Received: 21 March 2023 | Revised: 13 July 2023 | Accepted: 9 August 2023

DOI: 10.4218/etrij.2023-0109

SPECIAL ISSUE

ETRI Journal WILEY

EMOS: Enhanced moving object detection and classification via sensor fusion and noise filtering

Dongjin Lee[1,2] | **Seung-Jun Han**[1] | **Kyoung-Wook Min**[1] | **Jungdan Choi**[1] | **Cheong Hee Park**[2]

[1]Autonomous Driving Intelligence Research Section, Mobility Robot Research Division, Superintelligence Creative Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

[2]Department of Computer Science and Engineering, Chungnam National University, Daejeon, Republic of Korea

Correspondence
Cheong Hee Park, Department of Computer Science and Engineering, Chungnam National University, Daejeon, Republic of Korea.
Email: cheonghee@cnu.ac.kr

Funding information
Institute for Information and Communications Technology Promotion, Grant/Award Number: 2021-0-00891; Ministry of Science, ICT and Future Planning, Grant/Award Number: 2020-0-00002

Abstract

Dynamic object detection is essential for ensuring safe and reliable autonomous driving. Recently, light detection and ranging (LiDAR)-based object detection has been introduced and shown excellent performance on various benchmarks. Although LiDAR sensors have excellent accuracy in estimating distance, they lack texture or color information and have a lower resolution than conventional cameras. In addition, performance degradation occurs when a LiDAR-based object detection model is applied to different driving environments or when sensors from different LiDAR manufacturers are utilized owing to the domain gap phenomenon. To address these issues, a sensor-fusion-based object detection and classification method is proposed. The proposed method operates in real time, making it suitable for integration into autonomous vehicles. It performs well on our custom dataset and on publicly available datasets, demonstrating its effectiveness in real-world road environments. In addition, we will make available a novel three-dimensional moving object detection dataset called ETRI 3D MOD.

KEYWORDS
autonomous driving, deep learning, image classification, object detection, sensor fusion

1 | INTRODUCTION

Autonomous vehicles (AVs) can operate without the intervention of human drivers, likely reducing accidents and increasing driver convenience. With the development of advanced sensor technology, machine learning algorithms, and computing power, AVs have become a reality with the potential to revolutionize the transportation industry [1–3]. AV technology can be broadly divided into three stages: perception, planning, and control. In the perception stage, sensors such as light detection and ranging (LiDAR) sensors, cameras, and radars are used to perceive the driving environment.

LiDAR sensors achieve higher accuracy when estimating distance than radars and can be used at night without being affected by lighting conditions, unlike cameras. Since the introduction of LiDAR-based three-dimensional (3D) object detection [4] combined with deep learning, diverse studies have been conducted, and state-of-the-art performance has been achieved on various benchmarks [5–8]. However, the false-positive rate is relatively high in driving environments that are different from those in which the model was trained using a

LiDAR-based object detection model [9, 10]. This is because a LiDAR sensor has a lower spatial resolution than cameras and lacks texture and color information. Therefore, object classification using LiDAR is based solely on geometric information, leading to a higher false-positive rate.

LiDAR–camera fusion can be employed to overcome the limitations of using LiDAR-only approaches. However, existing studies on LiDAR–camera fusion require the synchronized collection of LiDAR and camera data in a specific environment before applying a model trained in a different environment [11–13]. Additionally, retraining the model for application to a new environment involves laborious 3D object labeling, which is time-consuming and expensive. To address these limitations, we introduce a LiDAR–camera fusion method, as depicted in Figure 1. The proposed method consists of three steps: LiDAR-based 3D object detection, camera-based image classification, and prediction refinement. In the first step, we predict the classification information of objects in 3D space and their sizes, positions, headings, and confidence scores using a LiDAR-based 3D object detection model. In the second step, the objects detected in the previous stage are projected onto the corresponding images, and the corresponding regions are cropped. The cropped images are processed by an image classification model. Finally, object classification is refined by fusing the predictions from the two preceding steps, and spurious objects are removed.

In the proposed method, publicly available image data can be used to train the camera-based image classification model without requiring the collection of synchronized LiDAR data for retraining. To improve the image classification performance in a new environment, public data collected in similar environments or newly collected camera data can be used to then perform two-dimensional (2D) bounding box (BB) labeling. More datasets are available for 2D object detection than for 3D object detection, and labeling 2D BBs is generally less time-consuming and costly than labeling 3D objects. Alternatively, we generate noisy data from images and use them for training an image classification model to reduce the false-positive rate.

