on SemanticKITTI, indicating that our method generalizes well in different scenes. The code and pre-trained models of our method will be released at `https://github.com/nubot-nudt/InsMOS`.**

I. Introduction

3D LiDAR-based environment perception is important for autonomous navigation systems due to its robustness to illuminational changes and large field of view [1]. However, moving objects can often disrupt LiDAR perception, leading to suboptimal performance in downstream tasks, such as point cloud registration [2], simultaneous localization and mapping (SLAM) [3], [4], static map creation [5], [6], and path planning [7], [8]. Therefore, identifying moving objects in LiDAR data is one of the most critical abilities for LiDAR-based applications of autonomous mobile systems.

In this work, we aim to tackle the challenge of segmenting moving objects from static environments using consecutive LiDAR point clouds. In order to achieve moving object segmentation (MOS) in LiDAR data, it is usually necessary to extract temporal features from the sequence of 3D LiDAR scans. Such features are used to determine whether object's position has changed compared to the previous observations. Many existing methods apply 2D convolutions to extract temporal features from residual range images [1], [9], [10] generated by spherical projection. These methods are usually fast in operation but relatively generalizing poorly in unseen environments. Some non-projection-based methods [11], [12] are also able to extract temporal features directly from the sequence of point clouds. For example, Mersch et al. [11], [13] detect moving objects using directly point cloud maps. Besides, other approaches in different domains can be used to tackle MOS task, such as scene flow estimation [14], [15], [16] and map cleaning [5], [6], [17], [18], [19], [20].



(a) 4DMOS

(b) Ours

Fig. 1: The comparison of moving object segmentation between our approach and 4DMOS. (a) 4DMOS only segments part of the LiDAR points of a moving object due to the lack of instance information. (b) Our approach detects all movable objects and distinguishes their category, displayed with different colored bounding boxes. By utilizing the instance information, we can completely segment a moving instance (red) and determine which instance is currently static (cyan).

Different from existing approaches that classify moving objects on a point-by-point basis, our method focuses more on a higher instance level and decides which instances are currently moving in the scene. As illustrated in Fig. 1, our approach predicts both point-wise moving labels and instance information within the current point cloud, using a bottom-up and top-down fusion fashion, which is more natural and can generalize to different setups. Moreover, this instance information can also be helpful for subsequent tasks, such

All authors are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China.
*corresponding authors, {lhmnew, xieyuanli.chen}@nudt.edu.cn
This work was supported in part by the National Science Foundation of China under Grant U1913202, U22A2059, and 62203460, as well as Major Project of Natural Science Foundation of Hunan Province under Grant 2021JC0004.

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10342277

as path planning [7], [8] and future state predictions [21].

The main contribution of this paper is a novel network that can efficiently and accurately segment moving objects in 3D LiDAR data while detecting instance information of common movable objects, such as pedestrians and vehicles. We use 4D sparse convolutions [22] to extract motion features from 4D voxels created by quantifying the consecutive input scans and inject the motion features into the current scan. Then, we concatenate the motion features with the original point features and use an instance detection module to extract spatio-temporal features from the current scan for instance detection and feature fusion. Finally, an upsample fusion module is designed to fuse the spatio-temporal features and instance information for achieving better segmentation within the instances. To further improve the performance, we propose an instance-based refinement algorithm to refine the network predictions and determine which instances are actually moving. Based on instance-level understanding, our method achieves state-of-the-art performance on SemanticKITTI dataset and generalizes well to Apollo dataset without any fine-tuning.

In sum, we make three key claims: Our approach is able to (i) predict instances and segment moving objects simultaneously; (ii) achieve the state-of-the-art MOS performance compared with existing methods; (iii) generalize well to different scenes without any fine-tuning. These claims are supported by our experimental evaluation.

II. Related Work

Existing MOS methods for 3D LiDAR data can be divided into two groups, projection-based and non-projection-based approaches.

Projection-based approaches. Many existing methods convert the 3D raw point cloud into a 2D image plane, such as range images [1], [9], [10], [23] and Bird's Eye View (BEV) images [14], [24], [25] to facilitate online LiDAR point cloud processing. To extract temporal features, they usually need to pre-process the past consecutive projected image representations. Chen et al. [9] first exploit the residual range images for online MOS. Their proposed method gets rid of the restraint of the pre-built maps and can be directly applied in a SLAM pipeline. Based on that, Sun et al. [10] design a dual-branch structure to extract spatial and temporal features separately, and then fuse them with motion-guided attention modules. By applying a coarse-to-fine network architecture, artifacts generated in the back-projection can be effectively reduced. Unlike [9], [10], Kim et al. [1] argue that semantic information is helpful for MOS task, so they insert semantic cues into the network and achieve better segmentation performance. Different from the spherical projection, BEV images are usually generated by compressing 3D point clouds in the height direction, which is more intuitive for observing the movement of objects. Mohapatra et al. [24] propose a lightweight network for processing BEV images and achieving real-time MOS on an embedded platform. However, although their method performs fast, they can only segment moving objects in BEV space with relatively low accuracy. Wu et al. [25] represent a given sequence of 3D LiDAR scans into BEV maps and then extract spatio-temporal features from the maps to estimate the motion status of objects. They can not only segment moving objects at the current moment but also predict the future trajectories. However, their method can only be performed in BEV space and cannot provide point-wise predictions. In general, such projection-based methods suffer from the loss of information and fixed pattern of the trained data, thus usually unable to generalize well into new setups and environments.

