IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 3, 2021, accepted November 15, 2021, date of publication November 25, 2021,
date of current version December 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130257

Semantic Point Cloud-Based Adaptive Multiple Object Detection and Tracking for Autonomous Vehicles

SOYEONG KIM, JINSU HA, AND KICHUN JO, (Member, IEEE)
Department of Smart Vehicle Engineering, Konkuk University, Gwangjin-gu, Seoul 05029, Republic of Korea

Corresponding author: Kichun Jo (kichun@konkuk.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] under Grant 2019R1G1A1099806 and Grant 2020R1C1C1007739, and in part by 'the Competency Development Program for Industry Specialists' of Korean Ministry of Trade, Industry and Energy (MOTIE), operated by the Korea Institute for Advancement of Technology (KIAT), HRD Program for Future Car, under Grant N0002428.

ABSTRACT LiDAR-based Multiple Object Detection and Tracking (MODT) is one of the essential tasks in autonomous driving. Since MODT is directly related to the safety of an autonomous vehicle, it is critical to provide reliable information about the surrounding objects. For that reason, we propose a semantic point cloud-based adaptive MODT system for autonomous driving. Semantic point clouds emerge with advances in deep learning-based Point Cloud Semantic Segmentation (PCSS), which assigns semantic information to each point in the point cloud of LiDAR. This semantic information provides several advantages to the MODT system. First, any point corresponding to any static object can be filtered. Because the class information assigned to each point can be directly utilized, filtering is possible without any modeling. Second, the class information of an object can be inferred without any special classification process because the class information is provided from the semantic point cloud. Finally, the clustering and tracking module can consider unique dimensional and dynamic characteristics based on class information. We utilize the Carla simulator and KITTI dataset to verify our method by comparing several existing algorithms. In conclusion, the accuracy of the proposed algorithm is improved by about 176% on average compared to the existing algorithm.

INDEX TERMS Semantic point cloud, point cloud semantic segmentation, multiple object detection and tracking (MODT), class-adaptive tracking, autonomous vehicle, LiDAR.

I. INTRODUCTION

The Point Cloud Semantic Segmentation (PCSS) research field is growing rapidly with the development of deep learning technology. The PCSS network can provide a Semantic Point Cloud (SPC), assigned semantic information to the point cloud obtained from the LiDAR (Light Detection And Ranging) sensor. Semantic information includes classification information of points needed in autonomous driving, such as a vehicle, pedestrian, cyclist, building, or terrain (Fig. 1). By providing three-dimensional and semantic information at once, the semantic point cloud helps understand the whole scene around the autonomous vehicle.

PCSS can be divided into offline PCSS and online PCSS according to inference time. Offline PCSS has higher accuracy than online PCSS, but taking a long inference time is the disadvantage. For that reason, offline PCSS is mainly used in applications that do not require real-time characteristics, such as map construction. Online PCSS performs semantic segmentation of the point cloud at a real-time level. Due to real-time capability, online PCSS can be applied to various autonomous driving tasks such as object detection, SLAM, and trajectory planning. According to the semantic KITTI benchmark, one of the popular benchmarks for PCSS, the 2D CNN-based online network with the highest performance has mIoU of 59.9 %, as shown in Table.1. The table shows that progress is being made actively enough to produce reliable performance within a short period. Due to this real-time capability and reliable performance characteristics, PCSS can be applied to many applications required to perceive the surrounding environments, such as object detection, recognition, and localization. This paper will focus on applying online

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

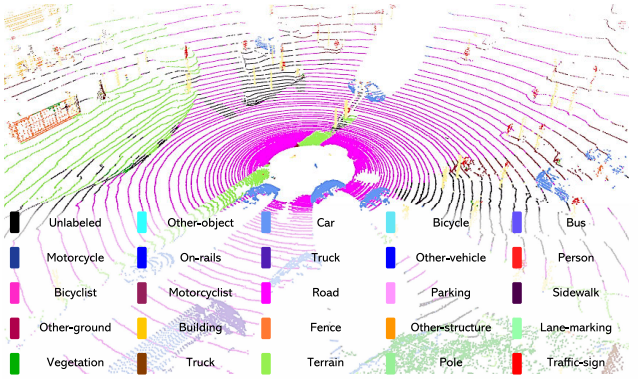


FIGURE 1. Semantic point cloud represented as RGB data.

PCSS to Multiple Object Detection and Tracking (MODT) for improving existing LiDAR-based MODT methods.

Multiple Object Detection and Tracking (MODT) is detecting and tracking multiple objects around the ego-vehicle. Through MODT, the state of objects, such as position, attitude, and shape, can be estimated. The estimated object state is directly associated with autonomous driving safety since this information helps predict the future trajectory and generate collision-free maneuvers. Various perception sensors are used for MODT, such as cameras, LiDAR, and radar. Among them, LiDAR provides accurate 3D depth and geometric information in the form of point cloud data. Since LiDAR calculates range information using Time of Flight (ToF), measured with the time for the reflected light to return to the receiver, it is robust to illumination change. Due to these advantages, LiDAR is emerging as an essential sensor for MODT.

There are three steps for performing LiDAR-based MODT: pre-processing, clustering, and tracking. At the first pre-processing step, raw point cloud data is refined with various pre-processing algorithms, such as the ROI extraction filter and ground removal filter. ROI extraction filter is for extracting point clouds which are included in the region of interest. A ground removal filter is used to remove the ground's point clouds since the ground does not need to be detected in most cases. Through these types of filters, computing resource loss is reduced by eliminating unnecessary point clouds. However, since they use only geometric shape information about the target object, point clouds not included in predefined geometric shapes are difficult to filter out. This problem can be solved if semantic information of each point, such as road or drivable road, is enabled.

The second step is clustering, which is objectifying point cloud data to provide detection information. Detection information includes location and dimension information of surrounding objects. For clustering point clouds, there are various algorithms such as k-mean clustering and DBSCAN (Density-based Spatial Clustering of Application with Noise). K-mean clustering algorithm requires the preset number of clusters; on the other hand, the DBSCAN algorithm does not require a preset number of clusters. For that reason, the DBSCAN algorithm is more suitable to apply in the dynamic environment of autonomous driving. The DBSCAN defines a point cloud as a cluster if the number of points within a specific radius is greater than the minimum number of points. Therefore, the performance of the DBSCAN depends on predefined parameters such as radius and the minimum number of points. However, clustering performance deteriorates when several objects with various classes are close together or overlap because suitable parameters are different according to dimension characteristics for each class. This problem can be overcome using semantic information from the semantic point cloud.

The last step is the tracking step, which estimates the dynamic state of objects based on the clustering information. The tracking step can be subdivided into three steps: data association, track management, and dynamic filtering. In the data association step, a track is associated with measurement for updating the track state. There are popular algorithms such as GNN (Global Nearest Neighbor) and JPDA (Joint Probability Data Association). GNN algorithm associates track with the nearest measurement. The JPDA algorithm is a statistical approach that is more robust to clutter than the GNN algorithm since it is based on expected value. However, since both algorithms depend on distance or uncertainty information except for class information, wrong associations with other classifications can occur. Wrong association problems can be overcome with class information in the semantic point cloud.

The track management step manages tracks with initializing, deleting, and updating. If a measurement is not associated with tracks in the data association step, a new track is created with initializing using this measurement information. On the other hand, if a track is not associated with measurement more than a few times, this track is deleted. Associated measurement with the track in the data association step is used for updating the state of track at the dynamic filtering step. In the dynamic filtering step, many kinds of dynamic filters, such as Kalman Filter (KF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Interacting Multiple Model (IMM) filter, are utilized for updating the state of the track. Among them, the IMM filter can consider multiple models at once based on probability and provide better tracking performance in model transition. These dynamic filters have the two-step: prediction and measurement update step. In the measurement update step, the state is updated with measurement data associated with the data association step. The prediction step predicts the state based on a motion model, which can be modeled with a mathematical equation. Existing algorithms depend on a preset motion model, which does not consider the unique dynamic characteristics of surrounding objects. However, the class-adaptive motion model can be adopted during the dynamic filtering step if there is class information.