Although some 3D object detection datasets that include domestic road environments are publicly available at AIHub [14], certain data within these datasets exhibit sensor quality and annotation issues. To address this problem, we propose a new 3D object detection dataset called ETRI 3D moving object detection (MOD) to facilitate research on autonomous driving in South Korea. Moreover, we constructed an ETRI 2D moving object classification (MOC) dataset to evaluate the proposed method.

The main contributions of this study are summarized as follows:

- We introduce a simple yet effective method to refine the object classification results by fusing camera data with LiDAR sensor data to overcome the limitations of LiDAR sensors and mitigate false positives.
- To showcase the applicability of the proposed method, we constructed the ETRI 2D MOC dataset, which consists of real-world road environments, and evaluated its effectiveness. Furthermore, we demonstrated the

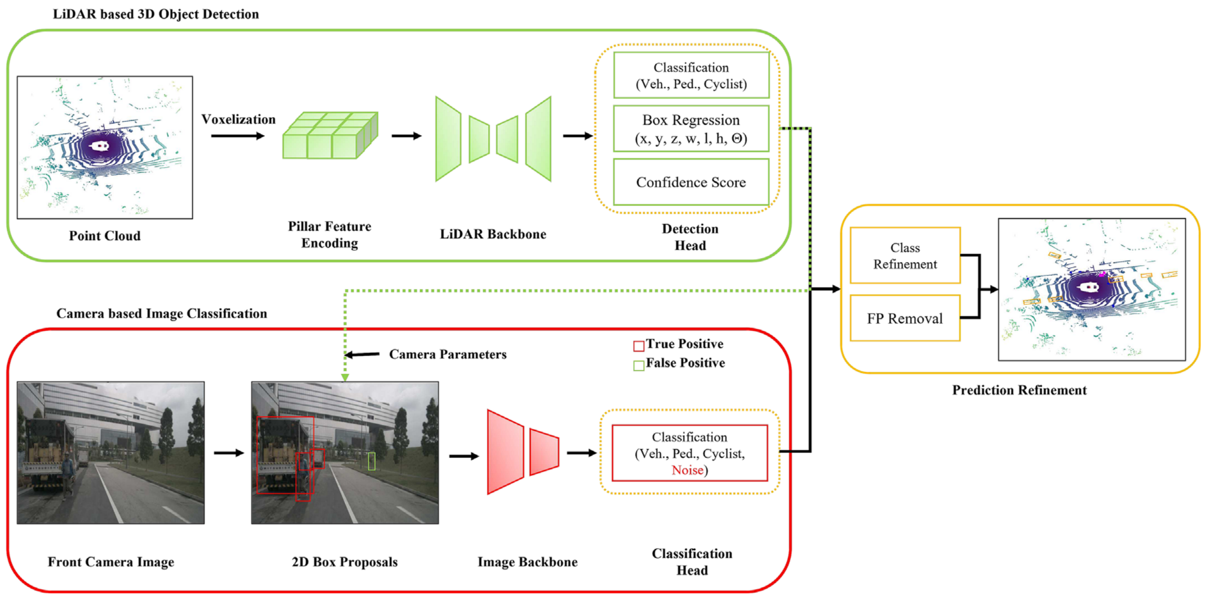


FIGURE 1 Overall architecture of proposed 3D object detection and image classification method.

generalization ability of our method, which achieved excellent results on the KITTI dataset. This dataset was not used for training. Hence, our method has a suitable performance on previously unseen data, further confirming its effectiveness and generalization ability.
- We constructed a new 3D object detection dataset, ETRI 3D MOD, through collection in urban and campus environments located in Daejeon City, Republic of Korea. We will publicly release this dataset.