Non-projection-based approaches. In order to improve the accuracy and generalization ability, some methods tackle the MOS task by directly processing 3D point clouds. Mersch et al. [11] propose 4DMOS that exploits sparse 4D convolutions to extract spatio-temporal features from the past consecutive point clouds and predict point-wise moving confidence scores. Their approach achieves a fine balance between performance and efficiency. Following the 4DMOS, Kreutz et al. [12] design a 4D LiDAR representation to identify moving objects by self-supervised learning, but it only works in stationary settings.

In addition to the above mentioned, some other methods can also segment moving objects in 3D LiDAR scans, including 3D scene flow estimation [14], [15], [16] and map cleaning [5], [6], [17], [18], [26]. The main purpose of 3D scene flow estimation is to predict a 3D displacement for each point between two consecutive input point clouds, which can be used to determine whether a point is moving. However, their performance on the MOS task is often restricted due to the short input time horizon. Identifying moving objects through map cleanup requires first building a scene map, e.g., Pomerleau et al. [17] maintain a global map, and segment moving points and static points based on repeated observations. Similarly, Pagad et al. [18] create an occupancy map for detecting and removing moving objects in LiDAR scans. Most recently, Lim et al. [26] propose ERASOR achieving good map cleaning results. However, these methods are time-consuming and usually run offline.

Unlike previous approaches, we insert instance information into our network to completely segment a moving object. We extract motion features from the sequence point clouds and inject them into each point in the current scan. And then we extract spatio-temporal features from the current scan for instance detection and feature fusion. Finally, we design an upsample fusion module to integrate spatio-temporal features and instance information and output point-wise moving labels of the current scan. Besides, we also propose an instance-based refinement to further refine the predictions of our network. To the best of our knowledge, our method is the first work to explicitly exploit instance information for LiDAR MOS task.

III. Our Approach

A. Preliminaries

Given the current LiDAR scan $\mathcal{S}_0 = \{\boldsymbol{p}_i \in \mathbb{R}^4\}_{i=1}^{M}, \boldsymbol{p}_i = [x_i, y_i, z_i, 1]^\mathsf{T}$ with M points represented as homogeneous coordinates and the past $N-1$ consecutive scans

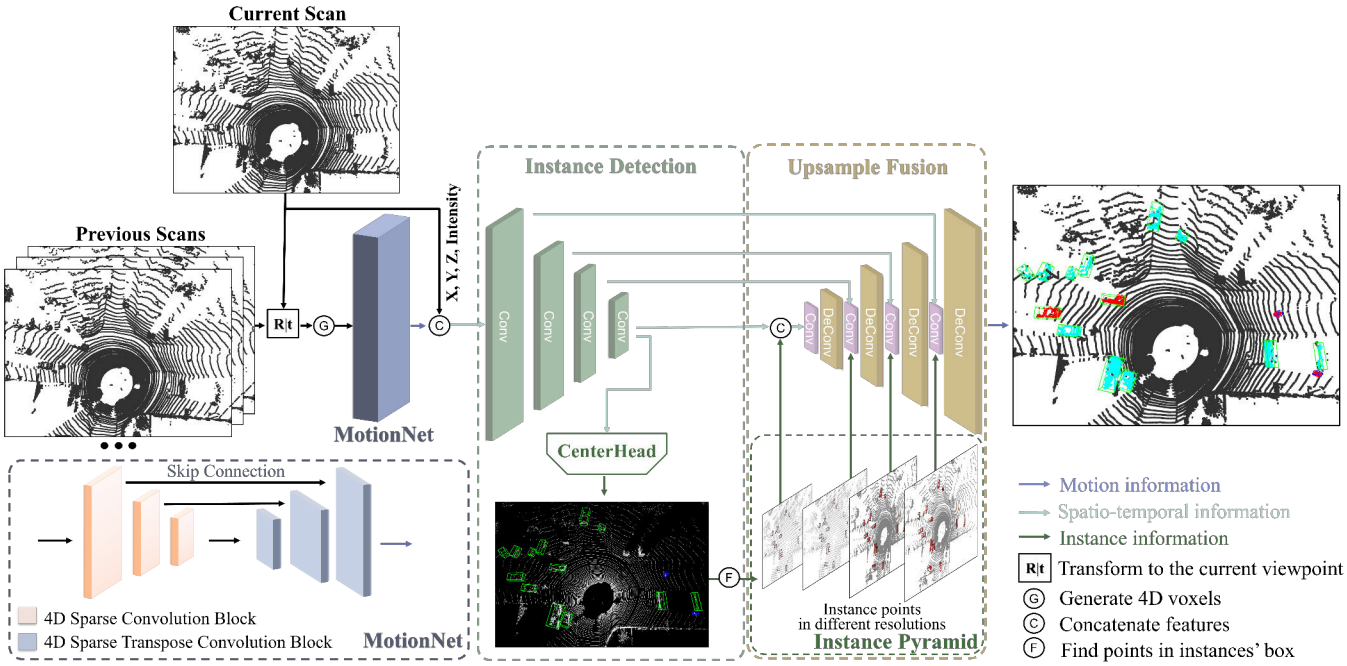


Fig. 2: Overview of our network. MotionNet mainly extracts motion features. Instance Detection Module extracts spatio-temporal features and detects instances. Upsample Fusion Module is applied to fuse the spatio-temporal and instance features, and predict point-wise moving confidence scores. By inserting instance information, our approach achieves better moving object segmentation performance and can distinguish between actually moving instances (red points) and static (cyan points). The green bounding boxes represent cars and the blue bounding boxes represent pedestrians.