LiDAR-based MODT can benefit from using semantic information of semantic point clouds. First, in pre-processing step, needed points for MODT are extracted directly using

semantic information assigned to each point. Semantic information-based pre-processing makes it possible to extract points that cannot be modeled with geometric equations. Second, class-adaptive clustering can be achieved. When performing clustering modules using semantic information in the case that multi-class objects are close together, they can be clearly classified by class and clustered. In addition, a parameter adaptive for the object mutual distance characteristic can be applied based on the class information. Finally, class-adaptive tracking can also be achieved using semantic point cloud. Using not only distance and uncertainty information but also class information during data association can be prevented from being related to other class. Also, a class-adaptive model can be set in consideration of the unique dynamic characteristics of the object through class information.

This paper proposes a novel method about semantic point cloud-based adaptive MODT for an autonomous vehicle. The overall system consists of two parts. The first part is semantic information-based filtering. This step transforms a raw point cloud to a semantic point cloud through an online PCSS network. Using converted semantic point clouds, points about on-road objects are divided into three semantic groups: pedestrian, cyclist, and car. The second part is class-adaptive MODT. Class-adaptive clustering provides 3D object detection (position and dimension of the object) for each semantic group by considering unique object geometric shape information. Using this detection information, class-adaptive tracking provides the track information (position and velocity) with considering unique object dynamic characteristics. To evaluate the proposed algorithm, the performance of the existing LiDAR-Based MODT and our proposed algorithm are compared using Carla simulation and the KITTI dataset.

The main contribution of our paper is a proposition of a new framework that applies semantic point cloud to LiDAR-based MODT. The main summary of contributions are as follows:

- Semantic point cloud-based pre-processing becomes simplified. Points can be processed using only semantic information of semantic point clouds with no need for any geometric modeling.
- Class-adaptive clustering can be achieved. Nearby multi-class objects can be clustered clearly since points have semantic information. Moreover, class-adaptive clustering parameters improve clustering performance by considering unique dimension characteristics.
- Class-adaptive tracking can be achieved. The class-adaptive motion model, which is considered unique dynamic characteristics of each object, can be applied to estimate the state of the object.

The rest of this paper is organized as follows. In section II, previous studies are introduced. Next, the system architecture is introduced in section III. In section IV and V, the proposed algorithm is explained in detail. Section VI describes the evaluation results, and we conclude the paper in the final section VII.

II. PREVIOUS STUDIES

A. POINT CLOUD SEMANTIC SEGMENTATION

Point Cloud Semantic Segmentation (PCSS) can be divided into offline PCSS and online PCSS according to inference time. In this paper, we utilize an online PCSS network to apply to an online MODT. There is various online PCSS framework, which can be subdivided into four categories in terms of input point cloud representation. **3D CNN-based model** uses 3D point cloud as input, which is not converted into other formations [1], [2]. **Graph-based model** used graph structures for representing point cloud. In 3D CNN and graph-based models, the overall performance is relatively higher than other types of models, but because of its long inference time, it is not often used as an online PCSS network [3]. **Point-wise MLP-based model** utilizes multi-layer perceptron (MLP) for extracting features. Especially, RandLa-Net, which has the fast performance to inference with light network architecture, is used for both online PCSS and offline PCSS [4]. **2D CNN-based model** projects point cloud into a 2D domain, providing fast inference time. SqueezeSegV2 and RangeNet++ project point cloud into a spherical coordinate for applying the 2D CNN model [5], [9]. PolarNet uses polar grid data representation about point cloud [6]. 3D-MiniNet learns a 2D representation from the 3D point cloud [7]. SalsaNext utilizes a context module as an encoder which replaces the ResNet encoder blocks and a pixel-shuffle layer as a decoder. This model has the highest performance among 2D projection-based methods, which also ensure real-time level inference time [8]. In this paper, RangeNet++ and SalsaNext are utilized for the online PCSS model since both models have reliable accuracy and fast inference time performance.

B. MULTIPLE OBJECT DETECTION AND TRACKING

1) OBJECT DETECTION

There are many algorithms for object detection, especially clustering. The clustering algorithm can be divided into four parts; partitioning-based, model-based, density-based, and ML-based. **Partitioning-based method** such as k-mean clustering algorithm constructs k partitions and then evaluates them by minimizing error. Since the number of clusters must be specified in advance, this method is not suitable for use in dynamic situations such as autonomous driving. **Model-based method** is based on a probabilistic model with considering uncertainty. However, there are disadvantages, such as computational complexity and clustering depending on the model. **Density-based method** clusters areas of higher density than the remainder of the data. There is a popular density-based algorithm, DBSCAN (Density-Based Spatial Clustering of Applications with Noise). This algorithm cluster points with radius and the minimum number of points parameter. If the number of points in a radius is more than the minimum number of points, points in a radius are considered as the same cluster. This algorithm can be used easily with the point cloud and provide a polygonal cluster that preserves the

TABLE 1. Accuracy and Frame Per Second (FPS) of online point cloud semantic segmentation networks([4]–[9]), which are based on 2D CNN network in the benchmark of Semantic KITTI.

Method	mIoU	Car	Bicycle	Motorcycle	Trunk	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	FPS (Hz)
SqueezeSegV2	39.7	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	**50**
RangeNet53++KNN	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	**91.8**	**65.0**	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.0	12
RandLa-Net	53.9	**94.2**	26.0	25.8	**40.1**	**38.9**	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7	22
PolarNet	54.3	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	**84.0**	**65.5**	**67.8**	51.8	57.5	16
3D-MiniNet-KNN	55.8	90.5	42.3	**42.1**	28.5	29.4	47.8	44.1	14.5	91.6	64.2	74.5	25.4	89.4	60.8	82.8	60.8	66.7	48.0	56.6	28
SalsaNext	**59.9**	91.9	**48.3**	38.6	38.9	31.9	**60.2**	**59.0**	**19.4**	91.7	63.7	**75.8**	**29.1**	**90.2**	**64.2**	81.8	63.6	66.5	**54.3**	**62.1**	24

shape of an object. However, since the same parameters are applied to the entire point cloud, there is a disadvantage that the size characteristics of objects with various classifications cannot be considered individually. Also, there is a disadvantage in that it is difficult to obtain object classification information from the clustering result. **ML-based method** utilizes state-of-the-art machine learning technology [10]–[12]. Most ML-based approaches provide not only position and shape information but also class information about 3D bounding boxes. However, most 3D object detection datasets provide label information in the form of a 3D box, which loses the detailed shape of the object when approximated to the 3D box format. In addition, it is hard to consider static objects around the ego-vehicle since most datasets do not provide the label information of static objects. For that reason, our algorithm is built upon the base DBSCAN algorithm and utilizes semantic point cloud for maintaining the details and considering the overall environment.

2) OBJECT TRACKING

Tracking algorithms can be divided into end-to-end ML-based approaches and dynamic filtering-based approaches. **End-to-end ML-based approaches** are developing fast and have reliable performance; however, they still have limitations [13]–[16]. They require a dataset for the training model and time to training them. If there is a non-labeled object, it is difficult to detect or track them. In addition, the physical properties of an object are hard to consider directly with this type of method. For that reason, **dynamic filtering-based tracking** that can directly consider the physical movement characteristics is still actively used in various situations. Kalman filter-based tracking algorithms can provide an optimal solution when the motion model is modeled to a linear function [17]–[19]. Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) can estimate nonlinear motion models; however, multiple motion models are hard to consider [20]–[22]. In contrast to the methods above, the Interacting Multiple Model (IMM) filter-based tracking method can consider multiple motion models simultaneously in the Bayesian framework [23], [24]. Most dynamic filtering-based approaches have an absence of class information about objects. This makes the tracking problem more challenging due to the difficulty of considering the dynamic characteristics of objects that are adaptive to class information. However, direct semantic information from the semantic point cloud can provide some solutions to this type of method.

III. SYSTEM ARCHITECTURE

The overall system is composed of two parts, as shown in Fig. 2. The first step is semantic information-based pre-processing. This step can be subdivided into online PCSS network (a) and Semantic information-based pre-processing (b). Using the online PCSS network (a), we can convert raw point cloud input into a semantic point cloud. RangeNet++ and SalsaNext are applied to the online PCSS, which have a good balance of accuracy and real-time characteristics. Through the semantic information-based pre-processing (b), on-road objects points, including pedestrians, cars, and cyclists, are extracted.