2 | RELATED WORK

2.1 | LiDAR-based 3D object detection

Object detection in 3D space involves classifying an object and estimating its oriented 3D BBs, which include position, size, and orientation. Deep-learning-based 3D object detection for point-cloud data from LiDAR sensors has emerged as a state-of-the-art method, showing superior results on public datasets [5–8]. VoxelNet [4] was one of the first models to learn feature representations from point-cloud data and estimate 3D BBs in an end-to-end manner. It uses point-cloud voxelization and PointNet [15] to encode the features of each voxel. The input for the convolutional middle layers is a sparse four-dimensional tensor, and 3D convolution and voxel-wise feature aggregation are applied for feature extraction and dimensionality reduction. Finally, a region proposal network accurately predicts the 3D BBs. The inference time is 225 ms, with the convolutional middle layers being the bottleneck that require 170 ms for processing. SECOND [16] refined VoxelNet [4] to reduce the training and inference times while enhancing the performance when predicting the orientation. A new data augmentation technique was also used to further improve convergence speed and detection performance. It requires 50 ms, making it suitable for real-time applications. To improve the runtime performance, PointPillars [17] introduced pillar-based voxelization, which transforms the convolution kernel of the middle layers from three to two dimensions. The runtimes of this method are 24 ms in PyTorch [18] and 16 ms in NVIDIA TensorRT [19]. In [20], a multi-view fusion technique called dynamic voxelization was proposed to extract features from two viewpoints, namely, bird's eye view (BEV) and perspective view. It concatenates features from three different sources and reveals the advantages of dynamic voxelization over hard voxelization in terms of four metrics. CenterPoint [21] is an anchor-free approach that learns a keypoint detector to detect an object center and estimates other attributes of the object, such as 3D shape, orientation, and velocity. In the second stage, the prediction results are refined using map-view features. Moreover, in [21], two different backbones (VoxelNet [4] and PointPillars [17]) were compared regarding detection performance and runtime.

2.2 | Camera-based image classification

Convolutional neural network (CNN)-based image classification technology has been widely studied and used in numerous practical applications since the introduction of AlexNet [22]. A visual geometry group network [23] stacks small (3×3) convolution kernels instead of 5×5 or 7×7 kernels, allowing a deeper network while reducing the number of parameters to be learned. GoogLeNet [24] utilizes deep and complex CNNs that consist of 22 layers with nine inception modules and uses several convolution filters to obtain multiscale features. Two additional auxiliary classifiers weighted by 0.3 are utilized to solve the vanishing gradient problem that usually occurs in deep networks. ResNet [25] is a residual learning method for training CNNs and is considerably deeper than previous methods [23, 24]. The key building block of the network is called the residual block, which contains more than two convolutional layers including shortcut connections. It achieves remarkable results for a wide range of computer vision tasks, such as image classification, object detection, and segmentation, on several benchmark datasets [26–28]. Their design principles have influenced many subsequent studies [29–31]. EfficientNet [32] utilizes a model-scaling method that employs a compound coefficient to balance the components of the model, such as the depth, width, and resolution, for a given resource budget. The effectiveness of this approach is evaluated using well-known CNN architectures [25, 33]. A neural architecture search is used to construct a novel baseline network, which is then expanded to produce a range of models known as EfficientNets.

Transformer-based image classification is a new area in computer vision, and several studies have explored this approach. A vision transformer [34] substitutes certain convolutional layers of CNNs with self-attention layers, thereby allowing the model to merge information from different patches. The model is pretrained on large datasets such as ImageNet, ImageNet-21k, and JFT-300M to learn a general representation of visual features, and it is fine-tuned for several downstream tasks. A swin transformer [35] uses a hierarchical framework with shifted windows for extracting visual features at multiple scales. Its design allows capturing local features and understanding the global context.

2.3 | Multisensor-fusion-based 3D object detection

Multisensor-fusion-based 3D object detection that integrates sensors such as LiDAR sensors, cameras, and radars can be broadly classified into early, intermediate, and late fusion methods [11, 12, 30]. Early fusion combines data from different sensors at the raw data level; intermediate fusion performs merging at the feature level; and late fusion combines the results at the output level. However, a dataset that includes 3D BB annotations is required to train the 3D object detection model using these methods, which can be time-consuming and expensive. To address this issue, a novel approach is proposed in this study to enhance the performance of 3D object detection using publicly available image classification datasets that are relatively inexpensive to annotate and contain numerous samples.

3 | METHOD

We propose a method for fusing point clouds acquired from LiDAR sensors with images acquired from cameras to predict 3D BBs and refine the classification information of the predicted class to mitigate false positives, as shown in Figure 1 [36]. The three steps of (1) LiDAR-based 3D object detection, (2) camera-based image classification, and (3) prediction refinement are detailed.

3.1 | LiDAR-based 3D object detection

To estimate 3D objects from point-cloud data, we employed CenterPoint-Pillars (CPP) [21], which has excellent object detection performance and operates in real time, making it suitable for applications in AVs. This method is divided into three stages: pillar feature encoding, backbone, and detection head.