$\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{N-1}$, we aim to segment the moving objects in $\mathcal{S}_0$ by classifying each point as either moving or static. To achieve that, we utilize the spatio-temporal information of the consecutive scans. We first align all the scans to the current viewpoint. Assuming the relative pose transformations between past consecutive scans are provided by an odometry method, noted as $\boldsymbol{T}_1^0, \boldsymbol{T}_2^1, ..., \boldsymbol{T}_N^{N-1}$, we transform the past $N-1$ scans to the current scan by:

$$\mathcal{S}_{j\to 0} = \{\boldsymbol{p}_i' = \boldsymbol{T}_j^0 \boldsymbol{p}_i | \boldsymbol{p}_i \in \mathcal{S}_j\},\ \boldsymbol{T}_j^0 = \prod_{k=0}^{j-1} \boldsymbol{T}_{j-k}^{j-k-1}. \quad (1)$$

We add an additional time dimension for each point in the aligned point clouds to feed the temporal information into our network. As an example, for a point $\boldsymbol{p}_i'$ in scan $\mathcal{S}_{j\to 0}$, we assign the scan time interval t_i to the point and get $\tilde{\boldsymbol{p}}_i{}' = [x_i', y_i', z_i', t_i]^\mathsf{T}$. we ignore the homogeneous term for simplicity. For memory saving and system efficiency, we quantize the aligned point clouds to 4D voxels with a time resolution Δt and a space resolution Δs, and only use the non-empty parts as the network input.

B. Network Structure

The overview of our network is shown in Fig. 2. Our network is composed of three main components: MotionNet, instance detection module, and upsample fusion module. MotionNet is designed to extract motion features of the input 4D voxels. We then concatenate motion features with original point features and use an instance detection module to extract the spatio-temporal features from the current scan for instance detection and feature fusion. Finally, the upsample fusion module achieves point-wise MOS by integrating spatio-temporal and instance information.

1) MotionNet: MotionNet utilizes consecutive LiDAR scans to extract motion information. It takes the 4D voxels introduced in Sec. III-A as input and outputs point-wise motion features $\boldsymbol{F} = \{\boldsymbol{f}_i \in \mathbb{R}^3\}_{i=1}^M$ of the current scan. We design our MotionNet based on 4DMOS [11]. It utilizes a 4D sparse convolution mechanism, i.e., Minkowski engine [22] and an hourglass structure to capture the features from different layers of the network. Furthermore, the skip connection is used to fuse the features of multiple layers of the network to maintain more details. Unlike 4DMOS, our MotionNet has fewer layers since we use this backbone only to extract motion features instead of directly outputting point-wise moving segmentation labels, which lightens the feature extraction backbone and is the key to achieving online performance. Even though the backbone is lightweight, our method still performs well in motion detection due to our instance-aware design detailed in the following subsections.

2) Instance Detection Module: Detecting moving objects through instance information is inspired by human perception of the world. As humans, we typically do not directly judge whether a point is moving or not. Instead, we first determine whether an object is moving and then reason the points on it are also moving. This top-down reasoning can avoid a problem in existing methods, where part of the same object is classified as moving while the other is considered static. Based on the fact that all points on the same object should be labeled consistently, we propose an instance detection module to extract the instance features and help the following

moving objects reasoning. In this instance detection module, we concatenate the learned point-wise motion features $\boldsymbol{f}$ with point coordinates x, y, z, and intensity r as input, which will be used to generate spatio-temporal features. Afterwards, we use multiple layers of 3D sparse convolution [27] to extract spatio-temporal features from the current scan and gradually compress it into a BEV image. After simple 2D convolutions, CenterHead [28] backbone is then used to generate C heatmaps $\hat{\boldsymbol{E}}$ of size $H \times W$ and instance information $\hat{\boldsymbol{O}} = \{\hat{\boldsymbol{o}}_u = [\tilde{x}_u, \tilde{y}_u, z_u, l_u, m_u, q_u, \sin(\omega_u), \cos(\omega_u)] \in \mathbb{R}^8\}_{u=1}^{U}$, where C represents the number of instance categories, U represents the number of instances, $\tilde{x}$ and $\tilde{y}$ represent coordinate quantization offsets, z represents the height, l, m and q represent 3D bounding box size and ω represents orientation. Each score in heatmap $\hat{E}_{cij}$ represents the confidence at location (i, j) belonging to the instance class c. For more details about CenterHead, please refer to [28].

3) Upsample Fusion Module: To introduce instance information into point-level moving segmentation, we find and extract points within the instance bounding box as instance features. Subsequently, we directly concatenate the instance features with the spatio-temporal features as input and utilize convolutional layers to fuse them. During the upsample fusion process, we utilize the same number of layers of 3D sparse deconvolutions [27] as those convolutions used in the instance detection module to resume the point-level features hierarchically. To strengthen the instance information in the final features and maintain more details learned in different layers, we propagate the instance features from multi-resolution point clouds and form the instance pyramid. After concatenating with the corresponding spatio-temporal features, we inject them into the upsample fusion module.

Finally, the upsample fusion module outputs the point-wise features $\boldsymbol{F}' = \{\boldsymbol{f}'_i \in \mathbb{R}^3\}_{i=1}^{M}$, which are then used to determine the moving label of LiDAR points after passing a softmax function. Specifically, three channels of $\boldsymbol{f}'_i$ represent the confidence of a point belonging to three categories, respectively, which are unlabeled, static, and moving. The category with the highest score is the final label of the point.

