EmPointMovSeg : Sparse Tensor Based Moving Object Segmentation in 3D LiDAR Point Clouds for Autonomous Driving Embedded System

Zhijian He, Xueli Fan, Yun Peng, Zhaoyan Shen, Jianhao Jiao and Ming Liu*

***Abstract*—Object segmentation is a per-pixel label prediction task which targets at providing context analysis for autonomous driving. Moving object segmentation (MOS) serves as a sub-branch of object segmentation, targeting at separating the surrounding objects into binary options: dynamic and static. MOS is vital for safety-critical task in autonomous driving because dynamic objects are often true potential threat to self-driving car comparing to static ones. Current methods typically address the MOS problem as a category feature to label mapping task, which is not rational in reality. For example, a parking car should be considered as static instead of moving object category. There is little systematic theory to differentiate object moving characteristics from non-moving characteristics in MOS. Furthermore, restricted by limit resource in embedded system, MOS is often in an off-line manner due to huge computational requirement. An on-line and low computational cost MOS is an urgent demand for practical safety-critical mission which takes immediate reaction as compulsory.**

In this paper, we propose EmPointMovseg, an efficient and practical 3D LiDAR MOS solution for autonomous driving. Leveraging the power of well-adapted autoregressive system identification (AR-SI) theory, EmPointMovseg theoretically explains moving-object feature in large scale 3D LiDAR semantic segmentation. An end-to-end sparse tensor based CNN which balances segmentation accuracy and on-line process ability is proposed. We construct our experiment on both representative dataset benchmarks and practical embedded systems. The evaluation result shows the effectiveness and accuracy of our proposed solution, conquering the bottleneck in the on-line large-scale 3D LiDAR semantic segmentation.

***Index Terms*—Embedded system, LiDAR moving-object segmentation, Deep learning.**

I. Introduction

OJECT segmentation serves as an important role in autonomous driving safety. Via predicting per-pixel label in driving scene, object segmentation algorithm is able to provide comprehensive context information to the system, which benefits the perception result so that path planning can be more accurate. For example, by detecting lane, traffic sign, .etc, autonomous driving system is able to get an excellent description of the environment so that smart the planning decision is obtained. However, for safety-critical mission in autonomous driving, the potential threat mainly lies on pedestrian, cars or trucks which have moving characteristics in common. Less danger is caused by static objects such as fence, trees, traffic signs, etc. This phenomenon generates a sub-research topic in object segmentation so called moving-object segmentation. Although many solutions exist in multi-category object segmentation, MOS domain which requires heavy computational expense and the knowledge of distinguishing dynamic objects from static ones [1], [2], [3], [4], is still a bottleneck in autonomous driving safety object segmentation.

The majority of current segmentation methods [5] [6] [7] address this task as a supervised-learning per-pixel classification problem. Typically, an U-Net [8] like CNN takes in category specific feature and minimize the network loss between prediction output and annotated label. For example, cluster of 3D LiDAR data containing trunks and branches feature is considered as tree. After finishing the capture process, these methods adopt an Encode-Decode pattern to train a context understanding network, resulting in high accuracy multiple class object segmentation. Unlike the majority of current existing segmentation methods which inputs raw 3D LiDAR data and outputs multiple label accordingly, MOS emphasizes a binary segmentation task that figures out moving objects from non-moving objects in autonomous driving scene. MOS considers the moving objects as the root cause of many traffic accidents so that it can not be constructed as a simple binary segmentation problem. For example, in Fig. 1, the car (yellow box) parking at the roadside, opposite to the car (red box) driving on the road, although belonging to moving-object category, should not be considered as moving semantically. Only the red mask in right sub-picture linking to red box in the left sub-picture is a correct moving-object segmentation.

As a result, an extra feature which contains the information of differentiating moving objects from static ones needs to be added into deep-learning training. The essential of this feature can be regarded as changing position in sequential data. As a result, finding temporal geometric change is a vital solution to the MOS. However, the varying density in large scale LiDAR data formulates the difficulty in exploring temporal geometric

This work was supported by Zhongshan Municipal Science and Technology Bureau Fund, under project ZSST21EG06, Collaborative Research Fund by Research Grants Council Hong Kong, under Project No. C4063-18G, and Department of Science and Technology of Guangdong Province Fund, under Project No. GDST20EG54, awarded to Prof. Ming Liu.

∗ Corresponding author

Zhijian He, Yun Peng, Jianhao Jiao and Ming Liu are with the RAM Lab of Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong SAR, CHINA. e-mail: (eehezhijian@ust.hk; pyun@connect.ust.hk; jjiao@connect.ust.hk; eelium@ust.hk).

Xueli Fan, is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, CHINA. e-mail: (csxfan@comp.polyu.edu.hk).

Zhaoyan Shen is the School of Computer Science and Technology, Shandong University, Qingdao 266200, China. e-mail: (shenzhaoyan@sdu.edu.cn).

Digital Object Identifier 10.1109/TCAD.2020.3020495

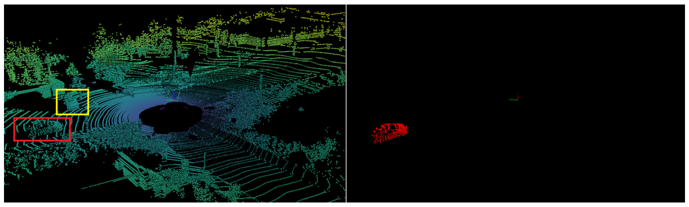

Fig. 1: Drawbacks in traditional classification-like 3D LiDAR semantic segmentation.

relationship in reality. This feature of point cloud limits on-line LiDAR segmentation application for the reason that both the temporal [1] and the spacial information [5] have to be considered at the same time, which generates two NP-hard problem: 1) Calculating too much feature in runtime requires expensive computational resource at the same time, especially in case that on-line moving segmentation is compulsory for safety-critical mission; 2) The temporal feature should be theoretical explained so that the result of differentiating binary mask in MOS can be guaranteed.

For the first problem, current methods prepare preloading relative pose information and raw point cloud in advance. Via powerful GPU cloud computing or excellent graphic hardware, 3D LiDAR data can be manipulated to achieve accurate segmentation results [6], [9], [10], which is not the real environment settings in practical autonomous driving embedded system. Although polar shape [5] or other rational spacial structure [10] in expressing point cloud is explored, current methods emphasize the effect of raising intersection over union (IoU) instead of considering both the efficiency and accuracy in segmentation. Due to the large batch size during the inference phase, accurate IoU can be achieved only in off-line manner, which is useless given no reaction possibility is considered in reality. In the safety-critical compulsory scenario, the true positive dynamic characteristics of an object is more important than high category accuracy. Hence, efficient on-line accurate MOS is the goal in guaranteeing autonomous driving safety. In this paper, in virtue of sparse tensor and sparse convolution, we overcome the varying density feature via avoiding the empty space of LiDAR data to operate in convolution so that segmentation inference time reduces. Furthermore, understanding the shape of moving object is enough for practical semantic segmentation comparing to describe the point cloud densely. These two novel operations shrink the inference time to make on-line MOS possible in embedded system.

For the second problem, the widely-used Encode-Decode architecture CNN [8] is used as an efficient and accurate solution network in semantic segmentation. Well-picked temporal feature can construct powerful input to the network to improve moving-object segmentation significantly [1]. However, current methods [1], [11], [12], [13], [14] just estimate relative object states intuitively to distinguish dynamic object from static ones. In this paper, we utilize the well-adapted AR-SI theory [15] to propose a systematic approach to discover the true dynamic characteristics of an object. Via AR-SI oracle judgement, a dynamic object in autonomous driving scene can be confirmed. This so-called AR-SI filter constructs a temporal feature serving as the following CNN input. Combining spacial feature in raw point cloud and this generated temporal feature, we map the label for segmentation training to generate accurate IoU result. Targeting at binary mask of discovering moving object in real scene, we establish the AR-SI theory into feature selection not only to describe the moving character of dynamic objects, but also block the perturbation from static objects which is totally a blind area in traditional methods [16], [17], [18] using geometric feature only.

In this paper, we construct the temporal and the spacial feature to input to an Encode-Decode architecture CNN, and use sparse network architecture to predict the moving binary mask (details of overall solution is illustrated in Section III.B). Specifically, the contributions of this work are listed as followings:

1) We theoretically explain the essential of the model that distinguishes static objects from dynamic ones in autonomous driving. Leveraging the power of 3D LiDAR residual depth in temporal representation, we propose a novel AR-SI based feature to improve the the Encode-Decode architecture based CNN prediction significantly. Combining both the filtered temporal feature and the geometric feature, the CNN prediction is able to detect true dynamic object ignoring the perturbation from same category.
2) We propose sparse tensor and sparse convolution to handle the unconstructed raw LiDAR point cloud, which shrinks the gap between segmentation algorithm in the powerful GPU only and its practical use in autonomous driving embedded system.
3) We evaluate our approach on widely-used large scale LiDAR dataset SemanticKITTI to prove the segmentation performance. Then, we deploy our method on a real autonomous driving embedded system to check its practical effect.