3.1.1 | Pillar feature encoding

Point-cloud data obtained from LiDAR sensors include the position (x, y, z) and intensity of reflectance (intensity) of each point in 3D space. We did not use intensity data in this study, unlike previous studies [4, 16, 17, 20, 21], because the meaning and range of intensity values can differ between LiDAR manufacturers, even if they have the same values. In this study, we used three different datasets, which also influenced our decision not to use intensity data. To unify the coordinate systems of the datasets, we expressed point-cloud data and annotation information in a vehicle coordinate system. Then, we divided the 3D point-cloud data into 2D pillar shapes by partitioning the data along the x and y directions. Let $P = \{(x_i, y_i, z_i)\}_{i=1}^{N}$ represent the point-cloud data contained within a pillar. x_c, y_c, z_c denote the mean values of the data points within a pillar and can be calculated using the following formulas:

$$x_c = \frac{\sum_{i=1}^{N} x_i}{N},\ y_c = \frac{\sum_{i=1}^{N} y_i}{N},\ z_c = \frac{\sum_{i=1}^{N} z_i}{N}, \tag{1}$$

where x_p, y_p represent the center coordinates of a pillar. Each point cloud was augmented as follows: $P_{\text{aug}} = \{(x_i, y_i, z_i, x_i - x_c, y_i - y_c, z_i - z_c, x_i - x_p, y_i - y_p)\}_{i=1}^{N}$. The augmented feature P_{aug} was used as an input to PointNet, which ultimately generated a 64-dimensional feature vector for each nonempty pillar. The generated features were transformed into a sparse pseudo-image format for use as inputs to a 2D CNN.

3.1.2 | Backbone

We utilized a backbone similar to CPP [21] consisting of top–down and upsampling networks. The top-down network extracts features from the output of pillar feature encoding, whereas the upsampling network generates a dense output tensor by increasing the spatial resolution of the feature maps produced by the top-down network. Further improvements in speed were necessary to integrate this approach into AVs. Therefore, we adopted the method implemented in SECOND [16] and adjusted the stride values of the upsampling network blocks from [1, 2, 4] to [0.5, 1, 2]. After this adjustment, the output feature map had half the width and height of the input, whereas the number of channels increased from 64 to 384.

3.1.3 | Detection head

In the detection head, we predicted class-specific heatmaps, height, 3D size, and rotation, and refined the sub-pillar location. Once class-specific heatmaps were generated, they were passed through a sigmoid function to calculate the probability values for each class, which were expressed as follows: $\text{cls}_{\text{LiDAR}} = \sigma\left(\left\{\text{hm\_cls}^{i}_{\text{LiDAR}}\right\}_{i=1}^{N}\right)$. The confidence score ($CS$) was calculated as follows:

$$\text{CS}_{LiDAR} = \max(\text{cls}_{\text{LiDAR}}). \tag{2}$$

3.2 | Camera-based image classification

Camera calibration parameters were used to project the 3D BBs predicted by the LiDAR-based object detector described in Section 3.1 onto an image. Let $\mathbf{B}^i_{\text{LiDAR}} = [x, y, z, l, w, h, r]$ denote the 3D BB detected by a LiDAR-based object detector for object i. The projection of this BB onto an image can be expressed using the following camera projection equation:

$$[uv1] = \mathbf{P}[x\ y\ z\ 1], \tag{3}$$

where $[uv1]$ is the projected 2D coordinates of the object in the image and $\mathbf{P}$ is the 3×4 camera projection matrix. Only the BBs that are projected within the field of view of the camera are utilized in the subsequent steps.

3.2.1 | Backbone

We used a backbone similar to that of ResNet50 [25]. Compared with the original ResNet50, our modified version, called ResNet50-Tiny, was adjusted as follows:

- The kernel size of layer conv1 was changed from 7 to 3, and its stride was changed from 2 to 1.
- The max pooling operation in layer conv2 was removed.

These modifications enabled the network to operate on smaller input images while reducing the computational cost. Table 1 lists the architectures of ResNet50 and ResNet50-Tiny. The output of the model during inference was a vector of the computed probabilities for each class, obtained by applying the softmax function to the output vector of the model as follows:

$$\text{cls}_{\text{camera}} = \text{softmax}(\mathbf{z}), \tag{4}$$

where $\mathbf{z}$ is the output vector of the model and $\text{cls}_{\text{camera}}$ is a vector of probabilities per class.

TABLE 1 Architectures of ResNet-50 and ResNet-50-Tiny.