The second step is class-adaptive MODT (Multiple Object Detection and Tracking). This step can be subdivided into two parts, class-adaptive clustering (c) and class-adaptive tracking (d). As can be seen in Fig. 2(c), independent class-adaptive clustering modules are executed with each of the three-point clouds as input: pedestrian SPC, cyclist SPC, and car SPC. Each class-adaptive clustering module comprises a DBSCAN (Density-Based Spatial Clustering of Application with Noise) algorithm. The search radius and the minimum number of points parameters in the DBSCAN algorithm are adapted according to the classification information of the input semantic point clouds. After that, clusters for each class can be obtained as output: pedestrian clusters, cyclist clusters, and car clusters. Using clusters for each class, independent class-adaptive tracking modules are performed (Fig. 2(d)). The class adaptive tracking module is based on the IMM-UKF-JPDAF tracker, which considers the unique dynamic characteristics of objects using semantic information. We can finally obtain class-adaptive MODT results in the form of bounding boxes for each classification, including track ID, position, velocity, heading, and dimension.

IV. ONLINE POINT CLOUD SEMANTIC SEGMENTATION

A. ONLINE POINT CLOUD SEMANTIC SEGMENTATION

Point Cloud Semantic Segmentation (PCSS) assigns semantic information to point cloud point-wisely through a deep learning network. PCSS Network for on-road MODT must have

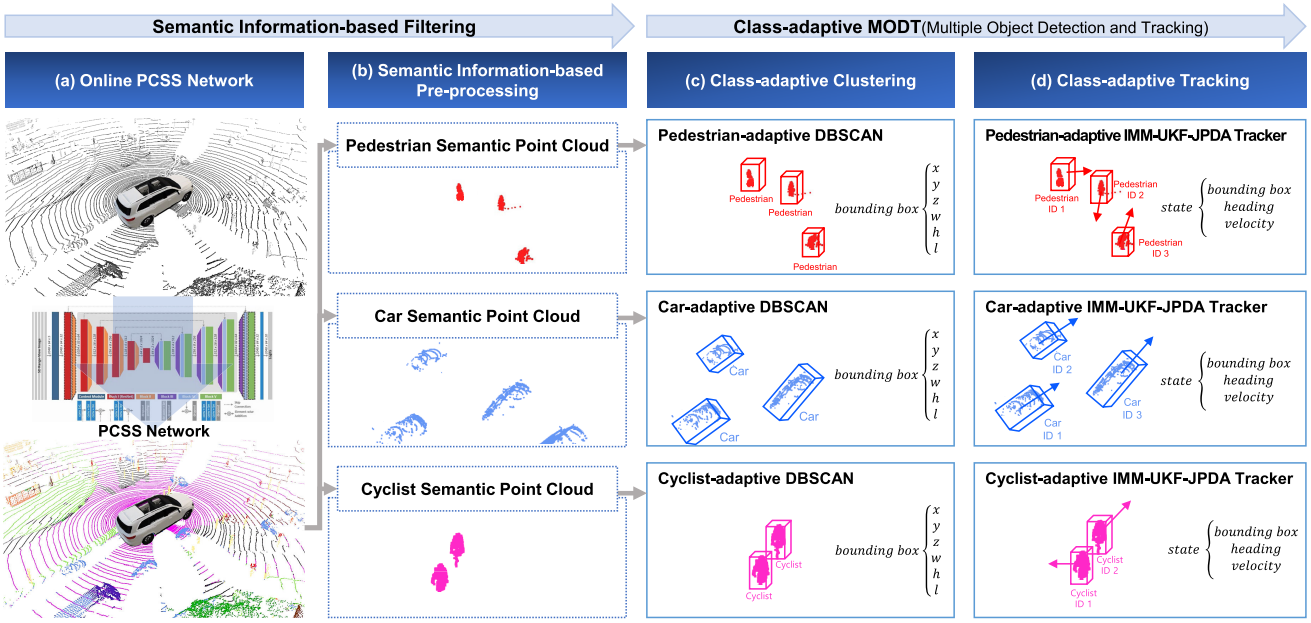


FIGURE 2. The proposed system architecture. The main processes are divided into four parts: (a) online PCSS network, (b) semantic information-based pre-processing, (c) class-adaptive clustering and (d) class-adaptive tracking.

the following conditions. First, the real-time capability must be guaranteed. Second, segmentation performance for an on-road objects must be reliable. In this paper, RangeNet++ and SalsaNext are adopted as PCSS networks that satisfy the above conditions.

RangeNet++ (Fig. 3) projects point cloud into 2D spherical coordinate and apply it to 2D CNN module [9]. Segmented results are reconstructed into a 3D point cloud, and KNN (K-nearest neighbor) search is used as post-processing. With these processes, RangeNet++ achieves 52.2% for mIoU and 12 fps for inference time as shown in Table.1. RangeNet++ has reliable performance for cars and road especially, however, low performance for pedestrians and motorcyclists.

SalsaNext (Fig. 4) is developed with the base of SalsaNet. (Fig. 4) SalsaNext utilized a contextual module for gathering global context information by larger receptive fields. Pixel-shuffle layer is applied for improving computational efficiency, and central encoder-decoder dropout enables higher network performance. Due to these advantages, SalsaNext has the highest performance among 2D projection-based methods, as shown in Table.1. Additionally, SalsaNext has reliable segmentation performance for pedestrians and cyclists as well as a car.

Consequently, in this step, the point cloud is converted into a semantic point cloud by utilizing the above online PCSS network. Semantic point cloud includes labels of various static objects and dynamic objects in the form of RGB information. For example, in Fig. 1, the car is represented as blue, the pedestrian is red, the motorcycle is deep blue, and the motorcyclist is claret color. Ultimately, this semantic information makes it possible to provide abundant environmental information to MODT for surrounding objects.

B. SEMANTIC INFORMATION-BASED PRE-PROCESSING

The pre-processing step is the essential process of extracting point clouds that need to be MODT. If there is no pre-processing step, unnecessary computation is conducted with decreasing computation resource and system accuracy. In this step, the semantic point cloud resulted from PCSS is pre-processed for extracting on-road object points. Since the semantic point cloud contains semantic information in RGB, points about the on-road object can be extracted using RGB information. In this paper, we consider the three On-road objects: cars, cyclists (including motorcycles and motorcyclists), and pedestrians. Extracted point clouds are divided and formed independent point clouds by class. As a result, three groups of point clouds corresponding to cars, cyclists, and pedestrians were obtained.

There are advantages to semantic information-based pre-processing compared with the existing algorithm. For example, small sculptures and children can be treated as the same objects if only dimension characteristics are considered. However, since small sculptures are static object, it does not need to be tracked. Instead, it damages computing resources and becomes a factor that lowers the tracking accuracy when it is close to children. Semantic information becomes a key to solve the above problems. Initially, semantic information-based pre-processing can extract points of any class that cannot be modeled with geometric equations. Secondly, the overall computing resource of the system

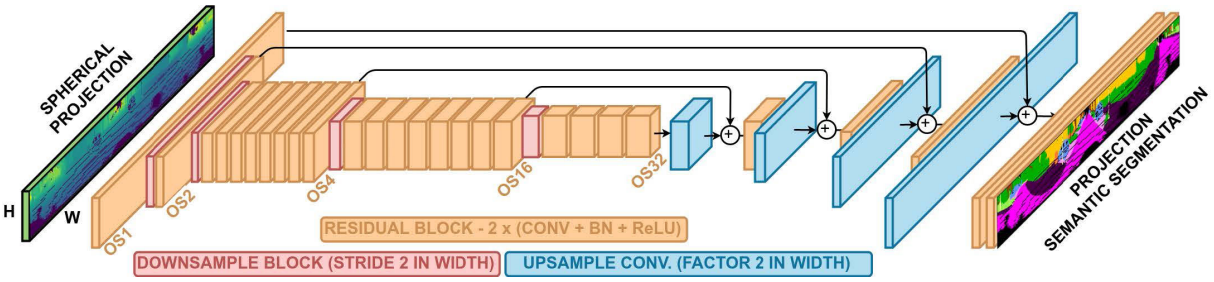


FIGURE 3. System architecture of RangeNet++ [9].