The rest of this paper is organized as follows. Section II gives the background and motivation of conquering traditional semantic segmentation drawbacks in autonomous driving embedded system. Section III presents the details of using AS-RI theory to distinguish moving and non-moving objects. Using this theoretical analysis result, we illustrate our network details in section IV. In Section V, we present experimental results. Finally, via comparing the related work in Section VI, Section VII provides the conclusion.

II. Background and Motivation

In this section, we first give a brief overview of LiDAR semantic segmentation. The basic strategy on current LiDAR semantic segmentation is then illustrated. Finally, we present a motivation example for the on-line 3D LiDAR moving-object segmentation on resource limited embedded system.

A. Large Scale 3D LiDAR point cloud segmentation Task

The autonomous driving core technologies can be mainly divided into two classes [19]: vision-based and LiDAR-based. LiDAR sensor measures the time gap between the reflected

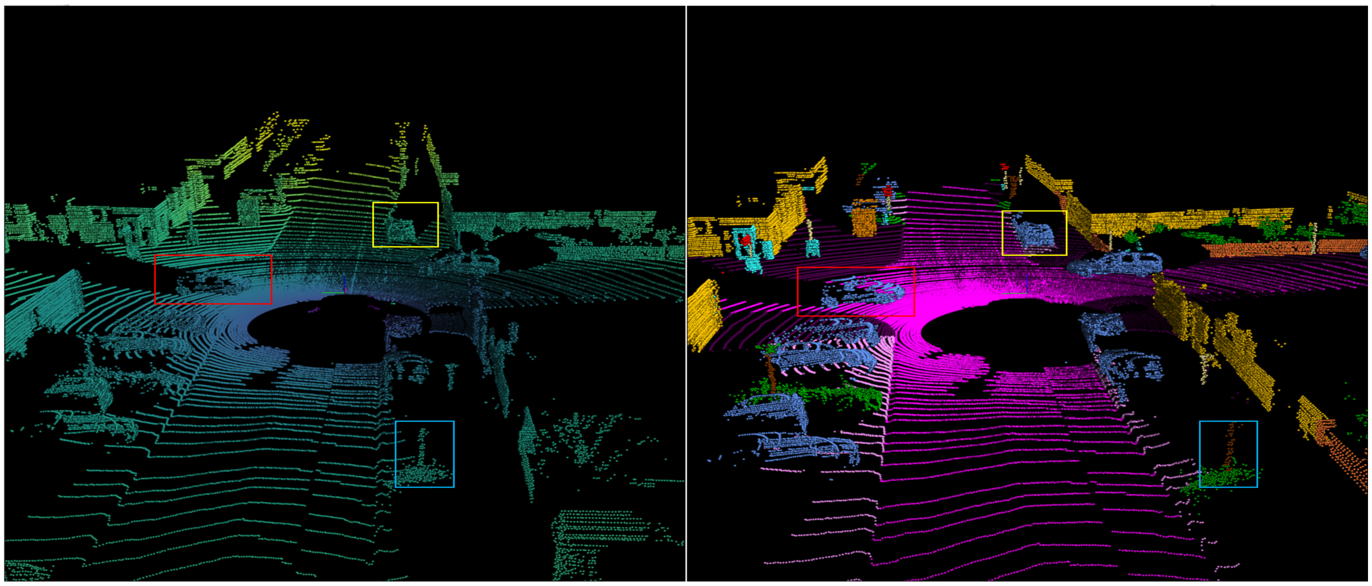

Fig. 2: 3D LiDAR semantic segmentation. The left picture is a raw LiDAR representation while the right one is indicated in an annotated manner

light sender and receiver, which can be used to construct 3D object points representation via multiplying LiDAR ware length. Denoted as $(x_i, y_i, z_i, intensity_i)$, raw LiDAR data contains reflected light position and its intensity, where i means sequence index. As Fig. 2 shows, the left image is the raw point cloud which displays spacial objects. However, this raw data scan can not satisfy perception requirements for the reason that no semantic or instance information is provided. For example, the car in the red box is meaningless to system unless pixel label is added. As the right image of Fig. 2 shows, object labeling made by annotation tool or handcraft provides scene understandable message, which can be used for either training or evaluation in segmentation task.

Current segmentation methods such as PointNet [16] input scan of dense 3D LiDAR data to a CNN to construct an end-to-end feature to label supervised training. The network model parameters are generated once the loss between the point prediction and the groundtruth is satisfied. The CNN plays a role of mapping geometric point cloud correlationship to annotated label. The inference is performed using new captured dense LiDAR data to produce prediction mask/label. Typically, this task contains tens of category labels for semantic segmentation, which is very similar to the deep-learning based classification problem [20].

B. Moving-object Segmentation

Comparing to all classes semantic segmentation, moving-object segmentation indicates a task of distinguishing static object from dynamic ones in large scale sequential 3D LiDAR perception. This process typically merges the original labels into three classes: moving, non-moving and unlabelled [1]. For example, car and bicyclist should be reorganized as moving label while fence and trunk are considered as non-moving. The blue bounding box in Fig. 2 indicates a tree labelled as non-moving while the red bounding box means a car labelled as moving.

The input to the evaluated method is a list of coordinates of three-dimensional points along with their remission. Each method should then output a label for each point of a scan. To assess the network performance, IoU over moving and non-moving parts of the environment are investigated [21]. Higher IoU means the performance of the network are better.

C. Corner Case in MOS

The current majority of methods in pure LiDAR data segmentation use feature to label mapping principal to train a CNN [5], [16], [22]. However, as Fig. 2 shows, the parking car in the yellow bounding box is a static object while the red bounding box car driving in the road stands for a real moving object. Although both of them are labelled as car in groundtruth, in moving-object segmentation, we should not treat it using category feature only. Misleading by wrong prediction label causes confusion in driving safety reaction because the parking car is actually harmless. More importantly, if incorrect static label given by the system, road accident may be triggered because the system treat potential danger as static safe condition.

D. Motivation

Static objects are relatively safe comparing to moving ones, for example, fence is not able to cause damage unless all the sensors on the car are disabled. In other words, moving object judgement plays a vital role in autonomous driving safety. This importance combining unsatisfied performance in existing moving-object segmentation approaches, motivates us to design our systematic solution with the following goals:

1) **Efficient**: A systematic moving-object segmentation method should handle the LiDAR batch fast enough

0278-0070 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Hong Kong University of Science and Technology. Downloaded on May 11,2022 at 01:48:15 UTC from IEEE Xplore. Restrictions apply.

to achieve on-line computation. We achieve this goal by leveraging the power of sparse tensor and sparse convolution.

2) **Accurate**: After considering the solution efficiency, segmentation design should be accurate not only to distinguish static and dynamic objects that belong to different classes, but also should identify true movable objects belonging to the same moving class. In this work, we use AR-SI theory to infer temporal feature which is added into traditional spatial feature for the CNN input.
3) **Practical**: The whole methodology should be able to deployed in an embedded system to prove its practical use. The system should be deployed in a real autonomous driving embedded system to meet the requirements of accurate moving-object segmentation while incurring low computational cost. On-line performance is critical and useful for guaranteeing autonomous driving safety.

III. Related Work

A. Semantic Segmentation

Semantic segmentation is a task that assigns each pixel a category label in an image or point cloud. Semantic segmentation appears in both vision and LiDAR domain, and gains much attention in recent years. Fully Convolutional Networks (FCN) [23], replaces fully connected layers in image classification task with convolution layers, which turns per pixel labelling possible in 2015. U-Net [8] creates the Encoder-Decoder architecture which constructs the cornerstone for vision semantic segmentation task. Badrinarayanan et al. [22] improves this architecture to develop it on both road scenes and SUN RGB-D indoor scene segmentation tasks [24]. Leveraged the power of Encoder-Decoder architecture, semantic segmentation in large scale dynamic 3D LiDAR point cloud has been successfully applied. Majority of the 3D LiDAR semantic segmentation target at mapping multiple categories label [5], [6] to each pixel on road scene so that the context of the open road can be understood. From semantic segmentation to panoptic segmentation [25], [26], polar coordinate [5] and other project coordinate [27] have been applied to organize the point cloud coordinate for calculating the according label in CNN. Dense point cloud approaches such as [6], [1], [28] have gain great achievement on multiple categories semantic segmentation. However, in the aspect of safety-critical mission, autonomous driving emphasizes the potential threat in true moving object other than typical non-moving object. The most important breakthrough is [1], which targets at distinguishing true dynamic object from false positive dynamic object using multiple categories semantic segmentation methods. For sparse tensor approach, PointMoSeg [29] combines temporal and spatial information in 3D LiDAR point cloud segmentation, which is mostly relative to our approach. However, it is not evaluated on a embedded platform to check its practical performance.