C. Moving Instance Refinement

The network can determine the moving instances by directly checking the predicted MOS label of the instance center point. However, such pure bottom-up fashion may misdetect, thus enlarging the amount of misclassified points. To further improve the performance, we then check again the point-wise predictions within an instance and propose the instance-based refinement algorithm in a combined bottom-up and top-down fashion, as described in Alg. 1.

This bottom-up aims to refine the results of the following second cases. Firstly, if the percentage of the moving points in an instance is greater than α_0, the instance is considered to be moving (see lines 8-9). Secondly, if the number of moving vehicles in the scan exceeds β_1, the scene is considered to be highly dynamic, such as the highway, and the points of the vehicle can be labeled as moving with lower confidence β_0, which means they can be easily labeled as moving (see lines 10-17). Besides, to improve the detection capability of slowly moving instances, we track instances through LiDAR pose alignment and bounding box dimension matching. If an instance has been classified as moving for θ_1 times in past θ_0 scans, the instance is considered to be moving in the current scan as well (see lines 18-24). During the top-down step, when an instance is identified as moving, all points within the instance are determined as moving, which is natural when considering each instance as a rigid body.

Note that the refinement algorithm is only performed on the points with instance labels. By this, we exploit the instance information in both top-down and bottom-up ways to improve the MOS performance.

Algorithm 1 Instance-based refinement algorithm

Input: Moving labels $\boldsymbol{L}$ and confidence scores $\boldsymbol{P}$, and instance information $\boldsymbol{B}$ of network prediction, and moving label of instances $\boldsymbol{I}$, relative pose transformations $\boldsymbol{T}$, LiDAR scan $\mathcal{S}$, and other thresholds α_0, α_1, β_0, β_1, θ_0, θ_1.
Output: Moving label $\boldsymbol{L}_0{}'$ of each point.

1: $\boldsymbol{L}_0{}' = \boldsymbol{L}_0$
2: $\boldsymbol{I}_0 = \text{Zeros}(\text{Length}(\boldsymbol{B}_0))$
3: $idx = \text{Find\_index}(\mathcal{S}_0, \boldsymbol{B}_0)$ %The index of points in instance.
4: $m_c = 0$ %The number of moving vehicles.
5: $\boldsymbol{Q}$=Zeros(Length($\boldsymbol{B}_0$)) % Instances' status in past scans.
6: $\boldsymbol{M}$=Zeros(Length($\boldsymbol{B}_0$)) % Instances in high dynamic scene.
7: **for** each instance b^i in $\boldsymbol{B}_0$ **do**
8: **if** $\frac{\text{Length}(\boldsymbol{L}_0[idx[b^i]]==1)}{\text{Length}[idx[b^i]]} > \alpha_0$ **then**
9: $\boldsymbol{I}_0[b^i] = 1$
10: **if** Instance_label(b^i)==car **then**
11: **if** $\frac{\text{Length}(\boldsymbol{L}_0[idx[b^i]]==1)}{\text{Length}[idx[b^i]]} > \alpha_1$ **then**
12: $m_c = m_c + 1$
13: **if** $\frac{\text{Length}(\boldsymbol{P}_0[idx[b^i]]>\beta_0)}{\text{Length}[idx[b^i]]} > 0.5$ **then**
14: $\boldsymbol{M}[b^i] = 1$
15: **for** each instance b^i in $\boldsymbol{B}_0$ **do** % For high dynamic scene
16: **if** $m_c > \beta_1$ and $\boldsymbol{M}[b^i] == 1$ **then**
17: $\boldsymbol{I}_0[b^i] = 1$;
18: $\boldsymbol{C}_B^0$, $\boldsymbol{O}_B^0 = \text{Get\_box}(\boldsymbol{B}_0)$ %Get center and box of instances.
19: **for** $j = 1, ..., \theta_0$ **do** %Integrating past observations.
20: $\boldsymbol{C}_B^{0\to j} = \text{Transform}(\boldsymbol{C}_B^0, \boldsymbol{T}_0^j)$
21: $\boldsymbol{C}_B^j$, $\boldsymbol{O}_B^j = \text{Get\_box}(\boldsymbol{B}_j)$
22: **if** $\text{Match}(\boldsymbol{C}_B^{0\to j}, \boldsymbol{C}_B^j, \boldsymbol{O}_B^0, \boldsymbol{O}_B^j)$ and $\boldsymbol{I}_j[\boldsymbol{B}_j] == 1$ **then**
23: $\boldsymbol{Q}[\boldsymbol{B}^0] = \boldsymbol{Q}[\boldsymbol{B}^0] + 1$
24: $\boldsymbol{I}_0[\boldsymbol{Q}[\boldsymbol{B}^0] > \theta_1] = 1$
25: **for** each instance b^i in $\boldsymbol{B}_0$ **do** % Output point-wise labels.
26: **if** $\boldsymbol{I}_0[b^i] == 1$ **then**
27: $\boldsymbol{L}_0{}'[idx[b^i]] = 1$

D. Loss and Network Training

We train our network once to achieve both instance detection and MOS. Our loss function is composed of motion loss L_{motion}, instance classification loss L_{cls}, bounding box regression loss L_{reg}, and moving segmentation loss L_{mos}:

$$L_{\text{total}} = L_{\text{motion}} + L_{\text{cls}} + L_{\text{reg}} + L_{\text{mos}}. \tag{2}$$