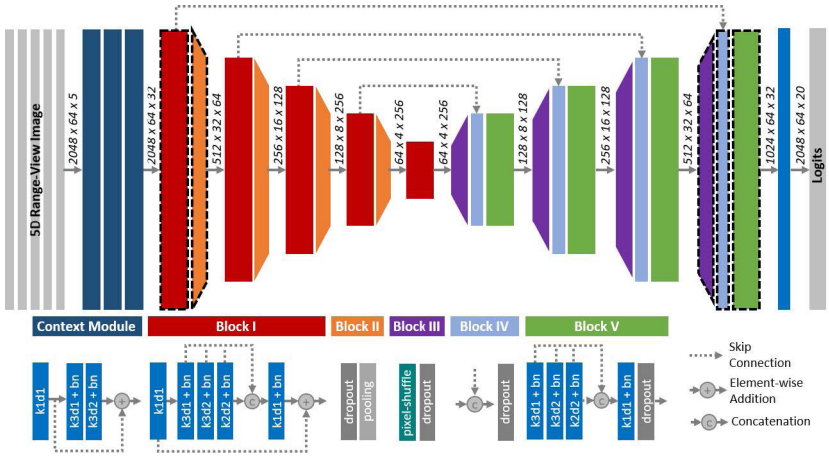


FIGURE 4. System architecture of SalsaNext [8].

can also be reduced by extracting only the points necessary for MODT.

V. CLASS-ADAPTIVE MULTIPLE OBJECT DETECTION AND TRACKING

A. CLASS-ADAPTIVE CLUSTERING

The class-adaptive clustering step consists of independent class-based clustering modules for cars, cyclists, and pedestrians as shown in Fig. 2(c). Each class-adaptive clustering module is based on the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. DBSCAN Algorithm is a density-based clustering algorithm, which clusters the high-density parts (Fig. 5(a)). DBSCAN need to be tuned with parameter *minPts* and ϵ representatively. *MinPts* means the minimum number of points, and ϵ means radius from the core point. If there are more than *minPts* of points within the radius ϵ from the core point, it is recognized as a cluster. As for *minPts*, it is related to the noise characteristics of LiDAR. For instance, the more noisy point cloud is provided from LiDAR, the larger *minPts* must be applied. Similarly, if the resolution performance is high, the larger *minPts* value is set. ϵ value needs to be set considering mutual distance properties of objects. According to the class of objects, objects have different minimum mutual distances. For example, large objects generally have large mutual distances. However, considering class-based characteristics is challenging since the raw point cloud cannot provide class information of each point.

In the previous step, the car, cyclist, and pedestrian point cloud can be obtained. Using these class-independent point clouds, class-adaptive clustering can be conducted by utilizing class information (Fig. 5(b)). The class-adaptive parameter can be constructed based on the class information. There is two-component to be considered when constructing DBSCAN parameters: mutual distance (distance between objects) according to dimension characteristic and LiDAR resolution characteristic. *MinPts* is set to be the same since this value is related to noise characteristics of LiDAR. However, ϵ must be tuned by considering the mutual distance and LiDAR resolution characteristic. ϵ has to be bigger than horizontal and vertical resolution value and smaller than minimum mutual distance. In the case of a car, the minimum mutual distance between objects is assumed when stopping in front of a traffic light. For that reason, $minPts_{car}$ is set at 1.0m, considering the average mutual distance of the car in the above case. Similarly, cyclist also uses mutual distance when stopping in front of a traffic light: $minPts_{cyc}$ is set 0.7m. In pedestrian cases, the stride length of the person is used as the criterion. Assuming the average pedestrian height is 160cm, $minPts_{ped}$ can be set to 0.6m. Finally, clustering results for cars, pedestrians, and cyclists can be obtained with this class-adaptive parameter.

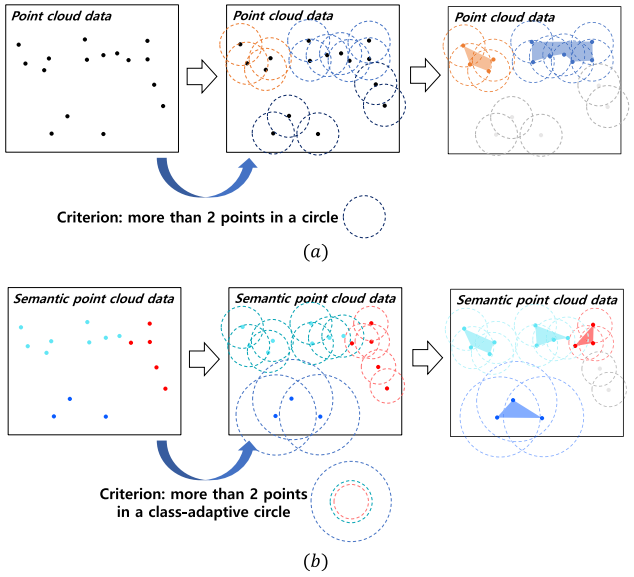


FIGURE 5. (a) Principle of Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The same size of the circle(ϵ) is applied to the whole point cloud. (b) Principle of Class-Adaptive Density-Based Spatial Clustering of Applications with Noise (Class-Adaptive DBSCAN). The semantic point cloud represents class information into RGB data. Different circle size means adaptive ϵ parameter.

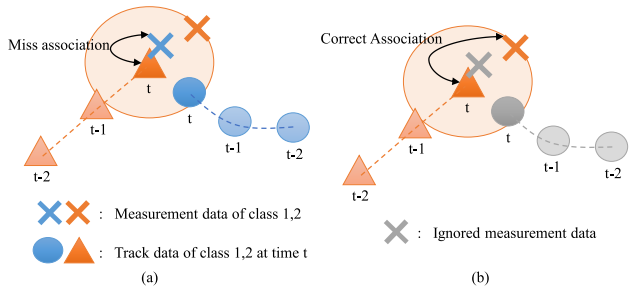


FIGURE 6. (a) Miss data association case. If there is no consideration of class information about measurement, miss association, which associates measurement with other classes, can be occurred. (b) Correct association case. Class information of measurement helps to associate with correct measurement data.

There are several advantages to class-adaptive DBSCAN-based clustering. First, since each class receives an independent class point cloud as an input, there is no influence between points with different classifications. Therefore, there is no problem that different class objects are treated as the same object when they are close or overlapped. Second, clustering performance can be improved by applying parameters suitable for object mutual distance characteristics based on class information.

B. CLASS-ADAPTIVE TRACKING

Class-adaptive tracking is conducted with the results of the class-adaptive clustering, which contains the position and dimension information of the objects. This step consists of three independent modules: car-adaptive tracking module, pedestrian-adaptive tracking module, and cyclist-adaptive

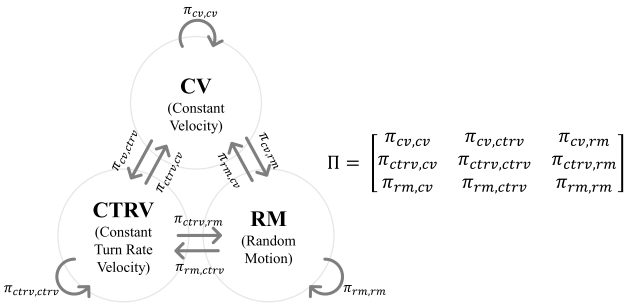


FIGURE 7. Transition probability between multiple motion models can be modeled into a transition matrix. Row 1, column 2 means the probability of conversion from the CV model to the CTRV model.

module as shown in Fig. 2(d). Each module is based on the Interactive Multiple Model-Unscented Kalman Filter-Joint Probabilistic Data Association Filter (IMM-UKF-JPDAF) tracking algorithm, which combines the IMM-UKF dynamic filter and JPDA data association algorithm. The tracks are managed with initialized, updated, or deleted through track management.