B. Moving-object Segmentation Task

Moving-object segmentation, which is sightly different from traditional multiple label mapping semantic segmentation [30], [27], aims to discover the real moving objects from non-moving ones [1]. This task can be divided into two kinds of approaches: Map-use and Map-free. Map-use approaches in 3D LiDAR semantic segmentation typically use two steps to manage moving-object segmentation. First, sensor data is captured off-line containing ego-motion and LiDAR data. A pre-built map is generated use time-consuming voxel or grid base methods in order to separate dynamic objects from static using real-time captured data in the second step [31], [32]. These kinds of approaches require a clean pre-built map and therefore can not be used in on-line semantic segmentation. Map-free approaches indicates on-line methods capture real-time LiDAR and ego-motion data to perform per pixel labelling task. Ruchti and Burgard et al. [33] use probability to predict movable objects. Dewan et al. [34] propose a method based on rigid body using LiDAR point cloud. It formulates the problem as an energy minimization problem that estimates motion vectors for rigid bodies. [35] proposes a deep convolutional neural network (DCNN) for semantic segmentation of a LiDAR scan. It proposes Bayes filter based method to make the predictions from the DCNN semantic state temporally consistent. These kinds of methods do not mention about their on-line segmentation performance, which is inevitable in practical safety-critical missions. Bogoslavskyi and Stachniss [36] present an effective method that segments the 3D data in a range image representation via removing the ground from the scan. These existing approaches do not theoretically explain the rational math model of moving-object segmentation. Instead, they make intuitive approaches in pose estimation or temporal 3D LiDAR scan sequence to solve moving-object segmentation. AR-SI theory [15] is well-established math in control-CPS, which can be serve as the oracle not only in moving/non-moving binary judgement, but also can be used in checking the moving track quality. This inspires us to systematically explain the binary feature in the input tensor of the CNN, which is novel in moving-object segmentation.

Context analysis task is often equal to high workload computational task. Projected image approaches consumes huge amount of computational time, although of high accuracy, is not practical in autonomous driving. Thanks to the power of sparse tensor/convolution, we can perform on-line approach in embedded system. Similar to [1], we train the network using binary masks to achieve an end-to-end fashion. However, we implement the CNN network using sparse tensor based manner while Chen et al. manage it in a dense tensor way.

IV. Solution Theory

Moving-object segmentation indicates an environment semantic context analysis in autonomous driving. Aiming to assist safety-critical reaction, this task addresses the problem of distinguishing dynamic object from static ones around the vehicle. In this section, we details the math theory in discovering moving object, which figures out the essential math model not similar to the majority of label-based deep-learning methods. Existing methods use sequential LiDAR scan as moving-object segmentation temporal feature but none provides a theoretical explanation.

A. Math Model in Distinguishing Moving from Non-moving Objects using AR-SI theory

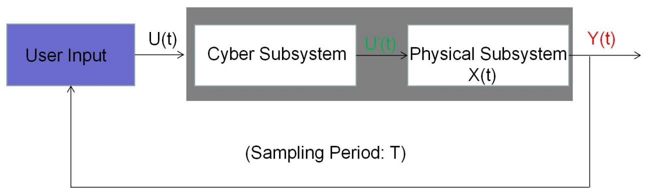

Fig. 3: A typical control-CPS architecture

We treat each object in the autonomous driving scene as a control-CPS system. For each control-CPS system, we regard static object as no control input while dynamic ones as the opposite [37]. For example, trees at the roadside can be considered as a typical control-CPS without user impulse, and cars can be treated as control-CPS with regular user pedal input. A classical control-CPS consists of user input module, cyber subsystem and physical subsystem as Fig.3 shows.

Generally, the user control input $U(t) \in R^q$ at time $t \in [0, +\infty)$ is generated in a regular period to pass onto the cyber subsystem [15] (For example, the car driver operates the pedal to the wheel at a preset period. If no user signal is captured, 0 is taken as the input of the cyber subsystem). The cyber subsystem inquires the positioning result from navigation subsystem and then calculate the error $U^{'}(t)$ between reference point and current position. $U^{'}(t)$ formulates the final impulse to the physical subsystem (also called ”physical plant”). The physical subsystem combines current state $X(t)$ and the impulse to produce the trajectory $Y(t)$ of the whole system. The user adjusts its input according to the physical plant trajectory during a sampling period T.

Specifically, if we regard the the gray area of Fig.3 as a black-box, whose input is $U_i \in R^q$ and output is $Y_i \in R^m$, then AR-SI typically models the relationship between U_i and Y_i $(\forall i = p, p+1, ...)$ as [15]:

$$Y_i = (\sum_{j=1}^{p} A_j Y_{i-j}) + BU_i + \xi_i, \tag{1}$$

, where Y_i stands for the physical subsystem position at time step i. p means the time steps tracking back Y_i, and the A_j is the corresponding weight. U_i is the user input at at time step i, and B is the weight parameter of user input. ξ_i indicates the SI error item. At time step i, last time step $i-1$ consecutive plant states $(Y_{i-p-1}, ..., Y_{i-2}, Y_{i-1})$ become available. The according parameters $A_1, A_2, ..., A_p$ and B of time step $i-1$ in Exp. 1 are optimized using AR-SI theory until the runtime accumulated SI error energy is minimized. AR-SI model uses these amount of optimized parameters and available plant states $(Y_{i-p}, ..., Y_{i-1})$ to predict Y_i with $\hat{Y}_i$ [15]:

$$\hat{Y}_i = (\sum_{j=1}^{p} A_j^{(*,i)} Y_{i-j}) + B^{(*,i)} U_i \tag{2}$$

When Y_i is available, the AR-SI prediction error becomes:

$$e_i = \hat{Y}_i - Y_i \tag{3}$$

A static object in autonomous driving scene can be considered as no user input, and its consecutive plant states are the same. This model can be calculated as:

$$Y_i = (\sum_{j=1}^{p} A_j Y_{i-j}) + \xi_i, \tag{4}$$

, where A_j equals to $1/p$. However, there exists certain sensor noise which forces the sum of the averaged Y_{i-j} can not perfectly be equal to Y_i. As a result, Exp. 3 is optimized as:

$$e_i = \left\| \hat{Y}_i - Y_i \right\| < \epsilon. \tag{5}$$

The threshold value ϵ can be inferred through Monte-Carlo [38] sampling. If the object trajectory approximation error using AS-RI model is lower than ϵ, we are able to predict it as non-moving and vice versa.

B. Proposed Solution

Using the math model in last section, we combine the advantages from two sides: 1) spatial feature as majority of feature-to-label mapping segmentation proposed; 2) temporal feature as AR-SI math describes. The overall solution is as the Fig.4 shown.

Each loop in the system takes in raw LiDAR point cloud containing spatial reflected information and remission measurement. A sequence of LiDAR frame, from current time step T_{N-1} to the frame back N steps T_0, generates a temporal vector using spherical coordinate projection. The temporal vector contains P residual depth (details explained in section iv). This set of residual depth, combing spatial raw vector from current time step T_{N-1}, will be pass into AR-SI filter. The AR-SI filter use Exp. 1 to 5 to check whether it is a static object or not. Temporal vector is turn into all zero in case of AR-SI filter error is lower than ϵ, which means AR-SI consider it as non-moving. This operation benefits the CNN training because non-zero trajectory value is actual noise if the object is confirmed as static, especially in case that the training problem is binary. In order to distinguish different static objects, spatial vector maintain the same for CNN training. In the aspect of dynamic ones, temporal vector maintains the residual depth value to concatenate with spatial vector to formulate the dense input of the sparse tensor module.

Then, dense input tensors are voxelized to construct sparse tensor for the CNN considering fast inference requirements in embedded system. Sparse tensor indexes each voxel with coordinate (c_x^N, c_y^N, c_z^N) with b^N, and attaches the according feature (f_x^N, f_y^N, f_z^N). Feature details will be illustrated in the Section. IV. The feature tensor is pass into the Encode-Decode sparse CNN to construct the sparse convolution. By dividing label into three categories only: unlabel, dynamic and static, we train the CNN model prediction to map the label accordingly. The prediction result is shown via discovering dynamic objects in red mask, which is the vital potential threat to autonomous driving safety.