L_{motion} and L_{mos} use the CrossEntropy Loss function [10], [11] defined as follow:

$$L_{\text{motion}}(y,\hat{y}) = -\sum_{i=1}^{M}\sum_{k=1}^{D} P_k(y_i)\log(P_k(\hat{y}_i)), \tag{3}$$

$$L_{\text{mos}}(y,\hat{y}') = -\sum_{i=1}^{M}\sum_{k=1}^{D} P_k(y_i)\log(P_k(\hat{y}'_i)), \tag{4}$$

where y_i is the ground truth, $\hat{y}_i$ is the prediction of MotionNet and $\hat{y}'_i$ is the prediction of upsample fusion module. $P_k(y)$ denotes probability that y belongs to the k-th class in [unlabeled, static, moving], and $P_k(\hat{y}_i) = \text{softmax}(\boldsymbol{f}_i)$, $P_k(\hat{y}'_i) = \text{softmax}(\boldsymbol{f}'_i)$.

We follow [29] to use a variant focal loss for L_{cls} and a smooth L1 loss for L_{reg}:

$$L_{\text{cls}} = \frac{-1}{U}\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases}(1-\hat{E}_{cij})^{\sigma}\log(\hat{E}_{cij}) & \text{if } g_{cij}=1\\ (1-g_{cij})^{\gamma}(\hat{E}_{cij})^{\sigma}\log(1-\hat{E}_{cij}) & \text{otherwise}\end{cases}, \tag{5}$$

$$L_{\text{reg}} = \frac{1}{U}\sum_{u=1}^{U}\text{smooth-L1-loss}(\boldsymbol{o}_u, \hat{\boldsymbol{o}}_u), \tag{6}$$

where g_{cij} indicates the Gaussian bumps of the heatmap encoded through the ground truth, both σ and γ are the hyper parameters ($\sigma = 2$, $\gamma = 4$ in our experiments), and $\boldsymbol{o}_u$ represents the ground truth. We refer more details for L_{cls} and L_{reg} in [28], [29].

IV. Experimental Evaluation

In this section, we conduct experiments to support our proposed three key claims and examine the effects of different modules within the network architecture.

A. Experimental Setup

Datasets. We train our model on both SemanticKITTI-MOS dataset [9], [30] and KITTI-road dataset[1] and evaluate it on the SemanticKITTI-MOS benchmark[2]. SemanticKITTI-MOS dataset contains a total of 22 sequences with a point-wise semantic category of moving or static. We use sequences 00-07 and 09-10 for training, sequence 08 for validation, and sequences 11-21 for testing. To overcome the imbalance distribution of quantities between moving and static objects on the SemanticKITTI-MOS dataset, we follow [10] and use their conducted training and validation sets based on KITTI-road, where sequences 30-34, 40 for training and 35-39, 41 for validation.

In addition to the ground truth of moving objects, our method also requires instance information for training. We generate 3D bounding boxes for both the SemanticKITTI dataset and KITTI-road dataset by first enclosing the clusters generated by the Euclidean clustering algorithm in PCL library [31], and then manually modifying erroneous cases. We will release the instance data together with our code for the convenience of the community. We furthermore show our instance detection results on the KITTI-tracking dataset[3].

[1]https://www.cvlibs.net/datasets/kitti/raw_data.php?type=road

[2]https://codalab.lisn.upsaclay.fr/competitions/7088

[3]https://www.cvlibs.net/datasets/kitti/eval_tracking_overview.php

TABLE I: Performance comparison with other baseline methods on SemanticKITTI-MOS benchmark. Best result in bold.

	IoU[%]
SpSequenceNet	43.2
KPConv	60.9
AutoMOS*	62.3
LMNet†	62.5
4DMOS	65.2
MotionSeg3D*	70.2
RVMOS†	74.7
Ours* ($N = 10$ Scans, $\Delta t = 0.1$ s)	**75.6**

* indicates the method is trained both on the SemanticKITTI dataset and KITTI-road dataset.

† indicates the method exploiting semantic labels.

Evaluation Metrics. We use intersection-over-union (IoU) [32] of the moving objects as the evaluation metric:

$$\text{IoU} = \frac{\text{TP}}{\text{TP}+\text{FP}+\text{FN}}, \tag{7}$$

where TP, FP, and FN represent true positive, false positive and false negative predictions of moving points.

Training Details. We set the number of consecutive input scans $N = 10$. Before being taken as input, the point clouds are quantized into 4D voxels with a time resolution $\Delta t = 0.1\,\text{s}$ and spatial resolution $\Delta s = 0.1 \times 0.1 \times 0.1\,\text{m}$. In the refinement, we set $\alpha_0 = 0.6$, $\alpha_1 = 0.3$, $\beta_0 = 10^{-5}$, $\beta_1 = 5$, $\theta_0 = 5$ and $\theta_1 = 3$.

Our framework is implemented in PyTorch [33] and trained on 4 NVIDIA RTX 3090 GPUs. We train the whole model using the Adam optimizer [34] with a batch size of 16. The learning rate is initialized as 10^{-4} and decays exponentially by 0.01 each epoch. We train the network for total 160 epochs for SemanticKITTI and KITTI-road, and 80 epochs for only SemanticKITTI dataset. The widely-used data-augmentation strategies: random flipping, scaling, and rotations are adopted to boost the performance.