Layer	ResNet-50	ResNet-50-Tiny
conv1	7 × 7, 64, stride 2	3 × 3, 64, stride 1
conv2_x	3 × 3 max pool, stride 2	–
	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix} \times 3$	
conv3_x	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix} \times 3$	
conv4_x	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 1024 \end{bmatrix} \times 3$	
conv5_x	$\begin{bmatrix} 1\times1,\ 512 \\ 3\times3,\ 512 \\ 1\times1,\ 2048 \end{bmatrix} \times 3$	
	average pool, 4-d fc, softmax	

3.3 | Prediction refinement

We propose a method to improve object classification performance and remove false positives by fusing the results of the LiDAR-based 3D object detection model (Section 3.1) and the camera-based image classification method (Section 3.2) for detected objects. The equations for combining the results of the two methods are as follows:

$$\begin{aligned} &\text{cls}_{\text{refined}} = \max(\lambda_1 \text{CS}_{\text{LiDAR}} + \lambda_2 \text{cls}_{\text{camera}}), \\ &\lambda_1 + \lambda_2 = 1, \end{aligned} \tag{5}$$

where CS_{LiDAR} and $\text{cls}_{\text{camera}}$ are the results obtained from (2) and (4), respectively. Hyperparameters λ_1 and λ_2 control the weight of each result in the final classification step and correspond to LiDAR-based 3D object detection and camera-based image classification, respectively.

4 | DATASETS

This section describes datasets used for 3D object detection and image classification. To perform 3D object detection, we used three publicly available datasets and the proposed ETRI 3D MOD dataset.

4.1 | Publicly available 3D object detection datasets

4.1.1 | Waymo open dataset (WOD)

WOD [7] is a widely recognized dataset in computer vision with the largest number of samples for 3D object detection. It provides sensor data, including data from one midrange LiDAR sensor, four short-range LiDAR sensors, and five cameras collected from several cities across the United States. The dataset comprised 1000 segments, each with a duration of 20 s and recorded at a frequency of 10 Hz, with a total of 12 million 3D BB labels for four distinct object classes: vehicles, pedestrians, cyclists, and traffic signs.

4.1.2 | PandaSet dataset

PandaSet [37] is a comprehensive dataset that contains data from various sensors employed in autonomous driving and aims to help researchers in related fields, such as 3D object detection and semantic segmentation. The data were collected from various sensors, including a mechanical LiDAR sensor (Hesai Pandar64), forward-facing LiDAR sensor (Hesai PandarGT), six cameras, and Global Navigation Satellite System inertial measurement unit (GNSS-IMU) (NovAtel PwrPak7). The dataset included more than 100 scenes, each of which was 8 s long and recorded at 10 Hz. In addition, the dataset provides labeled data for 3D object detection in 28 different classes.

4.1.3 | KITTI dataset

The KITTI dataset [6] is a collection of data gathered in the Karlsruhe region of Germany using a recording platform equipped with an inertial navigation system, LiDAR sensor, and cameras. The dataset provides a benchmark for evaluating object detection and contains 7481 training samples and 7518 test samples. The dataset primarily enables detection evaluation in three categories: cars, pedestrians, and cyclists. Typically, the training data are divided into two sets with approximately 3712 samples for training and 3769 samples for validation.

4.2 | Proposed ETRI 3D MOD dataset

We introduce a novel 3D object detection dataset called the ETRI 3D MOD, which was collected from urban and campus areas in Daejeon City, Republic of Korea. Three types of AVs were used for the data collection. Each AV was equipped with a 64-channel LiDAR sensor (Hesai Pandar64), high-resolution forward camera (FLIR CM3-U3-31S4C-CS), and GNSS-IMU (Advanced Navigation Certus Evo). The dataset consisted of 42 scenes, each ranging from 2 s to 10 s and collected at 10 Hz. An open-source annotation tool called SUSTechPoints [38] was used to annotate the 3D objects. The annotation methodology was based on the labeling specifications for WOD [7]. A total of 6680 frames were annotated, and the annotated data included more than 235,000 objects belonging to three classes: vehicles, pedestrians, and cyclists. Figure 2 shows the data collection platform and annotation data.

4.3 | Image classification data from AIHub

AIHub [14] is an integrated artificial intelligence platform that provides more than 20 types of autonomous driving datasets available only to domestic applicants. We downloaded the Sensor Fusion Multi-Object Tracking and Prediction Data dataset [39] from AIHub and used it to train and validate our image classification model. This dataset was created for research on multiple object detection and prediction tasks and consists of 400,000 instances of images and LiDAR data. However, in this study, we only considered images with a resolution of 2880×1860 pixels. Although the image data contained