1) DATA ASSOCIATION STEP WITH JPDA ALGORITHM

A track is associated with measurement, updating the track state at the data association step. In this step, Joint Probabilistic Data Association (JPDA) algorithm is utilized as a data association algorithm. The JPDA algorithm calculates marginal probabilities for track updates by enumerating all possible joint events. Since the JPDA algorithm considers all feasible joint events, it provides reliable performance even in the presence of clutter like the complex urban environment. However, conventional data association algorithms, including JPDA are hard to consider the class information of the measurement due to the absence of class information. As shown in Fig. 6(a), the absence of class information causes problems that can be related to measurements with other class information. Whereas, the above problem can be solved because the measurement information obtained based on the semantic point cloud is divided by class and goes through the data association step independently as shown in Fig. 6(b).

2) IMM-UKF DYNAMIC FILTERING

In IMM-UKF dynamic filter, the IMM filter is used to estimate the object's accurate and stable state by selecting a model suitable for the movement of an object among various motion models. Each model consists of independent filters, and three motion models are selected based on the UKF to consider even the nonlinear motion model: constant velocity, constant turn rate, and random motion models. State equation and measurement equation for model j(m_j) at sampling time k are as follow:

$$\begin{aligned} x_{k+1} &= f_j(x_k, u_k) + w_{j,k} \\ z_k &= h_j(x_k, u_k) + v_{j,k}. \end{aligned} \tag{1}$$

where f_j represents the system function about target motion m_j to estimate, and h_j represents the measurement function.

u_k means input vector, and z_k is measurement vector. $w_{j,k}$ and $v_{j,k}$ denote process noise and observation noise respectively, assumed to be zero-mean white Gaussian state. Each noise are with covariance matrices Q and R.

IMM filter integrates the estimated results of multiple UKF dynamic filters about the system model set $M = (m_1, m_2, \ldots, m_n)$. The transition among multiple models in the IMM algorithm is controlled by utilizing a time-invariant transition matrix, which conforms to the Markov chain:

$$\Pi = \begin{bmatrix} \pi_{11} & \cdots & \pi_{ni} \\ \vdots & \ddots & \vdots \\ \pi_{1n} & \cdots & \pi_{nn} \end{bmatrix} \tag{2}$$

where matrix component π_{ji} means mode transition probability from model j to i:

$$\pi_{ji} = P(m_i(k)|m_j(k-1)), m_i, \quad m_j \in M \tag{3}$$

using the above transition probabilities, mixing probabilities $\mu_k^{j|i}$ for each model are calculated as:

$$\mu_{k-1|k-1}^{ji} = \frac{\pi_{ji}\mu_{k-1}^{j}}{\sum_{l=0}^{n} \pi_{li}\mu_{k-1}^{l}} \tag{4}$$

Applying mixing probabilities, mixed estimates $\{\hat{x}_{k-1|k-1}^{0i}\}_{i=1}^{n}$ and covariances $\{\Sigma_{k-1|k-1}^{0i}\}_{i=1}^{n}$ can be calculated as:

$$\hat{x}_{k-1|k-1}^{0i} = \sum_{j=1}^{n} \mu_{k-1|k-1}^{ji} \hat{x}_{k-1|k-1}^{j} \tag{5}$$

$$\Sigma_{k-1|k-1}^{0i} = \sum_{j=1}^{n} \mu_{k-1|k-1}^{ji} [\Sigma_{k-1|k-1}^{j} + AA^T] \tag{6}$$

where A represents $\hat{x}_{k-1|k-1}^{j} - \hat{x}_{k-1|k-1}^{0i}$.

The unscented Kalman filter provides to predict and update the estimates and covariances for ith model with a non-linear stochastic model. Consequently, updated estimates $\hat{x}_{k|k}^{i}$ and covariances $\Sigma_{k|k}^{i}$ can be obtained. Finally, the overall estimate and covariance are given in the following equation:

$$\hat{x}_{k|k} = \sum_{i=1}^{n} \mu_k^i \hat{x}_{k|k}^i \tag{7}$$

$$\Sigma_{k|k} = \sum_{i=1}^{n} \mu_k^i [\Sigma_{k|k}^i + (\hat{x}_{k|k}^i - \hat{x}_{k|k})(\hat{x}_{k|k}^i - \hat{x}_{k|k})^T] \tag{8}$$

3) CLASS-ADAPTIVE IMM-UKF DYNAMIC FILTERING

Based on the class information of the detected object, three tracking modules are designed to consider the dynamic characteristics of each class. When designing each tracking module, several parameters need to be considered: Q matrix and transition matrix Π. Q matrix is related to the unique dynamic properties of the object, especially acceleration characteristics. For that reason, it must be tuned according to the class information of the object. When the state x_k is,

$$x_k = \begin{bmatrix} x & y & v & \theta & \dot{\theta} \end{bmatrix} \tag{9}$$

where v means velocity and θ is yaw angle. Based on the above state, the process noise covariance matrix Q is expressed as:

$$Q = \begin{bmatrix} a & b & c & 0 & 0 \\ b & d & e & 0 & 0 \\ c & e & f & 0 & 0 \\ 0 & 0 & 0 & g & h \\ 0 & 0 & 0 & h & i \end{bmatrix} \tag{10}$$

Each component of the above matrix is:

$$\begin{aligned} a &= \frac{\Delta t^4 cos^2(\theta)\sigma}{4}, \quad b = \frac{\Delta t^4 cos(\theta) sin(\theta)\sigma}{4} \\ c &= \frac{\Delta t^3 cos(\theta)\sigma}{2}, \quad d = \frac{\Delta t^4 sin^2(\theta)\sigma}{4} \\ e &= \frac{\Delta t^3 sin(\theta)\sigma}{2}, \quad f = \Delta t^2 \sigma \\ g &= \frac{\Delta t^4 \sigma_{\ddot{\theta}}}{4}, \quad h = \frac{\Delta t^3 \sigma}{2}, \quad i = \Delta t^2 \sigma \end{aligned} \tag{11}$$

In this equation, σ can be tuned differently as a class considering the object's unique acceleration characteristics. Generally, the average acceleration of a car in an urban case is between 1 to $1.5 m/s^2$. The average acceleration of cyclist is $1.8 m/s^2$ and pedestrian is up to $0.5 m/s^2$. By applying the acceleration characteristics for each class, the σ parameter in Eq. 10 constituting the Q matrix are set as follows.

$$\begin{aligned} \sigma_{car} &= 1.5 \\ \sigma_{cyclist} &= 1.8 \\ \sigma_{pedestrian} &= 0.5 \end{aligned} \tag{12}$$

The transition matrix needs to be set considering the class information because the dynamic characteristics are different depending on the class type. However, since existing algorithms suffer from obtaining class information of the object, it is challenging to apply a class-adaptive transition matrix. Whereas, since our method can get the class information of the object, the class-adaptive transition matrix can be used based on the class information. Our approach applies three motion models to the IMM-UKF dynamic filter for estimating on-road objects: constant velocity model, constant turn rate velocity model, and random motion model. Based on these models, the transition matrix can be set as shown in Fig. 7. It can be seen that the first row and the second column represent the probability value of the transition from the CV model to the CTRV model. In order to consider the model transition characteristics for each class, the transition matrix for car, cyclist, and pedestrian is set as follows:

$$\Pi_{car} = \begin{bmatrix} 0.90 & 0.10 & 0 \\ 0.20 & 0.80 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{13}$$

$$\Pi_{cyclist} = \begin{bmatrix} 0.77 & 0.23 & 0 \\ 0.35 & 0.65 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$\Pi_{pedestrian} = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \\ 0.33 & 0.33 & 0.33 \end{bmatrix} \quad (15)$$

where Π_{car} and $\Pi_{cyclist}$ need to consider non-holonomic constraints of car and cyclist so that third row and column values are set to zero.

The class-adaptive tracking module offers various advantages in LiDAR-based MODT systems. First, class information provides reliable data association without being disturbed by clusters with different class information. Second, the unique dynamic characteristics of each class can be considered. The dynamic filtering stage has parameters that need to be adjusted to account for different dynamics, these parameters can be modeled using class information. As a result, class adaptive tracking can be performed based on a semantic point cloud.