B. MOS Performance Evaluation

We evaluate the performance of our method in SemanticKITTI-MOS benchmark and compare it with several baseline methods, including SpSequenceNet [35], KPConv [36], AutoMos [23], LMNet [9], 4DMOS [11], MotionSeg3D [10] and RVMOS [1]. SpSequenceNet and KPConv are originally used for semantic segmentation, while we present their MOS results reported in [23]. In addition, we obtain the experimental results of all other methods from the SemanticKITTI-MOS benchmark, which are also made public in their paper.

The quantitative comparison is shown in Tab. I, and the results support our second claim that our approach achieves the state-of-the-art MOS performance with 75.6% IoU. LMNet, 4DMOS and RVMOS are trained on the SemantiKITTI dataset, and RVMOS achieves great performance improvement up to 74.7% IoU. That is mainly due to the fact

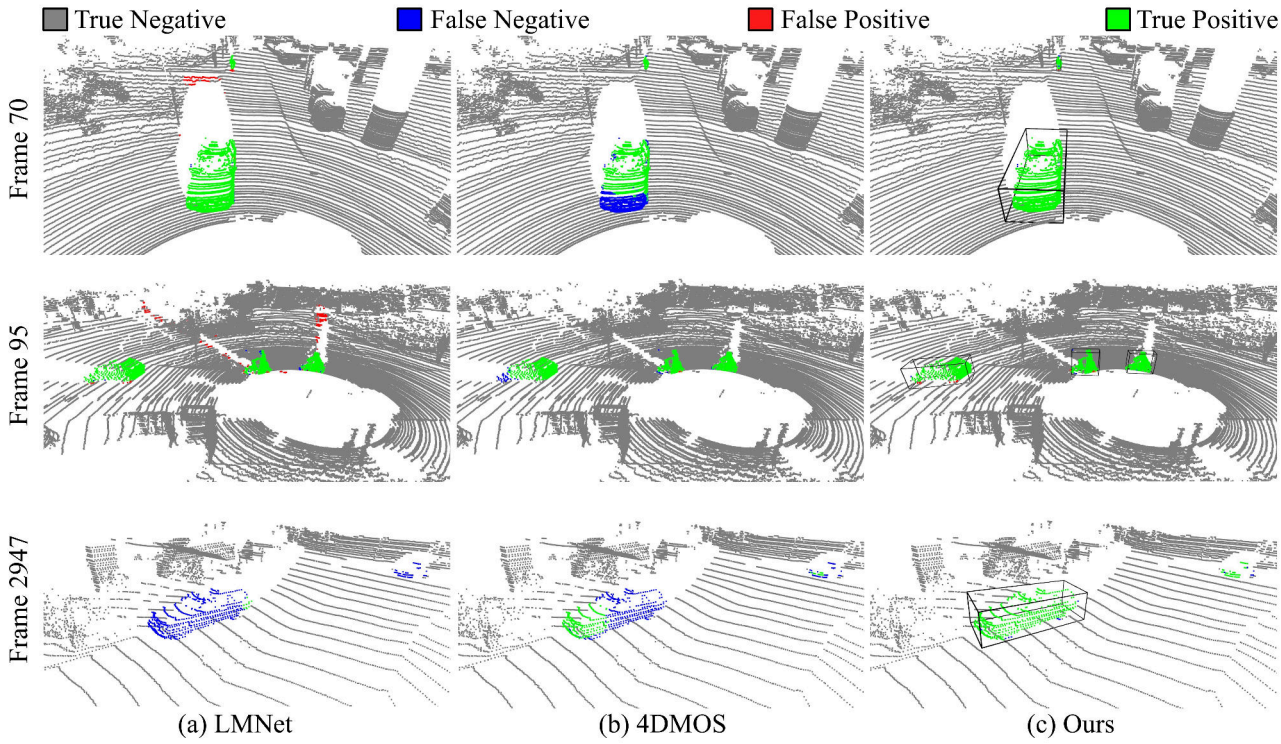


Fig. 3: Qualitative results of MOS compared with LMNet and 4DMOS on SemanticKITTI validation set. Best viewed in color.

that RVMOS inserts semantic information in the network implementations. AutoMos, MotionSeg3D and our network perform training on the SemantiKITTI and an additional labeled KITTI road dataset [9], [10] in order to reduce the impact of unbalanced data distribution.

For a fair comparison, we also report the validation set result of our network trained only on the SemanticKITTI dataset and compare it with these baselines. The 4DMOS result is re-evaluated using the same pose as ours, which is estimated by the SuMa [37] algorithm with loop closure. As shown in Tab. II, our approach still achieves the best performance.

Fig. 3 visualizes qualitative comparisons of our method with LMNet and 4DMOS on the SemanticKITTI validation set. LMNet is the first work on LiDAR MOS exploiting range images, which is fast but results in many wrong predictions. 4DMOS performs well on fast-moving object segmentation, but not as well on slow-moving object segmentation. As 4DMOS cannot capture the instance information of the moving points, only partial points of the moving instance can be correctly predicted. However, our approach can segment moving objects completely and has the ability to detect slow-moving instances by integrating past observations in the instance-based refinement algorithm. The qualitative experimental results support our first claim and further demonstrate that instance information is highly valuable for MOS task.