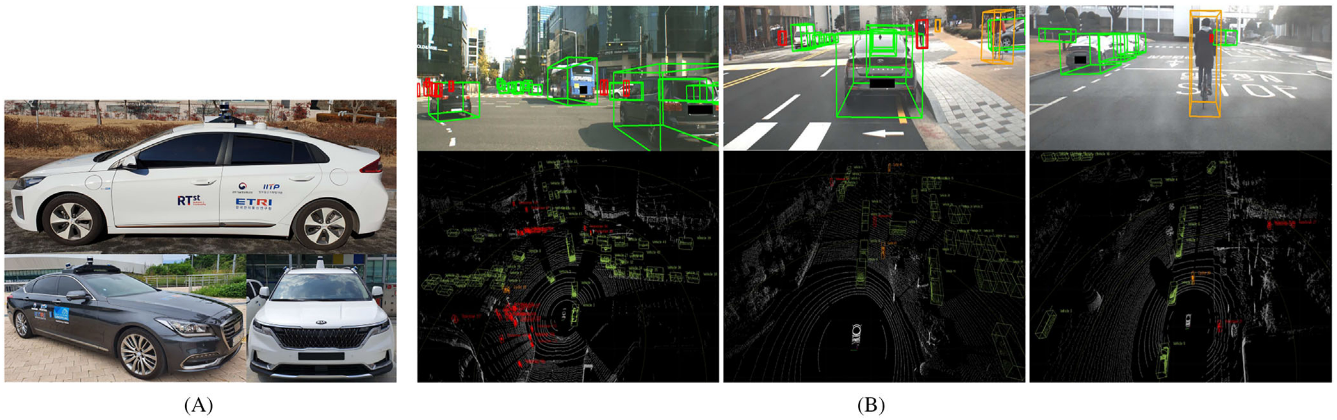


FIGURE 2 Data recording platforms and examples of the ETRI 3D MOD dataset. (A) Three different types of platforms: Ioniq, Genesis G80, and Carnival electric cars. (B) Projected 3D BBs in images (upper three photographs) and point-cloud data (lower three photographs). Green, vehicles; red, pedestrians; orange, cyclists.

annotation information for eight different classes, we utilized only three classes, namely, vehicles, pedestrians, and cyclists, and cropped the images using the following equations to store them:

$$\text{crop\_img}_w \geq 10,\ \text{crop\_img}_h \geq 48. \tag{6}$$

where crop_img_w and crop_img_h are the width and height of a cropped image, respectively. To create noise-class data, we first generated candidates by generating random numbers based on the following conditions:

$$\begin{gathered} 0 \leq x_1 \leq \text{img}_w - 300, \\ \frac{\text{img}_h}{4} \leq y_1 \leq \text{img}_h - 300, \\ 9 \leq \text{crop\_img}_w \leq 300, \\ 48 \leq \text{crop\_img}_h \leq 300, \end{gathered} \tag{7}$$

where x_1 and y_1 represent the coordinates of the upper-left corner of the candidates and img_w and img_h represent the width and height of the image, respectively. As the top and bottom parts of the image represented the sky and ground, respectively, we excluded these areas from data generation. Next, we compared the candidates with the ground truth and saved only one candidate per image file with an intersection over union (IoU) [40] of zero with the ground truth. The IoU was calculated as follows:

$$\text{IoU}(B_1, B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2}. \tag{8}$$

The BBs of the generated candidates and ground truth are denoted by B_1 and B_2, respectively.

5 | EXPERIMENTS

5.1 | Experimental settings

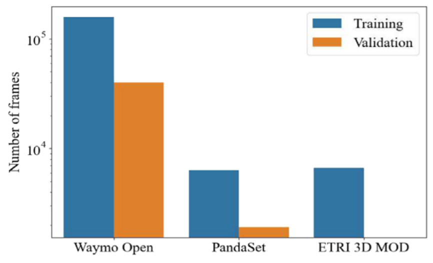
As the numbers and types of classes differed for each dataset, we renamed them using the class names of the ETRI 3D MOD dataset, as indicated in Table 2. Classes not shown in the table were not included in this study.

The 3D object detection model in the upper-left block of Figure 1 was trained and validated using the WOD, PandaSet, and ETRI 3D MOD datasets. Figure 3 shows the distribution of the data in terms of the number of frames and objects per frame within the training and validation sets of the three datasets. Because of the relatively small size of the ETRI 3D MOD dataset, it was used only as part of the training dataset. This prevented the 3D object detection model from overfitting the ETRI 3D MOD dataset and prioritized the establishment of the generalization performance by verifying the performance of the model using other datasets such as PandaSet.