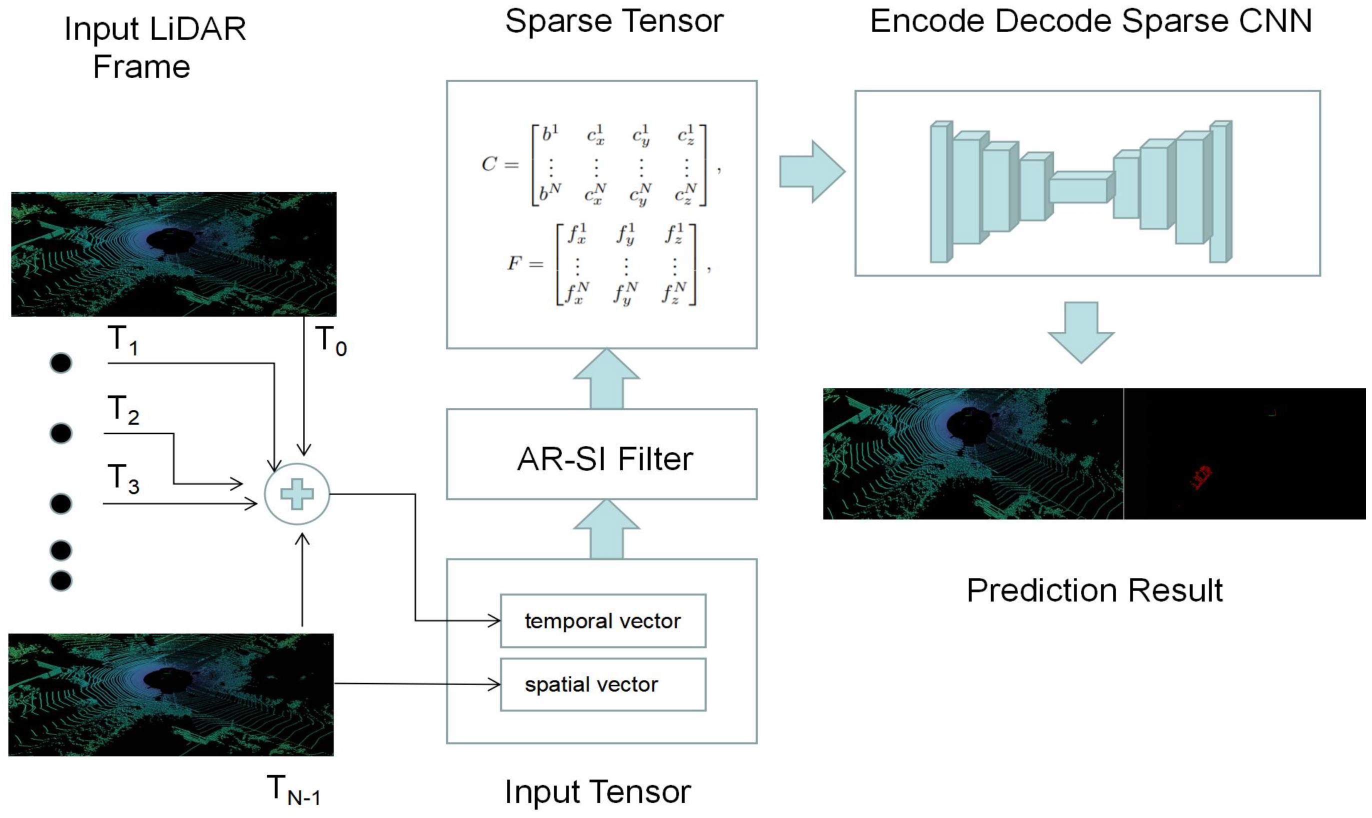
Input LiDAR Frame
Sparse Tensor
Encode Decode Sparse CNN
$C = \begin{bmatrix} b^1 & c_x^1 & c_y^1 & c_z^1 \\ \vdots & \vdots & \vdots & \vdots \\ b^N & c_x^N & c_y^N & c_z^N \end{bmatrix}$,
$F = \begin{bmatrix} f_x^1 & f_y^1 & f_z^1 \\ \vdots & \vdots & \vdots \\ f_x^N & f_y^N & f_z^N \end{bmatrix}$,
T1
T0
T2
T3
AR-SI Filter
Prediction Result
temporal vector
spatial vector
TN-1
Input Tensor

Fig. 4: Overview of our proposed method

V. Sparse Tensor Based Moving Object Segmentation

A. Sparse Tensor and Sparse Convolution

Moving-object segmentation leverages the power of captured point cloud geometric information to extract low-level features. The features are used to fill into learning network in the following step to predict segmentation mask. However, typical methods such as PointNet [16] operate the original unconstructed data directly to fetch ready-for-train geometric features, which consumes huge computation resources. In this work, we adapt sparse tensor to improve computational efficiency to make the moving-object segmentation practical in embedded system.

The sparse tensor uses a voxel coordinate matrix C to index an associated feature matrix F. The first step is to voxelize the point cloud and make one voxel contain only one point cloud, which generates the according voxel coordinate C. This step takes the advantage of removing redundant points in a same voxel, which avoids empty 3D voxel computation so that massive memory footprint generation can be avoided. The second step is to calculate the associated feature matrix F according to the generated index coordinate C:

$$C = \begin{bmatrix} b^1 & c_x^1 & c_y^1 & c_z^1 \\ \vdots & \vdots & \vdots & \vdots \\ b^N & c_x^N & c_y^N & c_z^N \end{bmatrix}, \quad F = \begin{bmatrix} f_x^1 & f_y^1 & f_z^1 \\ \vdots & \vdots & \vdots \\ f_x^N & f_y^N & f_z^N \end{bmatrix}, \quad (6)$$

where for point i, $\{c_x^i, c_y^i, c_z^i\} \in \mathbb{Z}^3$ is the voxelized integer coordinate. The b^i is the batch index attached to the coordinate $\{c_x^i, c_y^i, c_z^i\}$. $\{f_x^i, f_y^i, f_z^i\} \in \mathbb{R}^3$ is the float-type coordinate captured by a 3D LiDAR, $i \in [1, N]$, N is the number of points after quantization, which is determined by the voxel size, $N \le N_o$, where N_o is the original number of points before quantization.

The sparse convolution [28] inputs a sparse tensor and also outputs a sparse tensor with varying batch length. Specifically, it first generates the coordinate matrix C^{out} for the output sparse tensor from the given input coordinate matrix (details are described in [28]). Then, it calculates the feature vector $\mathbf{f}_\mathbf{c}^{\text{out}}$ for an output coordinate $\mathbf{c}$ with the formula [29]:

$$\mathbf{f}_\mathbf{c}^{\text{out}} = \sum_{\mathbf{s} \in \mathcal{N}(\mathbf{c}, K)} W_\mathbf{s} \mathbf{f}_{\mathbf{c}+\mathbf{s}}^{\text{in}}, \quad \mathbf{f}_\mathbf{c}^{\text{out}} \in F^{\text{out}}, \ \mathbf{c} \in C^{\text{out}}, \quad (7)$$

where $\mathbf{s}$ is the offset to find the corresponding input coordinates, they are centered in $\mathbf{c}$ position and covered by the kernel size K, which is denoted as $\mathcal{N}(\mathbf{c}, K)$. $\mathbf{f}_{\mathbf{c}+\mathbf{s}}^{\text{in}}$ means the input feature vector at the input coordinate $\mathbf{c}+\mathbf{s}$. $W_{\mathbf{s}}$ represents the coefficient parameter, which is to be learned through the training process. The output sparse tensor, coordinate matrix $C^{\text{ out}}$ and feature matrix $F^{\text{ out}}$, can be generated via training.

B. Spacial Residual Depth

Inspired by the observation and conclusion in [1] and [11], residual depth projected in spherical coordinates is the vital factor in distinguishing moving and non-moving objects. We take residual depth $d^i_{(u,v)}$ [1] as the trajectory symbol in Exp. 5, so our AR-SI filter expression can be calculated as:

$$e_i = \left\| \hat{d^i_{(u,v)}} - d^i_{(u,v)} \right\|. \tag{8}$$

If the residual depth error is less than the calculated filter value ($e_i < \epsilon$), we consider it as static and turn all the value in temporal vector to zero. In Exp. 8, $d^i_{(u,v)}$ is the residual depth in position (u, v) of the projected spherical coordinates. The $\hat{d^i_{(u,v)}}$ means using residual depth data in time slice $i-1$ and the data back P steps in Exp.4. If an object is static, the sum of its depth variance within this slicing temporal window should be less than ϵ. Considering that noise exists in the sensor, the ϵ can use Monte-Carlo sampling to calculate.

Guided by the goal from Exp. 8, we require the value of the residual depth $d^i_{(u,v)}$ in time sequence $i+1-P$ to i. We use a range projection of LiDAR data to generate this residual depth d. The strategy is to convert each LiDAR point at time i, $P = (x_i, y_i, z_i)$ to time slice $i+1-P$ coordinate. Two range information $r^i_{(u,v)}$ and $r'^{i}_{(u,v)}$ exists in the same projected point (u, v) which can be mapped to LiDAR point (x, y, z) in 3D space. The depth variance at the same point can be measured as [1]:

$$d^i_{(u,v)} = \frac{\left\| r^i_{(u,v)} - r'^{i}_{(u,v)} \right\|}{r^i_{(u,v)}}. \tag{9}$$

The range information $r'^{i}_{(u,v)}$ is calculated as the following:

$$p_{(u,v)}{}^{i} = T^{i+1-P \to i} \cdot p^{i+1-P}_{(u,v)}, \tag{10}$$

$$r'^{i}_{(u,v)} = \sqrt{p_x^{i,(u,v)^2} + p_y^{i,(u,v)^2} + p_z^{i,(u,v)^2}}, \tag{11}$$

where $T^{i+1-P \to i}$ means the pose transformed from time step $i+1-P$ to i. We describe a consecutive sequence from time 0 to N of pose information as T^1_0, ..., T^N_{N-1}, where T^1_0 is the pose transforming from time step 0 to 1. To benefit from the continued product character of homogeneous coordinates, we represent the LiDAR data as $p_i = (x_i, y_i, z_i, 1)$.