C. Generalization Analyses

To demonstrate the third claim that our approach generalizes well to different scenes, we conduct experiments on the Apollo [38] dataset. We compare our method with three open source baseline methods, including MotionSeg3D [10], LMNet [9] and 4DMOS [11]. All methods are only trained on the training set of SemanticKITTI and evaluated on the Apollo dataset without modifying any settings or parameters fine-tuning. Besides, we also present the result of fine-tuned LMNet combined with AutoMOS [23], noted as LMNet+AutoMOS+Fine-Tuned [9], see [23] for details. The results are shown in Tab. III. Two range image-based methods MotionSeg3D and LMNet show bad performance in the generalization test, while 4DMOS and our method maintain good segmentation ability in unknown environments. The possible reason is that the projection-based approaches

TABLE II: Performance comparison with other baseline methods on SemanticKITTI validation set, and all methods are trained using the same dataset. Best result in bold.

	IoU[%]
LiMoSeg	52.6
LMNet	67.1
MotionSeg3D	68.1
RVMOS	71.2
4DMOS	71.9
Ours ($N = 10$ Scans, $\Delta t = 0.1$ s)	**73.2**

TABLE III: The generalization performance evaluation on the Apollo dataset. Best result in bold.

	IoU[%]
MotionSeg3D	7.5
LMNet	16.9
LMNet+AutoMOS+Fine-Tuned	65.9
4DMOS	73.1
Ours ($N = 10$ Scans, $\Delta t = 0.1$ s)	**78.0**

TABLE IV: Ablation study of different components and setups in SemanticKITTI validation set. N and Δt refer to the number and the temporal resolution of input sequence LiDAR scans, mentioned in Sec. III-A. The instance layer is the component of the Instance Pyramid. Refinement refers to our proposed instance-based refinement algorithm in Sec. III-C.

	Method	Train		Inference		
		N	Δt	N	Δt	IoU[%]
[A]	MotionNet	5	0.1	5	0.1	55.6
[B]	+ Instance Detection + Upsample Fusion(without instance)	5	0.1	5	0.1	59.8
[C]	+ Instance Detection + Upsample Fusion(with 1 instance layer)	5	0.1	5	0.1	60.4
[D]	+ Instance Detection + Upsample Fusion(with 4 instance layers)	5	0.1	5	0.1	60.8
[E]	+ Instance Detection + Upsample Fusion(with 4 instance layers)	5	0.2	5	0.2	63.2
[F]	+ Instance Detection + Upsample Fusion(with 4 instance layers)	10	0.1	5	0.1	65.2
[G]	+ Instance Detection + Upsample Fusion(with 4 instance layers)	10	0.1	10	0.1	70.7
[H]	+ Instance Detection + Upsample Fusion(with 4 instance layers) + Refinement	10	0.1	10	0.1	73.2

overfit the setup of the sensor and specific patterns in the trained environments, which will affect the performance of projection-based approaches while the point cloud-based approaches will not be affected. Benefiting from the learned instance information, our method still outperforms 4DMOS.

D. Ablation Study

We conduct ablation studies on the SemanticKITTI validation set to better understand the effectiveness of individual modules in our method. The results shown in Tab. IV.

From the comparison of [A] and [B], we show that the performance is boosted by introducing the spatio-temporal information extracted by our instance detection module and upsample fusion module. By integrating the instance information, the results are further improved, as can be seen in the comparison of [B] and [C], [D]. The advantage of introducing the instance information can be further demonstrated by comparing [G] and [H].

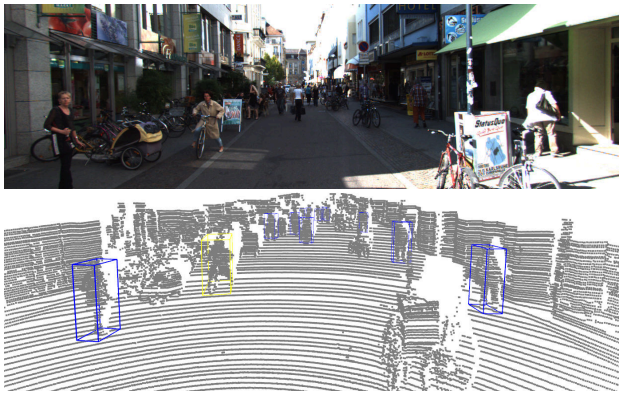
The comparison of [D], [F] and [G] shows that feeding more information to the network by increasing the number of input scans N both in training (see [D] and [F]) or the inference (see [F] and [G]) improves the performance. Raising temporal resolution Δt can also improve the performance, as can be seen in the comparison of [D] and [E]. The reason is increasing the temporal resolution will extend the time horizon of the input sequence, which can help the MotionNet capture the motion information.

E. 3D Instance Detection

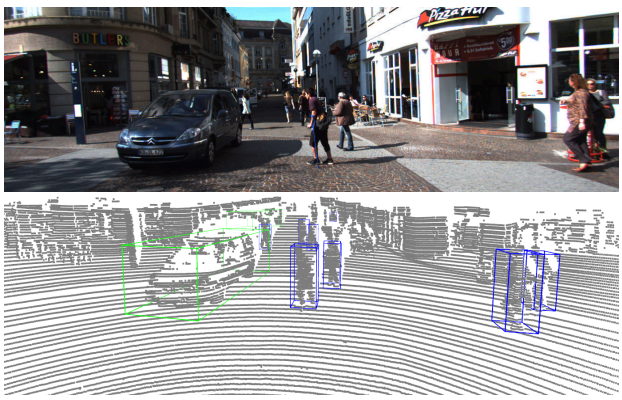
To further demonstrate the ability of our method to detect instances, we evaluate the performance on the KITTI tracking dataset with the model pre-trained on SemanticKITTI. The KITTI tracking dataset contains more instances compared to the SemanticKITTI dataset. The results presented in Fig. 4 show that our method is able to accurately predict the instance category and 3D bounding box of major traffic participants, albeit in a complex environment.