VI. VERIFICATION

To verify our method, the tracking performance result from MODT is derived using KITTI Dataset and Carla Simulator. Since the KITTI dataset provides non-synthetic data, tracking performance can be verified in real-world scenarios. However, KITTI raw dataset used in this paper does not provide a ground-truth labeled semantic point cloud. Therefore, the segmentation performance of the PCSS network can affect the overall tracking performance. For this reason, the Carla simulator is also used to evaluate our method. The Carla simulator provides a semantic point cloud labeled as ground truth, so performance can be verified without affecting PCSS performance.

This paper validates the class adaptive module by comparing the performance of our method with the base classical algorithm that does not use SPC. In addition, the object tracking algorithm based on the Kalman filter is also compared to show the appropriateness of selecting the basic algorithm for multi-class object tracking. Three algorithms are compared with the proposed method: IMM-UKF-JPDA algorithm, KF-GNN algorithm, and KF-GNN algorithm with SPC. Since two PCSS models are used in the KITTI dataset verification, two comparison groups are generated in the proposed algorithm and the KF-GNN algorithm, respectively, to which SPC is applied. Those algorithms are evaluated using the evaluation tool from the KITTI dataset. This tool is based on the CLEAR MOT metrics, one of the most used tracking metrics. However, since it was developed to evaluate 2D MOT, the 2D IoU in this tool is modified to 3D IoU to consider 3D MOT. Finally, by using the Carla simulator and KITTI dataset, we can achieve verification of four algorithms, including our method.

A. EVALUATION METRIC

To evaluate our method, the CLEAR MOT metrics are adopted. There are two representative metrics: Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). MOTA usually indicates the overall tracking performance, and MOTP shows localization precision. These are computed as:

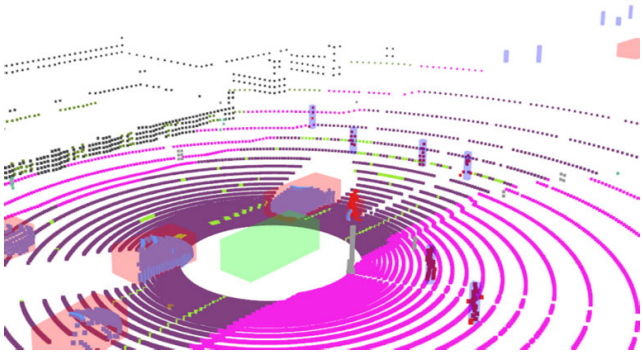
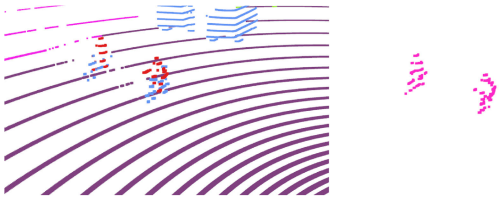


FIGURE 8. Ground-truth labeled semantic point clouds and spawned objects in the Carla simulator. Red boxes are for cars and bicycles, blue boxes are for pedestrians, and green box is for ego vehicle.



(a) (b)

FIGURE 9. (a) Before re-segmentation of cyclists, the cyclist point cloud is divided into pedestrian and car point clouds. (b) After re-segmentation of the cyclist point cloud, it is represented as a claret color.

$$MOTA = 1 - \frac{\sum_t m_t + fp_t + mme_t}{\sum_t g_t} \quad (16)$$

where m_t is the number of misses, fp_t means the number of false positives, and mme_t is the number of mismatches,

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \quad (17)$$

where $\sum_{i,t} d_t^i$ is the total error in estimated position for matched track-measurement pairs and $\sum_t c_t$ means the total number of matches made. This value means the 3D IoU value for the tracked object. In this paper, the overlap between 3D cuboids is measured with a 0.25 IoU threshold. Furthermore, Mostly Tracked object (MT), Mostly Lost object (ML), and a total number of Identity Switches (IDS) are used for evaluating tracking performance. MT and ML are metrics related to tracking quality: how many percent of the track is tracked against the ground truth over the entire track life span. MT threshold is set 80% and ML of 20%. IDS means the number of track IDs changed. This metric shows how consistently the track has been tracking.

B. VERIFICATION WITH CARLA SIMULATOR

1) ENVIRONMENTAL SETUP

Carla simulator provides a synthetic environment for autonomous driving simulation, including various vehicle models, buildings, pedestrians, street signs, etc. Moreover,

TABLE 2. Verification results of a car in the Carla simulator.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	54.33%	60.1%	28.57%	42.86%	15
KF-GNN+*SPC*	59.5%	64.82%	42.86%	42.86%	10
IMM-UKF-JPDA	56.02%	64.91%	35.71%	42.86%	10
Ours	**65.07%**	**67.07%**	**50.0%**	**35.71%**	**2**

TABLE 3. Verification results of a cyclist in the Carla simulator.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	59.71%	34.57%	30%	66.7%	2
KF-GNN+*SPC*	76.26%	41.09%	33.3%	66.7%	1
IMM-UKF-JPDA	81.29%	49.11%	33.3%	66.7%	2
Ours	**94.96%**	**50.94%**	**100%**	**0%**	**1**

a flexible setup of sensor suites and generating similar urban environments are enabled. For verifying our method, the LiDAR sensor is mounted on the ego vehicle's top of the roof and is set to the exact specifications as the Velodyne 32-channel LiDAR. Carla simulation provides a ground truth-labeled semantic point cloud so that it can be utilized to verify this system as shown in Fig.8. Since the cyclist is labeled as a car and pedestrian (Fig.9(a)), the point cloud corresponding to the cyclist is relabeled. As shown in Fig.9(b), the cyclist point cloud can be distinguished by determining whether the car and the pedestrian point cloud exist in proximity. Surrounding objects information obtained from the Carla simulator, including a position in ego-vehicle coordinates, dimensions, class, and unique ID, is used to generate ground-truth data. However, since the vehicle and the cyclist are provided in the same category, the tracklet information is regenerated by additionally utilizing the dimension information of the object. Town 10 map in Carla simulator is utilized, which is the most similar environment to real-world. Surrounding on-road objects such as cars, cyclists, and pedestrian objects are spawned about 100 each. The whole scenario is about 50 seconds long. For verifying tracking performance in this setup, evaluation is conducted targeting objects within 50m around the ego-vehicle.

2) VERIFICATION RESULTS

As shown in Table.2-4, our method offers the best performance for all metrics in all classes. These results are averaged after calculating the results five times for the same scenario. Significantly, there are many improvements compared to the IMM-UKF-JPDA algorithm selected as the basic algorithm, and the effect of the class adaptation module can be shown. MOTA improved by 16.16% in cars, 16.87% in cyclists, and 226% in pedestrians. MOTP improved by 3.33% in cars, 3.73% in cyclists, and 14.95% in pedestrians. Pedestrian shows a more significant improvement in tracking performance. As shown in Fig.11(a), since pedestrians are affected by nearby static objects, the base algorithm shows relatively lower performance. However, as shown in Fig.11(b), by extracting only the target point cloud through semantic filtering, the problem of miss clustering or miss association with a nearby static object is solved. The smallest number of IDS in

TABLE 4. Verification results of a pedestrian in the Carla simulator.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	4.21%	35.13%	0	99.9%	0%
KF-GNN+*SPC*	34.61%	39.60%	0%	28.57%	0
IMM-UKF-JPDA	11.18%	36.11%	0%	85.71%	0
Ours	**36.58%**	**41.51%**	**0%**	**28.57%**	**0**

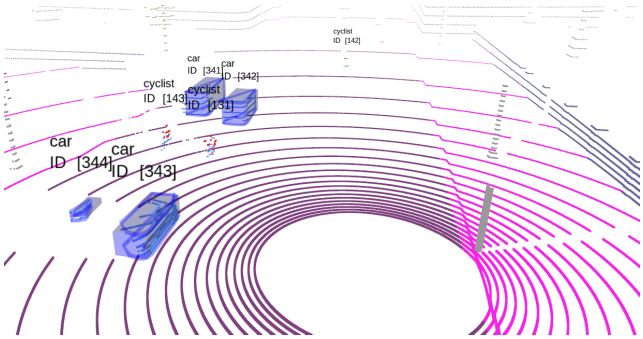


FIGURE 10. Tracking scenario in Carla simulator. The boxes show the results of our tracking system, and the polygon inside the box represents the clustering results. It shows that the tracked objects are well classified and tracked using our method. The label above the tracked object indicates the track's ID and class information.

our method implies a reduction of miss association with different class objects and improvement of tracking performance through class-adaptive modules. KF-GNN comparison also shows performance improvement in most metrics compared with the method not using SPC. Through these results, semantic point cloud-based class-adaptive MODT modules can improve the performance of existing LiDAR-based MODT algorithms.