The last step is to associate the 3D LiDAR point to 2D projected point, saying: $R^3 \to R^2$ to spherical coordinates, and finally to image coordinates:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right]\omega \\ \left[1 - \left(\arcsin(zr^{-1}) + f_{down}\right)f^{-1}\right]h \end{pmatrix}, \tag{12}$$

where (u, v) are image coordinates representation mapping the 3D LiDAR point (x, y, z), f defines the sensor vertical field of view as $f = |f_{down}| + |f_{up}|$. Following the work of [39], we considered a 3D 360°field-of-view during the projection process. The h, ω stands for the height and width of the desired range image. The calculated coordinates (u, v) serves as the index in a hash map which contains the information of each point cloud data x, y, z coordinates, its remission e, and its range information r.

C. Network Architecture

We adapt the most promising Encoder-Decoder architecture to our sparse tensor based network design. Inspired by [27], the overall architecture of the proposed network is shown as Fig.5. The input to proposed network is a sparse tensor containing the spatial information from the raw 3D LiDAR data and the temporal information from residual image. Specially, each pixel (u, v) of the projected range image contains a vector $(x, y, z, r, e, d_0, d_1, ..., d_P)$ [1], where (x, y, z, r) is the spatial information of the raw 3D LiDAR point cloud. Range r of each point can be calculated as $r = \sqrt{x^2 + y^2 + z^2}$.

The overall network can be divided into four modules, namely, contextual enhance module, encoder module, decoder module and prediction logit module. The contextual enhance module plays a role of fusing local feature with a larger one at the beginning of the network. However, due to the varying length of the sparse tensor and multiple dimensional input (feature and coordinate), we use MinkowskiEngine [40] to handle the batch for each training.

The contextual convolution follows the pattern in [27] which has one $1*1$ and two $3*3$ kernels with dilation rate of $(1, 2)$. Concatenating two context block to generate a N*32 tensor serves as the input of the encoder module. The purpose of this convolution is to make the $1*1$ sparse convolution captures the local information while $3*3$ gets the global feature. Via concatenating these two features together, we get a $(N, 32)$ sparse tensor that captures context information of the raw 3D LiDAR point cloud.

The encoder module contains four residual blocks for down-sampling sparse convolution. Each residual block has 3 sparse convolution, ReLU, BatchNorm layers. All these basic blocks are revised using MinkowskiEngine to fit the sparse convolution requirement. These residual blocks output concatenate together to formulate output using residual principle as [41]. One dropout and one pooling layer attach to the end of the residual block to avoid the overfiting problem.

The decoder module has a sequence of upsampling blocks similar to [27]. Each upsample block has 3 sparse convolution, ReLU, BatchNorm layers. 3 upsampling block outputs are concatenated together. One dropout layer follows this result at the end of the decoder module. Logit module maps the upsampling output to point cloud label to generate the final prediction. F block is a tool in MinkowskiEngine used to convert output sparse tensor to normal tensor for segmentation result.

D. Loss Function

The class label imbalance problem exists in both typical dataset and practical scene. For example, in the campus where

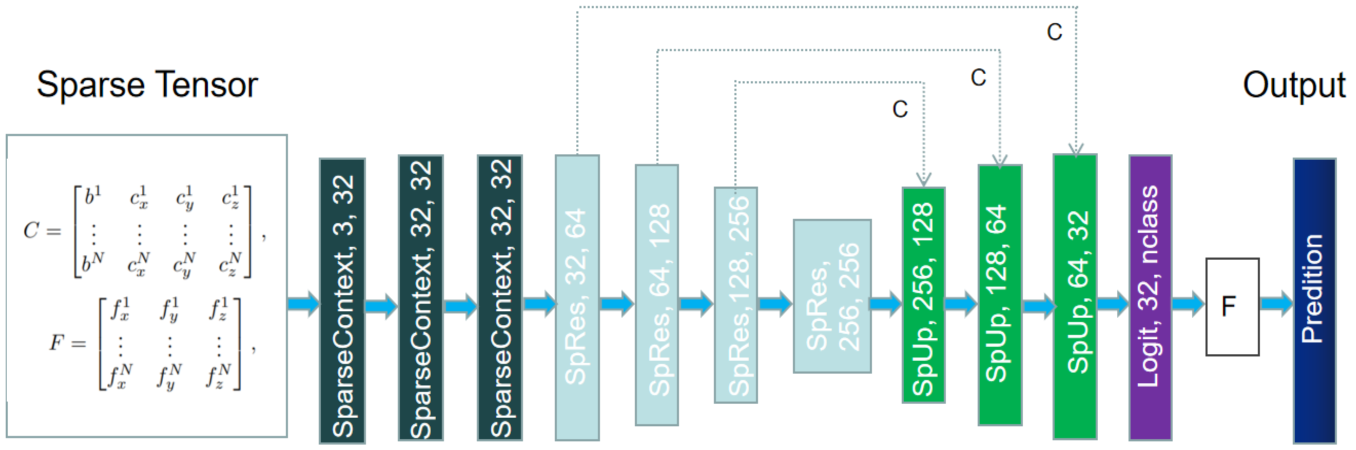
Sparse Tensor
$C = \begin{bmatrix} b^1 & c_x^1 & c_y^1 & c_z^1 \\ \vdots & \vdots & \vdots & \vdots \\ b^N & c_x^N & c_y^N & c_z^N \end{bmatrix},$
$F = \begin{bmatrix} f_x^1 & f_y^1 & f_z^1 \\ \vdots & \vdots & \vdots \\ f_x^N & f_y^N & f_z^N \end{bmatrix},$
SparseContext, 3, 32
SparseContext, 32, 32
SparseContext, 32, 32
SpRes, 32, 64
SpRes, 64, 128
SpRes,128, 256
SpRes, 256, 256
SpUp, 256, 128
SpUp, 128, 64
SpUp, 64, 32
Logit, 32, nclass
F
Predition
C
C
C
Output

Fig. 5: Architecture of EmPointMovSeg: Sparse Tensor Based Encode-Decode Sparse Convolution Network

we collect data, static objects are the majority of cases comparing to dynamic ones. As a result, training becomes useless if one sequence overfits static ones and ignores dynamic cases, which yields poor performance in network inference.

Therefore, we propose a strategy to cope with this class labelling imbalance problem similar to [27]: calculating loss by emphasizing the effect of under-presented classes via taking its appear frequency into account. The loss formula is expressed as:

$$\mathcal{L}_{wce}(y,\hat{y}) = -\sum_i a_i p(y_i) log(p(\hat{y_i})) \quad with \quad a_i = 1/\sqrt{f_i}, \tag{13}$$

where $p(y_i)$ stands for the probability of label y_i and $p(\hat{y_i})$ means the probability of prediction label of y_i. f_i indicates the frequency. i is the i_{th} class which contains only three class in moving-object segmentation. This loss function takes the class imbalance problem into consideration to avoid the partially overfitting certain class phenomenon.

VI. Experimental Evaluation

This paper targets at developing a practical moving-object segmentation solution in 3D LiDAR scans. We construct our first experiment on a most representative dataset to prove the efficiency of our proposed method. The second evaluation is performed on a practical embedded system platform to check the real application effect. The following evaluation results show that: 1) our propose method achieves comparative segmentation performance while maintain a low computational cost; 2) the relatively comparative low computational cost leads to practical moving-object segmentation success, which is able to apply proposed on-line segmentation method in safety-critical tasks.

A. Evaluation based on Dataset

To fulfill the on-line segmentation performance in embedded system, the first experiment in this section is to support our claim of achieving comparative segmentation performance while low inference time required.

We first test all the methods on a powerful server equipped with:

* Intel(R) Xeon(R) Gold 5118 CPU.
* Sytem memory up to 128GB.
* NVIDIA GeForce RTX 3090 Graphic Card.

The dataset we choose should fulfill the harsh autonomous driving requirements comparing to traditional dataset just for single object segment:

* Large scale sequential 3D LiDAR-based data including cloud point spacial and reflection information.
* Containing groundtruth in dynamic traffic participants with distinct classes, which can reason our model rationality.

We choose SemanticKITTI [21] as our testing dataset. Besides fulfilling the above requirements, SemanticKITTI provides LiDAR-based pose information so that odometry, object detec- tion, tracking, semantic segmentation, panoptic segmentation, and scene completion tasks are proposed as open challenge issues for dataset users. SemanticKITTI are currently the most representative benchmark for moving-object segmentation.

SemanticKITTI dataset contains 22 dense LiDAR individual scans. For sequences $00-10$, dense annotations were provided for semantic scene interpretation, like semantic segmentation and semantic scene completion. For sequences $11-21$, labels for the test set are not provided so that we consider them as test scans. SemanticKITTI contains in total 28 semantic classes such as vehicles, pedestrians, buildings, roads, etc. Thanks to the effort in [1], moving-object segmentation benchmark contains three types of classes: unlabelled, moving and non-moving. The actually moving vehicles and humans belong to moving objects and all other classes belong to the non-moving objects. Here, we choose sequence 08 for validation during training process as [1] does.