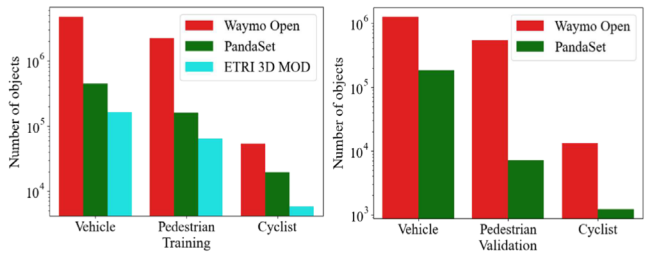
The 2D image classification model, shown in the lower-left block of Figure 1, was trained and validated using the AIHub dataset. Figure 4A shows the data distribution for each class in the training and validation sets from the AIHub data.

TABLE 2 Renaming class names of two different datasets based on ETRI 3D MOD dataset.

ETRI 3D MOD	Waymo Open	PandaSet
Vehicle	Vehicle	Car
		Pickup_Truck
		Medium-sized_Truck
		Semi-truck
		Towed_Object
		Other_Vehicle_Construction_Vehicle
		Other_Vehicle_Uncommon
		Other_Vehicle_Pedicab
		Emergency_Vehicle
		Bus
Pedestrian	Pedestrian	Pedestrian
		Pedestrian_with_Object
Cyclist	Cyclist	Motorcycle
		Bicycle
		Personal_Mobility_Device
		Motorized_Scooter

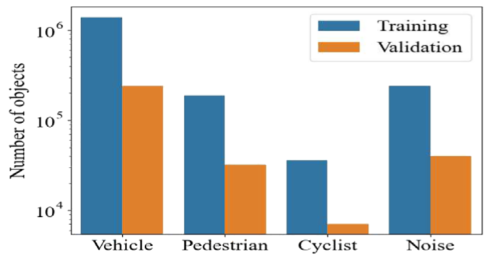


(A)

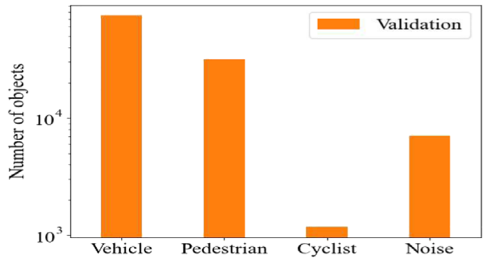


(B)

FIGURE 3 (A) Distributions of data for three different datasets of 3D objects in terms of numbers of frames in training and validation sets. (B) Numbers of objects within these frames.



(A)



(B)

FIGURE 4 Statistics for two different image classification datasets: (A) AIHub and (B) ETRI 2D MOC.

5.2 | 3D object detection results

5.2.1 | Implementation details

We tested three 3D object detection models to construct a LiDAR-based object detection model, as shown in Figure 1. The primary source of training data was the WOD dataset. Model$_{\mathrm{WOD}}$ refers to a 3D object detection model trained and validated using WOD. The other two models were trained on the PandaSet and ETRI 3D MOD datasets using different approaches: training from scratch and utilizing a pretrained model subsequently trained on WOD. The model trained from scratch was denoted as Model$_{\mathrm{P\&E}}$. Model$_{\mathrm{P\&E\_F}}$ represented a fine-tuned model that used both the PandaSet and ETRI 3D MOD datasets from Model$_{\mathrm{WOD}}$, which enhanced the accuracy and generalization ability of the model.

To train Model$_{\mathrm{WOD}}$, we utilized the training set shown in Figure 3 and configured the detection range of the model to [−74.88 m, 74.88 m] along the X and Y axes and [−2 m, 4 m] along the Z axis. Additionally, we configured the sizes of the pillars to [0.32 m, 0.32 m, 6 m] along the $[X, Y, Z]$ axes. The maximum number of pillars was set to 32,000, with a maximum of 20 point-cloud data points per pillar. For data augmentation, we applied ground-truth sampling as proposed in [16]. Furthermore, we excluded the ground-truth samples containing fewer than five point-cloud datapoints within a 3D cuboid to improve the quality of the training data. For global rotation and scaling, we applied values within the ranges of $[-\pi/4, \pi/4]$ and [0.95, 1.05], respectively. The other training methods were consistent with CPP [21]. The model was trained for 36 epochs using the AdamW optimizer [42] with a maximum learning rate of 3×10^{-3} and a batch size of 4 per graphics processing unit (GPU) on eight NVIDIA RTX 3090 GPUs.

To train Model$_{\mathrm{P\&E}}$, we used the AdamW optimizer [42] and trained it for 36 epochs with a maximum learning rate of $3\times10-4$. Model$_{\mathrm{P\&E\_F}}$ was fine-tuned for 5 epochs using the same optimizer with a maximum learning rate of $3\times10-3$. Ground-truth sampling was not utilized because it decreased the detection performance. Instead, we maintained consistency with the training methods used for Model$_{\mathrm{WOD}}$.