It shows the objects corresponding to the tracked car and cyclist in Fig.10. In addition, the polygon inside the box means the result of the clustering module. The unique ID and classification information are marked on the labels of the tracked objects. Through this, it can be seen that the proposed method can classify the object and perform the class-adaptive MODT based on the semantic point cloud. Furthermore, it can be seen that tracking of surrounding static objects that do not require tracking is not performed.

C. VERIFICATION WITH KITTI DATASET

1) ENVIRONMENTAL SETUP

The KITTI dataset is utilized to validate our method in the real world. KITTI dataset contains various sensor data like camera, GPS, and IMU, especially including a high-resolution LiDAR Velodyne HDL-64E. However, the KITTI tracking dataset is for 2D bounding boxes containing only the area visible to the camera in front of the ego vehicle. Since the ground truth label is provided as a 2D image plane, it is hard to evaluate tracking performance in a 3D coordinate. For this reason, we utilize a raw dataset from KITTI, which contains 3D bounding box tracking information. Still, a ground truth labeled semantic point clouds are not provided from the KITTI dataset so that we utilize two PCSS

TABLE 5. Verification results of a car in the KITTI raw dataset.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	10.52%	46.45%	2.78%	83.79%	2
KF-GNN + RangeNet++	15.35%	44.03%	0%	66.67%	**1.67**
KF-GNN + SalsaNext	20.84%	31.07%	0%	71.11%	2
IMM-UKF-JPDA	25.04%	45.37%	3.28%	54.86%	12.11
Ours (RangeNet++)	31.06%	**47.62%**	0%	50.18%	17.32
Ours (SalsaNext)	**36.78%**	39.01%	**3.68%**	**40.41%**	5.57

TABLE 6. Verification results of a cyclist in the KITTI raw dataset.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	1.3%	31.68%	0%	100%	0
KF-GNN + RangeNet++	1.65%	**44.38%**	0%	100%	0
KF-GNN + SalsaNext	3.04%	28.12%	0%	100%	0
IMM-UKF-JPDA	2.17%	33.05%	0%	100%	0
Ours (RangeNet++)	8.36%	40.88%	0%	89.06%	4
Ours (SalsaNext)	**23.48%**	33.68%	**0%**	**62.5%**	4

TABLE 7. Verification results of a pedestrian in the KITTI raw dataset.

Method	MOTA↑	MOTP↑	MT↑	ML↓	IDS↓
KF-GNN	5.02%	35.28%	0%	79.52%	4
KF-GNN + RangeNet++	7.72%	**42.47%**	0%	80%	3
KF-GNN + SalsaNext	20.43%	37.29%	6.6%	77.6%	0.6
IMM-UKF-JPDA	6.06%	33.29%	0%	77.91%	2
Ours (RangeNet++)	11.64%	37.74%	0%	98.57%	4
Ours (SalsaNext)	**22.26%**	35.71%	**11.36%**	**68.25%**	**0**

models (RangeNet++, SalsaNext) for verifying tracking performance according to PCSS performance.

2) VERIFICATION RESULTS

Tracking validation is performed using the KITTI raw dataset. However, if the PCSS network cannot segment the point cloud of the target object, evaluation of a specific dataset is not possible. Therefore, the evaluation is performed by excluding datasets that could not be evaluated. The utilized datasets in KITTI raw dataset are 5, 51, 59, 84, and 91 for cars, 14 and 91 for cyclists, and 1, 5, 13, 17, 18, 48, 57, and 59 for pedestrians. The evaluation result for all metrics is the average value of the evaluation values of the datasets used. As shown in Table.5-7, the proposed method is significantly improved compared to the basic algorithm. In particular, the improvement rate of the average MOTA is higher than that of the Carla verification case. Because real-world sensor data is noisier than synthetic data, the base algorithms are vulnerable to ambient static objects or environmental noise. This can be addressed by extracting on-road object data through a semantic point cloud. With these improvements, we found that the semantic point cloud-based class-adaptive MODT system is effective for the existing LiDAR-based MODT.

Two PCSS models are used in this verification, which are RangeNet++ and SalsaNext. As shown in Table.1, SalsaNext has high segmentation performance in car, cyclist and pedestrian. For that reason, our method with SalsaNext also shows higher performance at MOTA, MT, ML, and IDS than the RangeNet++-based method as shown in Table.5-7. In particular, since the segmentation performance of the two networks

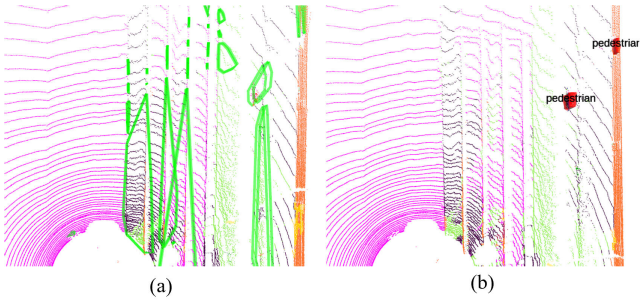


FIGURE 11. (a) DBSCAN clustering results without semantic point cloud applied. The pedestrian on the right side of the figure is incorrectly clustered by a nearby stationary object. (b) Class-adaptive DBSCAN results based on semantic point cloud. Pedestrian are clearly clustered without interference from nearby static obstacles.

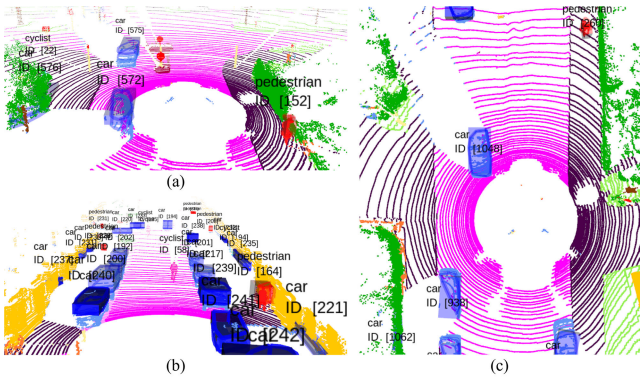


FIGURE 12. (a) and (b) represent the tracking scenarios using KITTI dataset. (c) shows a bird-eye view of the tracking scenario.

for pedestrians and cyclists is significantly different, the difference in tracking performance is also huge. However, MOTP shows lower than the method based on RangeNet++. The segmentation performance of SalsaNext is high, but problems that do not clearly classify even the outer edges of objects can affect tracking precision. Nevertheless, it can be seen that MOTA is significantly improved compared to RangeNet++, indicating that the classification performance of PCSS affects the tracking performance of the proposed algorithm. Our algorithm sometimes tends to have a slightly higher number of IDS or lower MOTP than the base algorithm or KF-GNN algorithm; however, those algorithms have an obviously smaller number of the tracked object. Consequently, it can be verified that the proposed algorithm can improve existing LiDAR-based MODT algorithms.

In addition, the possibility of applying our system to the real-time application is verified by measuring the execution time of the proposed method. The PCSS network SalsaNext and RangeNet++ are 20 fps and ten fps, respectively. Considering these speeds, the overall system speed recorded about 12 fps with SalsaNext and seven with RangeNet++. Since the frame rate of LiDAR is about 10 Hz, it may be challenging to use in real-time applications in the case of a system to which RangeNet++ is applied. However, the SalsaNext-based method can be sufficiently applied to

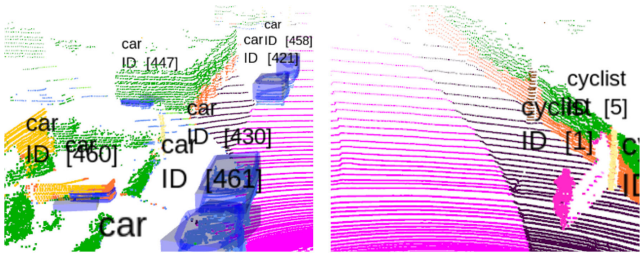


FIGURE 13. Misclassification problems due to segmentation errors in PCSS networks. (a) A problem occurred that the point cloud representing the surrounding static objects is misclassified as a car. (b) On the right side, a point cloud that is about the wall is misclassified as a pedestrian.

real-time applications at ten fps. Accordingly, it is verified that our method based on SalsaNext can be applied to real-time application with reliable performance and speed.