For quantitative study in moving-object segmentation performance, we use intersection-over-union (IoU) metric [42] over the objects:

$$IoU = \frac{TP}{TP+FP+FN}, \tag{14}$$

where TP, FP, and FN correspond to the number of true positive, false positive, and false negative predictions for the class.

1) Ablation Study on Input and AR-SI filter: According to the theory in Section 3, the AR-SI filter targets at calculating rational threshold value to separate moving objects from non-moving ones. This step involves two modules to generate final sparse input tensor:

* Efficient temporal LiDAR sequence module containing rational P value in Exp.1, which stands for the number of time steps we need to track back.
* Robust AR-SI filter module blocks residual depth error caused by sensor ego-motion noise to affect the sparse convolution network.

For the first module, according to the theory in [15], we compare different P value to maximize the IoU performance. Current methods do not give a rational interpretation about the temporal length theoretically. In this paper, we leverage the AR-SI theory to calculate P value, which presets P value to 10 according to [15] and compares the IoU performance around this preset value.

TABLE I: IoU segmentation result using different P value

P value	4	6	8	10	12
ResUnet	0.21	0.25	0.28	0.33	**0.34**
EmPointMovSeg	0.31	0.37	0.42	0.46	**0.47**

From table I, we can observe that selecting P value to 12 have best IoU performance. While the tracking back length set to $4, 6, 8$, we can see the IoU is improving because the depth information enhances. This enhancement serves as the temporal feature in sparse tensor so that IoU performance can be improved. However, there is not that much improvement when P value is bigger than 10. The con side of prolonging the temporal sequence is that the residual image requires more time to be ready for the CNN inference (explained in the 4th subsection). We have to balance the computational expense and segmentation accuracy so that 10 is chosen as the most rational sequence length P value.

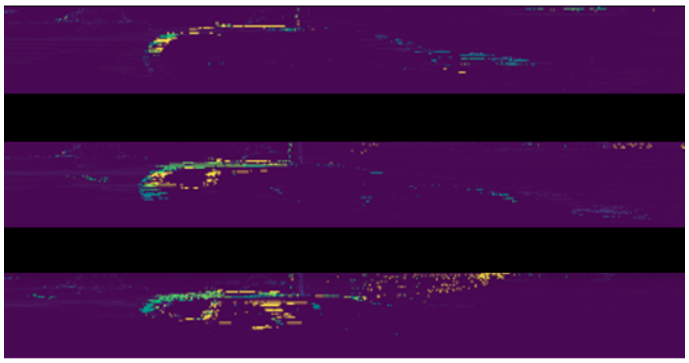
Fig. 6: Residual depth when P is set to 4, 8, 10, respectively. A projected moving car becomes more obvious when the P value increases in residual image.

As the AR-SI theory indicates, a static object projected depth equals to zero in a sequential residual depth image, which means the same color as the background. In other words, moving ones owns contrasting color to the background. As Fig. 6 shows, the residual depth feature of moving object becomes more obvious when the P value increases from 4 to 10. This feature contributes to CNN to get a better IoU performance in segmentation task. What is more, residual depth feature of the non-moving objects becomes zero helps to distinguish these two classes significantly. However, due to ego-motion sensor error or pose calculation error, non-moving objects are sometimes considered as moving ones. We use AR-SI filter to block this error to propagate in the CNN prediction. We prove the AR-SI filter effect in both sparse and dense convolution methods with P value fixed in 10. Here we select or reconstruct ResUnet, SpSequenceNet [43] and our proposed EmPointMovSeg in sparse convolution methods. For dense convolution solutions, Unet [8], SalsaNext [27] and KPConv [44] are chosen respectively.

TABLE II: Sparse convolution network MOS IoU results with/without AR-SI filter

	Non-filter	AR-SI filter
ResUnet	0.33	**0.36**
SpSequenceNet	0.42	**0.46**
EmPointMovSeg	0.46	**0.51**

TABLE III: Dense convolution network MOS IoU results with/without AR-SI filter

	Non-filter	AR-SI filter
Unet	0.12	**0.15**
SalsaNext	0.51	**0.53**
KPConv	0.59	**0.61**

We choose SemanticKITTI scan 00 sequence to check all the non-moving object. We calculate outlier threshold e_i value via the same theory in [15] which regards a data point outside of (mean ± 6std) as an outlier. Both Table. II and Table. III indicate AR-SI filter is able to improve segmentation results in both sparse and dense convolution approaches.

2) Inference Time Study: As shown in Fig.4, the total inference time contains the residual depth calculation time and the CNN inference time. We fix P value to 10 as last section discussed and discover the key constraint on time consumption: max dense residual depth calculation uses 251ms and CNN inference uses 42ms in scan 00 of SemanticKITTI. Since the LiDAR frequency of SemanticKITTI is around 10HZ, traditional method [12], [27], using residual depth is not able to operate on-line segmentation task on a powerful server, not even to mention embedded system.

To overcome this bottleneck, we choose to voxelize each scan of point cloud to generate sparse tensor. Less depth information is generated in order to accelerate the residual depth calculation step in inference. The voxel size is the key factor in voxelization, which influences the granularity of a sparse tensor so that efficiency is affected. Smaller voxel size can achieve better IoU performance but efficiency would be reduced. We test the voxel sizes with 0.1m, 0.3m, 0.5m and

0.7m in order to discover the best segmentation performance while fulfilling the real time requirement in LiDAR.

TABLE IV: Inference time for different voxel sizes

voxel size	0.1m	0.3m	0.5m	0.7m
Time	135.49ms	78.23ms	60.88ms	**52.34ms**
IoU	**0.51**	0.43	0.32	0.25

As shown in Table IV, with AR-SI filter turned on and P value fixed to 10, we can get low segmentation performance (0.25) if we want to speed up the EmPointMovSeg using $0.7m$ voxel. However, given the practical velodyne LiDAR frequency is below 10HZ, we choose voxel size equals to $0.3m$ to achieve real time performance. With the model inference time $36.14ms$, the total inference time is $78.23ms + 36.14ms = 114.37ms$, which can fulfill the on-line requirements in LiDAR segmentation with an acceptable IoU performance 43.13%. Comparing to 62.5% [27] in dense residual image, our EmPointMovSeg suggests a way of balancing computational time and segmentation IoU, which makes online LiDAR moving-object segmentation possible in embedded system.

3) Moving-object Segmentation Performance Comparative Study: In this section, we create baselines in the following aspects for comparative study:

* **No-Seq Non-Sparse Method:** This baseline indicates traditional segmentation method which input raw point cloud data and output segmentation label directly. We **do not** input sequential data and use each scan raw data to predict segmentation mask. We choose the most representative method **PointNet** as this baseline. Also, considering the performance on embedded system, we also add **MINet** [45] into this baseline for comparison. For MINet, projected image is compulsory input to the network so that the total processing time should take project map calculation into account.
* **Seq Non-Sparse Method:** This baseline means using the sequential raw point cloud containing residual depth information as input. Combing the spacial and temporal information as input, we choose residual depth equals to 10 which is the same as our proposed method. We train the network using the label in [1] as current majority dense segmentation methods. Here, we select most recent method **SalsaNext** as this candidate. Also, we reconstruct **MINet** to serve as sequential segmentation competitor for the reason that: MInet is projected-based multi-label LiDAR segmentation, which is very similar to our proposed method. In order to keep fairness in the comparison, we set the sequence length N to 10 in MINet. The time in total inference progress should add projected image time consumption into calculation.
* **No-Seq Sparse Method:** This baseline picks up each frame of raw point cloud converting into sparse tensor. This sparse tensor, selecting $0.3m$ voxel size as discussed in the last section, generates the input of the sparse convolution network which outputs the final prediction result. Here, we use **MinkUNet** in MinkowskiEngine [40], the most representative segmentation method in sparse tensor, to serve as this baseline.
* **Seq Sparse Method:** The final method, which is our proposed method, **EmPointMovSeg**, use sequential point cloud as input and use the sparse convolution network architecture as Fig. 5 shown. This method works as sequential input and sparse tensor/convolution competitor.

Since one complete segmentation process involves raw point cloud captured to residual depth images generation if sequential data is used, and inference via CNN network. As a result, residual depth image generation time should be added into final inference time in case that the temporal sequential data is required. Otherwise, no such time slot is added, and only raw point cloud is used directly. All the residual depth images are calculated in Python3.8 environment. The quantitative results for the comparison are shown in Table. V.