F. Runtime

We measure the runtime of our unoptimized Python implementation method with a single NVIDIA RTX 3090 GPU. Our network takes an average time of 82.4 ms for $N = 5$ and 120 ms for $N = 10$. And refinement takes 7 ms. According to the ablation studies in Sec. IV-D, one can choose a suitable number of input scans N to achieve faster inference with a little loss of accuracy.



(a) Frame 158



(b) Frame 674

Fig. 4: The instance detection results of main traffic participants in sequence 19 of the KITTI tracking dataset. The blue bounding boxes represent pedestrians, the yellow bounding boxes represent cyclists, and the green bounding boxes represent cars.

V. Conclusion

In this paper, we presented a novel network to predict both point-wise moving labels and instance information in the current scan. Our network extracts motion features, spatio-temporal features and instance information from different modules and fuses them to achieve moving object segmentation. By again integrating instance information in the instance-based refinement algorithm, our approach can distinguish between moving and static instances and fully

segment the points within moving instances. The experimental results on the SemanticKITTI and the Apollo dataset demonstrate that our approach is effective and generalizes well to different environments.

REFERENCES

[1] J. Kim, J. Woo, and S. Im. Rvmos: Range-view moving object segmentation leveraged by semantic and motion features. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):8044–8051, 2022.

[2] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 12–21, 2019.

[3] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[4] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss. Learning an Overlap-based Observation Model for 3D LiDAR Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[5] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Mapping the Static Parts of Dynamic Scenes from 3D LiDAR Point Clouds Exploiting Ground Segmentation. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021.

[6] G. Kim and A. Kim. Remove, then revert: Static point cloud map construction using multiresolution range images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[7] P. Chen, J. Pei, W. Lu, and M. Li. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing*, 497:64–75, 2022.

[8] C. Lee and K. Song. Path re-planning design of a cobot in a dynamic environment based on current obstacle configuration. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[9] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 6:6529–6536, 2021.

[10] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen. Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 11456–11463. IEEE, 2022.

[11] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss. Receding moving object segmentation in 3d lidar data using sparse 4d convolutions. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7503–7510, 2022.

[12] T. Kreutz, M. Mühlhäuser, and A. S. Guinea. Unsupervised 4d lidar moving object segmentation in stationary settings with multivariate occupancy time series. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pages 1644–1653, 2023.

[13] B. Mersch, T. Guadagnino, X. Chen, I. Vizzo, J. Behley, and C. Stachniss. Building Volumetric Beliefs for Dynamic Environments Exploiting Map-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):5180–5187, 2023.

[14] S. A. Baur, D. J. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger. SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[15] X. Liu, C. R. Qi, and L. J. Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] G. Dong, Y. Zhang, H. Li, X. Sun, and Z. Xiong. Exploiting rigidity constraints for lidar scene flow estimation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12776–12785, 2022.

[17] F. Pomerleau, P. Krüsiand, F. Colas, P. Furgale, and R. Siegwart. Long-term 3d map maintenance in dynamic environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[18] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla. Robust Method for Removing Dynamic Objects from Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[19] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss. Static map generation from 3d lidar point clouds exploiting ground segmentation. *Robotics and Autonomous Systems*, 159:104287, 2023.

[20] H. Lim, L. Nunes, B. Mersch, X. Chen, J. Behley, H. Myung, and C. Stachniss. ERASOR2: Instance-Aware Robust 3D Mapping of the Static World in Dynamic Scenes. In *Proc. of Robotics: Science and Systems (RSS)*, 2023.

[21] P. Ghorai, A. Eskandarian, Y. K. Kim, and G. Mehr. State estimation and motion prediction of vehicles and vulnerable road users for cooperative autonomous driving: A survey. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 23(10):16983–17002, 2022.

[22] C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzo, J. Behley, and C. Stachniss. Automatic labeling to generate training data for online lidar-based moving object segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6107–6114, 2022.

[24] S. Mohapatra, M. Hodaei, S. Yogamani, S. Milz, H. Gotzig, M. Simon, H. Rashed, and P. Maeder. Limoseg: Real-time bird's eye view based lidar motion segmentation. *arXiv preprint*, 2021.

[25] P. Wu, S. Chen, and D. N. Metaxas. MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] H. Lim, S. Hwang, and H. Myung. ERASOR: Egocentric Ratio of Pseudo Occupancy-Based Dynamic Object Removal for Static 3D Point Cloud Map Building. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2272–2279, 2021.

[27] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] T. Yin, X. Zhou, and P. Krahenbuhl. Center-Based 3D Object Detection and Tracking. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[29] H. Law and J. Deng. CornerNet: Detecting Objects as Paired Keypoints. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2018.

[30] 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 Intl. Conf. on Computer Vision (ICCV)*, 2019.

[31] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1–4. IEEE, 2011.

[32] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, Natalia N. Gimelshein, L. Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 8026–8037, 2019.

[34] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proc. of the Int. Conf. on Learning Representations (ICLR)*, abs/1412.6980, 2015.

[35] H. Shi, G. Lin, H. Wang, T. Y. Hung, and Z. Wang. SpSequenceNet: Semantic Segmentation Network on 4D Point Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[36] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[37] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[38] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.