As shown in Fig.12, the proposed method can detect and track objects corresponding to cars, pedestrians, and cyclists. Even object classification information can be obtained, and since only the target object is tracked, static objects are not tracked. In addition, it can be seen that pedestrians are tracked unaffected by nearby static objects. However, there are some false positives, which are not the target objects but are tracked as target objects. As shown in Fig.13, the segmentation error of the PCSS network leads to the creation of false-positive tracks.

VII. CONCLUSION

In this paper, a class-adaptive MODT framework based on semantic point cloud was proposed. First, semantic information-based filtering was introduced to extract the target point cloud. Through this step, it was possible to extract unmodeled objects by utilizing the semantic information assigned to each point. Second, class-adaptive clustering and tracking modules were designed taking into account their unique dimensional and dynamic characteristics. In the class-adaptive clustering module, each class-based module was constructed by considering the mutual distance of each object. The class-adaptive tracking module is designed by setting the appropriate Q matrix and transition matrix in consideration of the dynamic characteristics. Finally, the entire system was validated using the Carla simulator, providing ground-truth labeled SPC and the KITTI dataset for real-world validation. As a result, there was a significant improvement in performance verification, compared with several existing algorithms. Moreover, to analyze the effect of PCSS network performance on this system, a performance comparison was performed according to the PCSS network. To sum up,

- **Semantic information-based filtering**: through semantic point cloud, unmodeled objects' point cloud could be filtered simply. Points corresponding to a specific object could be extracted, which helps improve system accuracy and reduce computational resources in MODT.
- **Semantic point cloud-based class-adaptive MODT**: based on the semantic information, we designed the class-adaptive module for clustering and tracking to consider mutual distance and acceleration characteristics. Consequently, the average accuracy of the validation improved by about 86.34% on the Carla simulator and 267% on the KITTI dataset.

Although the method proposed in this paper has improved performance compared to the existing classical methods, it still could not record very high values in terms of accuracy or precision. In addition, since the PCSS network performance is highly dependent, a study considering the classification uncertainty of the PCSS network will be required. As future work, we plan to apply the state-of-the-art online PCSS network and various MODT algorithms in the proposed system to improve overall performance. Specifically, machine learning-based detection or tracking algorithms will be utilized in future works for improving MODT performance. Additionally, by considering the uncertainty of the PCSS network, we will reduce the dependence of the performance of the PCSS network and develop a more robust MODT algorithm.

REFERENCES

[1] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Comput. Graph.*, vol. 88, pp. 24–34, May 2020.
[2] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Jun. 2019, pp. 6411–6420.
[3] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4558–4567.
[4] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 11108–11117.
[5] 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 *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 4376–4382.
[6] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9601–9610.
[7] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3D-mininet: Learning a 2D representation from point clouds for fast and efficient 3D LiDAR semantic segmentation," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5432–5439, Jul. 2020.
[8] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds," in *Proc. Adv. Vis. Comput.*, 2020, pp. 207–222.
[9] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Nov. 2019, pp. 4213–4220.
[10] J. Lehner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Nessler, and S. Hochreiter, "Patch refinement—Localized 3D object detection," 2019, *arXiv:1910.04093*.
[11] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1711–1719.
[12] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li;, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10529–10538.

[13] S. Wang, Y. Sun, C. Liu, and M. Liu, "PointTrackNet: An end-to-end network for 3-D object detection and tracking from point clouds," *IEEE Robot. Autom.*, vol. 5, no. 2, pp. 3206–3212, Apr. 2020.
[14] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "FANTrack: 3D multi-object tracking with feature association network," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2019, pp. 1426–1433.
[15] V. Vaquero, I. Del Pino, F. Moreno-Noguer, J. Sola, A. Sanfeliu, and J. Andrade-Cetto, "Dual-branch CNNs for vehicle detection and tracking on LiDAR data," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6942–6953, Nov. 2020.
[16] A. Shenoi, M. Patel, J. Gwak, P. Goebel, A. Sadeghian, H. Rezatofighi, R. Martin-Martin, and S. Savarese, "JRMOT: A real-time 3D multi-object tracker and a new large-scale dataset," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10335–10342.
[17] X. Weng, J. Wang, D. Held, and K. Kitani, "3D multi-object tracking: A baseline and new evaluation metrics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2020, pp. 10359–10366.
[18] B. Fortin, R. Lherbier, and J. C. Noyer, "A model-based joint detection and tracking approach for multi-vehicle tracking with lidar sensor," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1883–1895, Apr. 2015.
[19] K. Burnett, S. Samavi, S. Waslander, T. Barfoot, and A. Schoellig, "AUTo-Track: A lightweight object detection and tracking system for the SAE AutoDrive challenge," in *Proc. 16th Conf. Comput. Robot. Vis. (CRV)*, May 2019, pp. 209–216.
[20] T. Miyasaka, Y. Ohama, and Y. Ninomiya, "Ego-motion estimation and moving object tracking using multi-layer LIDAR," in *Proc. IEEE Intell. Veh. Symp. Process.*, Dec. 2009, pp. 151–156.
[21] T. Kim and T. H. Park, "Extended Kalman filter (Ekf) design for vehicle position tracking using reliability function of radar and lidar," *Sensors*, vol. 20, no. 15, pp. 1–18, 2020.
[22] J. Zhang, W. Xiao, B. Coifman, and J. P. Mills, "Vehicle tracking and speed estimation from roadside lidar," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 5597–5608, 2020.
[23] M. Sualeh and G.-W. Kim, "Dynamic multi-LiDAR based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, p. 1474, Mar. 2019, doi: 10.3390/s19061474.
[24] M. Sualeh and G.-W. Kim, "Visual-LiDAR based 3D object detection and tracking for embedded systems," *IEEE Access*, vol. 8, pp. 156285–156298, 2020, doi: 10.1109/ACCESS.2020.3019187.
[25] C. Zhao, C. Fu, J. M. Dolan, and J. Wang, "L-shape fitting-based vehicle pose estimation and tracking using 3D-LiDAR," *IEEE Trans. Intell. Veh.*, early access, May 10, 2021, doi: 10.1109/TIV.2021.3078619.
[26] K. Samal, H. Kumawat, P. Saha, M. Wolf, and S. Mukhopadhyay, "Task-driven RGB-lidar fusion for object tracking in resource-efficient autonomous system," *IEEE Trans. Intell. Veh.*, early access, Jun. 8, 2021, doi: 10.1109/TIV.2021.3087664.

SOYEONG KIM received the B.S. degree in smart vehicle engineering from Konkuk University, Seoul, South Korea, in 2021, where she is currently pursuing the master's degree with the Automotive Intelligence Laboratory. Her main research interests include applications for object tracking, localization, and mapping of intelligence vehicles.

JINSU HA is currently pursuing the B.S. degree with the Automotive Intelligence Laboratory, Konkuk University. His main research interests include applications for vehicle tracking and detection based on deep learning of intelligence vehicles.

KICHUN JO (Member, IEEE) received the B.S. degree in mechanical engineering and the Ph.D. degree in automotive engineering from Hanyang University, Seoul, South Korea, in 2008 and 2014, respectively. From 2014 to 2015, he was with the ACE Laboratory, Department of Automotive Engineering, Hanyang University, doing research on system design and implementation of autonomous cars. From 2015 to 2018, he was with the Valeo Driving Assistance Research, Bobigny, France, working on the highly automated driving. He is currently an Assistant Professor with the Department of Smart Vehicle Engineering, Konkuk University, Seoul. His current research interests include localization and mapping, objects tracking, information fusion, vehicle state estimation, behavior planning, and vehicle motion control for highly automated vehicles.