TABLE V: Comparative study of different segmentation methods in IoU and inference time performance

	IoU	Time
PointNet [16]	0.16	273.78ms
MINet [45]	0.19	**92.21ms**
SalsaNext/N = 10 [27]	**0.62**	293.12ms
MINet/N = 10	0.31	105.29ms
MinkUNet [40]	0.17	106.46ms
EmPointMovSeg	0.43	114.37ms

From Table.V we can see that our proposed EmPointMovSeg is able to achieve on-line LiDAR data segmentation while maintaining a relatively high IoU performance. Although MINet with sequential input can satisfy the on-line segmentation requirement, lower IoU is achieved comparing to EmPointMovSeg. Since PointNet baseline uses raw point cloud and process directly, dense and spacial varying density data requires long inference time, which can neither perform on-line segmentation, nor accurate semantic segmentation in 3D LiDAR data. MINet is able to achieve best on-line performance ($21ms$ in reference time and $71.21ms$ in projected image calculation time). But low IoU performance in movable class segmentation because no sequential feature is imported. SalsaNext serves as our non-sparse convolution competitor using sequence point cloud to calculate residual depth image and then use CNN network for inference. Although best achievement in segmentation IoU, we can see dense raw point cloud data generates high residual image computational consumption which is not practical for real-time segmentation. The real LiDAR frequency ranges from $8HZ$ to $20HZ$ [46], which means on-line segmentation inference time varying from $125ms$ to $50ms$. Under this real-time constraint, MINet using sequential data can give an acceptable movable class IoU performance. For rest two methods using sparse tensor, both MinkUNet and our proposed EmPointMovSeg can satisfy the on-line segmentation requirements. However, with poor IoU performance of MinkUNet, our proposed EmPointMovSeg can narrow the gap between accurate segment and real-time performance. Our proposed EmPointMovSeg is not able to achieve best IoU as dense convolution network SalsaNext which use sequential and dense input. Instead, considering practical use in embedded system on autonomous vehicle, we sacrifice IoU for an rational ratio but give possibility for on-line LiDAR segmentation task, which is more important for safety-critical cases.

4) Moving-object application study: In this section, we analyze the effect of AR-SI to assist LiDAR-based odometry/SLAM. SLAM applications highly rely on the static objects to converge to an precise pose estimation. Dynamic objects should be constructed as outliers in SLAM reprojection error. As a result, MOS and AR-SI methods, both having moving/non-moving objects separation character, are supposed to improve the rotational accuracy.

TABLE VI: KITTI Odometry Benchmark Results

	SuMa++	SuMa+MOS	SuMa+AR-SI
Seq.00-10 (Train)	0.32/0.74	0.32/0.71	**0.31/0.69**
Seq.11-21 (Test)	0.34/1.10	0.34/1.08	**0.32/1.05**

Similar to [1], we apply our enhanced prediction mask before feeding the point cloud into the SLAM pipeline in KITTI odometry benchmark. We evaluate these odometry methods, namely, SuMa++ [47], SuMa+MOS and our proposed SuMa+AR-SI. We select two key values to illustrate odometry accucacy: relative rotational error in degrees per 100m and relative translational error in $\%$. The quantitative results in Table. VI shows MOS and AR-SI methods are the same as our claim of improving the odometry results slightly. Base on the well-designed semantic-enhanced SuMa, our proposed method AR-SI raise up the accuracy a little because the AR-SI method directly targets on filtering out moving objects.

B. Study on Practical Embedded System

In this section, we focus on evaluating our algorithm on a real embedded system. In aspect of safety, we mainly check two capabilities in details:

1) Whether the algorithm runtime can fullfill real-time detection requirement;
2) Whether the algorithm can detect real potential threat to the vehicle.

1) Embedded system hardware: The runtime hardware we choose NVIDIA Jetson Xavier Developer kit. This platform contains 512-core NVIDIA Volta GPU with 64 tensor cores, 16GB 256-bit wide LPDDR4X memory and 64-bit 8-core NVIDIA Carmel CPU on board. The real-time 3D LiDAR sensor have 16 channels, vertical field of view ranging from $+15.0°$ to $-15.0°$ and 10 HZ rotation rate. The LiDAR sensor communicates with Xavier board via gigabit Ethernet interface.

2) Practical results: The proposed method involves two operations to calculate the final segmentation result: first to use odometry data to generate residual depth, then to combine captured data $(x, y, z, remission)$ with residual depth for model inference.

As shown in Fig. 8, EmPointMovSeg is evaluated in the most representative segmentation dataset, SemanticKITTI. We

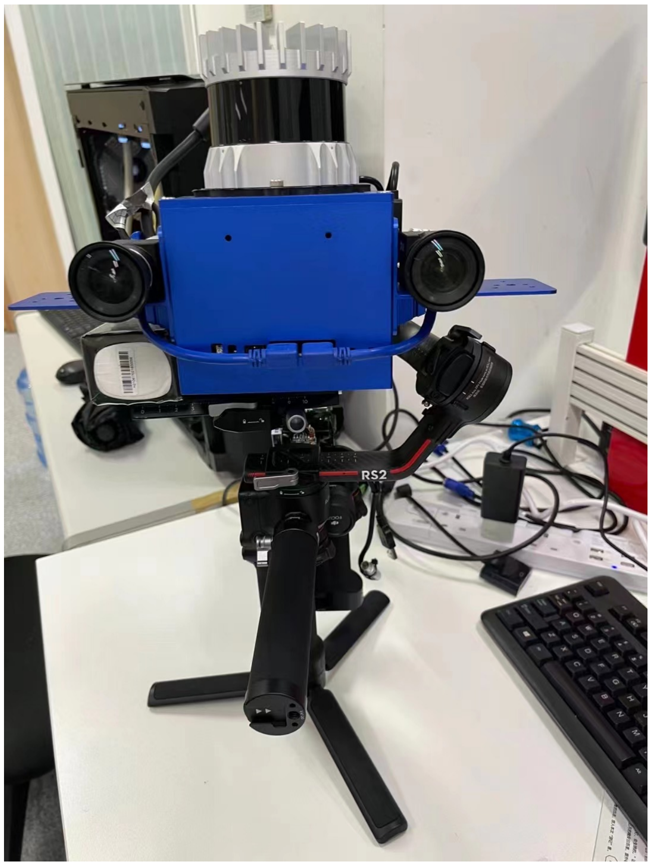

Fig. 7: Embedded system equipped with velodyne vlp-16 LiDAR sensor.

can reach a consensus that EmPointMovSeg distinguishes moving car clearly from the static environment around. In Fig. 9, EmPointMovSeg is deployed in the mentioned equipment in Fig. 7 and process real-time point cloud. The original road has fence and tree on the left side and a truck on the right side. From Fig. 9, we can see clearly our proposed method, EmPointMovSeg, filters out the static objects on the left side and predicts a truck as moving-object using red mask. Also, benefit from AR-SI filter, the fence point cloud on the right side is also filtered. This result proves that EmPointMovSeg, not only manages the segmentation task on a resource-limit embedded system platform, but also discover the potential safety-critical threat to self-driving car. This achievement is meaningful to practical moving-object segmentation task in autonomous driving.

EmPointMovSeg takes $81.47ms$ in pose estimation and input tensor preparation, and then use $36.14ms$ for inference the prediction mask. The total time consumption is $81.47ms + 36.14ms = 117.61ms$, which stands for $1/117.61ms = 8.5HZ$ in the above mentioned embedded system. This performance can fully satisfy on-line LiDAR segmentation which is novel to current existing methods only evaluated in powerful computer or GPU. EmPointMovSeg narrows the gap between accurate segmentation performance and high computation time consumption, which is a vital step for practical LiDAR segmentation use in autonomous driving.

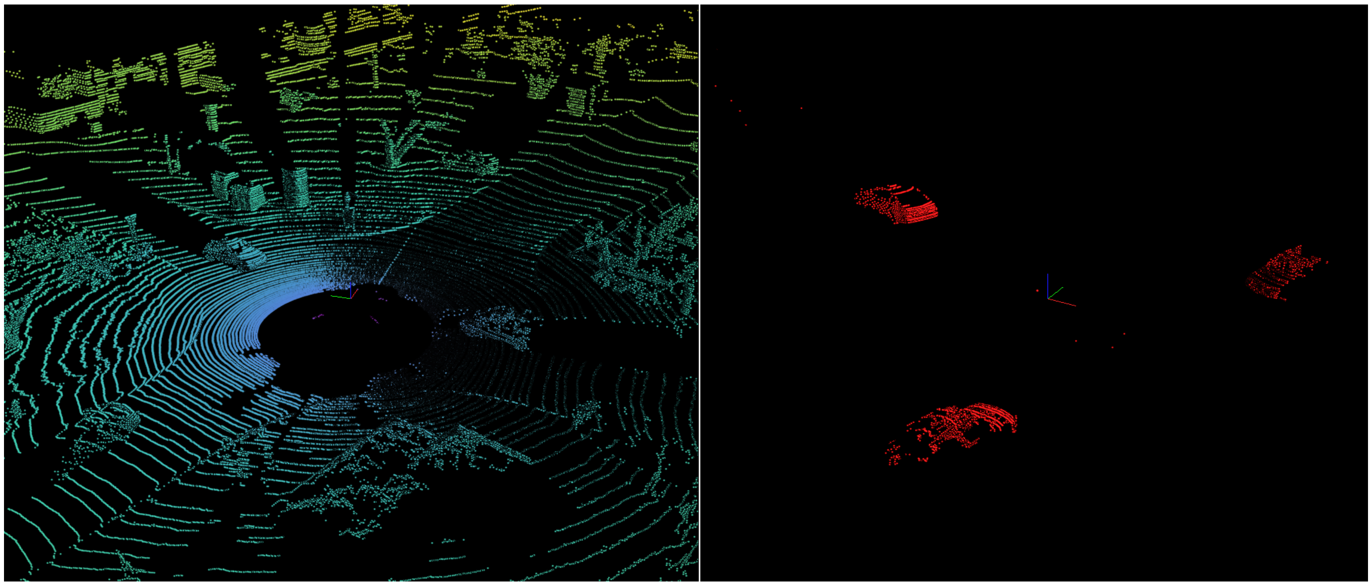
Fig. 8: 3D LiDAR MOS using EmPointMovSeg in SemanticKITTI.

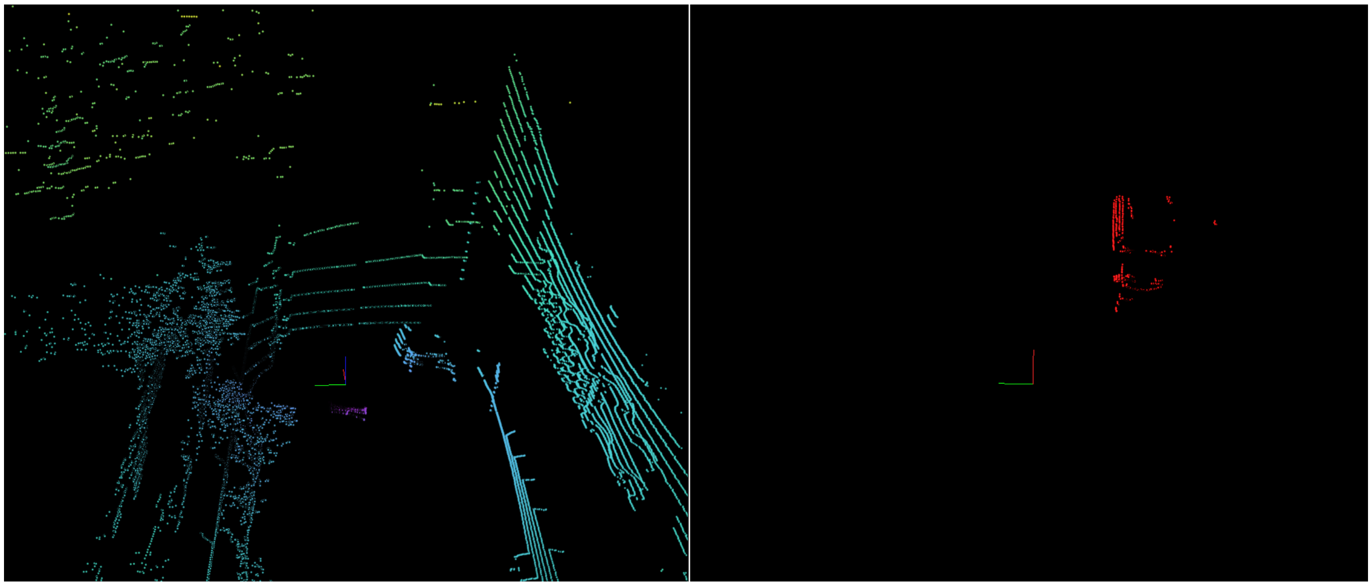
Fig. 9: 3D LiDAR MOS using EmPointMovSeg in practical HKUST campus.

What is more, EmPointMovSeg provides a hint to accelerate the safety-critical task in autonomous driving reaction rather than optimizing the hardware only. It provides a solution in algorithm calculation that is cost insensitive.

VII. Conclusion

In this work, we first figure out the importance of moving-object segmentation in autonomous driving safety-critical scene context analysis and dig into current drawbacks that constrains moving-object segmentation used in reality. We theoretically solve the moving-object problem using AR-SI math, which is novel comparing to current existing large scale 3D LiDAR segmentation. Using the the advantages of combining the temporal and the spatial features in AR-SI filter, we propose our sparse tensor based CNN network, EmPointMovSeg, to balance the process efficiency and segmentation accuracy. The balance is the key in making large scale 3D LiDAR segmentation practical in autonomous driving embedded system, and most importantly, provides possibility for the mission of safety-critical task reaction. This algorithm is cost insensitive so that it can be transplanted to other platforms easily. The evaluation results show that our proposed scheme works efficiently in both synthetic dataset and real hardware platforms.

References

[1] 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,” *arXiv preprint arXiv:2105.08971*, 2021.

[2] Z. Shen, L. Han, C. Ma, Z. Jia, T. Li, and Z. Shao, "Leveraging the interplay of raid and ssd for lifetime optimization of flash-based ssd raid," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[3] Y. Lyu, L. Bai, and X. Huang, "Real-time road segmentation using lidar data processing on an fpga," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[4] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: a review," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[5] H. Zhou, X. Zhu, X. Song, Y. Ma, Z. Wang, H. Li, and D. Lin, "Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation," *arXiv preprint arXiv:2008.01550*, 2020.

[6] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation," *arXiv preprint arXiv:2103.12978*, 2021.

[7] X. Yan, J. Gao, J. Li, R. Zhang, Z. Li, R. Huang, and S. Cui, "Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion," *arXiv preprint arXiv:2012.03762*, 2020.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[9] R. Razani, R. Cheng, E. Taghavi, and L. Bingbing, "Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions," *arXiv preprint arXiv:2103.08852*, 2021.

[10] M. Gerdzhev, R. Razani, E. Taghavi, and L. Bingbing, "Tornado-net: multiview total variation semantic segmentation with diamond inception module," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9543–9549.

[11] G. Singh, S. Gupta, M. Lease, and C. Dawson, "Range-net: A high precision streaming svd for big data applications," *arXiv preprint arXiv:2010.14226*, 2020.

[12] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[13] B. Wu, A. Wan, X. Yue, and K. 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)*. IEEE, 2018, pp. 1887–1893.

[14] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. 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)*. IEEE, 2019, pp. 4376–4382.

[15] Z. He, Y. Chen, E. Huang, Q. Wang, Y. Pei, and H. Yuan, "A system identification based oracle for control-cps software fault localization," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 116–127.

[16] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 652–660.

[17] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[19] vested, "Self-driving technology: Lidar vs camera," https://vested.co.in/blog/self-driving-technology-lidar-vs-camera/.

[20] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[21] t. C. V. G. Photogrammetry Robotics Group and the Autonomous Intelligent Systems Group, "Semantickitti, a dataset for semantic scene understanding using lidar sequences," http://www.semantic-kitti.org/index.html.

[22] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[24] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1625–1632.

[25] R. Razani, R. Cheng, E. Li, E. Taghavi, Y. Ren, and L. Bingbing, "Gp-s3net: Graph-based panoptic sparse semantic segmentation network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 076–16 085.

[26] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "Efficientlps: Efficient lidar panoptic segmentation," *arXiv preprint arXiv:2102.08009*, 2021.

[27] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving," *arXiv preprint arXiv:2003.03653*, 2020.

[28] R. Cheng, R. Razani, Y. Ren, and L. Bingbing, "S3net: 3d lidar sparse semantic segmentation network," *arXiv preprint arXiv:2103.08745*, 2021.

[29] Y. Sun, W. Zuo, H. Huang, P. Cai, and M. Liu, "Pointmoseg: Sparse tensor-based end-to-end moving-obstacle segmentation in 3-d lidar point clouds for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 510–517, 2020.

[30] T. M. Quan, D. G. Hildebrand, and W.-K. Jeong, "Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics," *arXiv preprint arXiv:1612.05360*, 2016.

[31] J. Gehrung, M. Hebel, M. Arens, and U. Stilla, "An approach to extract moving objects from mls data using a volumetric background representation." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 4, 2017.

[32] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE robotics and automation letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[33] P. Ruchti and W. Burgard, "Mapping with dynamic-object probabilities calculated from single 3d range scans," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6331–6336.

[34] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Rigid scene flow for 3d lidar scans," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1765–1770.

[35] A. Dewan and W. Burgard, "Deeptemporalseg: Temporally consistent semantic segmentation of 3d lidar scans," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2624–2630.

[36] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[37] M. U. Khan, S. Li, Q. Wang, and Z. Shao, "Cps oriented control design for networked surveillance robots with multiple physical constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 778–791, 2016.

[38] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[39] F. Zampella, A. R. Jiménez, F. Seco, J. C. Prieto, and J. Guevara, "Simulation of foot-mounted IMU signals for the evaluation of PDR algorithms," in *2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011, Guimaraes, Portugal, September 21-23, 2011*, 2011, pp. 1–7.

[40] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[41] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[42] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*. Springer, 2016, pp. 234–244.

[43] H. Shi, G. Lin, H. Wang, T.-Y. Hung, and Z. Wang, "Spsequencenet: Semantic segmentation network on 4d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4574–4583.

[44] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[45] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall, "Multi-scale interaction for real-time lidar data segmentation on an embedded platform," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 738–745, 2021.

[46] V. LiDAR, "Velodyne lidar puck datasheet," https://www.mapix.com/wp-content/uploads/2018/07/63-9229_Rev-H_Puck-_Datasheet_Web-1.pdf.

[47] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4530–4537.