5.2.2 | Results

We used the evaluation metric for WOD to evaluate the performance of our 3D object detection model:

$$\mathrm{AP}=\frac{1}{100}\sum_{r\in R}\max(p(r')|r'\geq r), \tag{9}$$

where $R=\{0.0, 0.01, 0.02, ..., 1.0\}$ and $\max(p(r')|r'\geq r)$ represent the precision, which is the highest among all the recall values greater than or equal to r. The IoU threshold for the vehicle class was 0.7, whereas that for the pedestrian and cyclist classes was 0.5. The label difficulty was divided into two levels. LEVEL_1 corresponded to objects that had more than five points in the 3D cuboid or were not marked as LEVEL_2. LEVEL_2 corresponded to objects that had at least one but no more than five points in the 3D cuboid or were marked as LEVEL_2.

To validate the performance of Model$_{\mathrm{WOD}}$ on WOD, we used the validation set shown in Figure 3. We then evaluated the predicted results for the validation set using an evaluation server operated by WOD. The results are listed in Table 3 and show the changes in performance for the WOD validation set for different modifications to the CPP [21]. These modifications included removing the intensity data (NI), using smaller strides in the upsampling block (SUS), and combining both straegies (Model$_{\mathrm{WOD}}$). The evaluation metrics included the mean average precision (mAP) for all objects and AP/L1 and AP/L2 for individual object classes (vehicles, pedestrians, and cyclists). Compared with CPP, our proposed Model$_{\mathrm{WOD}}$ exhibited reductions in mAP/L1 and mAP/L2 of 3.45% and 3.32%, respectively. However, the computational cost was significantly reduced by 45.4%, which is a noteworthy advantage for real-world deployment in autonomous driving applications, where computational resources are often limited.

To evaluate the model trained on the PandaSet and ETRI 3D MOD datasets, we used the library of detection metrics provided by WOD because we conducted the evaluation on a local server, unlike the model trained on WOD. The results are summarized in Table 4. Model$_{\mathrm{P\&E\_F}}$ had the best performance for all classes among the three models, particularly for the pedestrian and cyclist classes. This suggests that pretraining a model on a large dataset such as WOD can improve its performance, even when trained on data from different LiDAR sensors. Based on our findings, the use of pretrained models is recommended for LiDAR-based 3D object detection tasks to improve performance.

5.3 | Image classification results

5.3.1 | Implementation details

We utilized the AIHub dataset [39] in Figure 4A to train our image classification model and classified the objects into four classes: vehicles, pedestrians, cyclists, and noise. The vehicle class accounts for 75% of the AIHub dataset, whereas the cyclist class accounts for only 1.9%, making it a highly imbalanced dataset. When training a model on imbalanced data, overfitting to the majority class may occur, resulting in a low classification performance for the minority class. To address this issue, we used a data resampling technique called repeat factor sampling [43]

TABLE 3 Performance comparison of 3D object detection on Waymo Open validation set. We compared the performance of 3D object detection models, including those capable of real-time operation on modern GPUs toward integration into AVs. †, reported in [41]; ‡, reported in [7]. No intensity (NI) indicates the case where intensity data were not used, while smaller upsampling strides (SUS) denotes the case where the strides of the upsampling block were [0.5, 1, 2] instead of [1, 2, 4]. L1 and L2 refer to LEVEL_1 and LEVEL_2, respectively, while mAP is the mean average precision.

Model	Params. (M)	FLOPs (G)	Vehicle AP/L1 AP/L2	Pedestrian AP/L1 AP/L2	Cyclist AP/L1 AP/L2	All mAP/L1 mAP/L2
SECOND [16]†	–	–	67.90 59.40	57.80 49.80	54.00 52.30	59.90 53.83
PointPillars [17]‡	4.84	261.36	63.30 55.60	62.10 55.90	–	–
CPP [21]	5.22	346.50	**74.24 66.12**	**72.94 65.16**	**57.58 55.40**	**68.25 62.23**
SECOND+AGO-Net [41]	–	–	69.20 60.60	59.30 51.80	55.30 53.50	61.27 55.30
CPP-NI	5.22	346.48	73.61 65.47	71.01 63.42	54.78 52.70	66.47 60.53
CPP-SUS	**4.81**	157.25	72.43 64.44	70.47 62.92	57.51 55.35	66.80 60.91
CPP-NI-SUS (Model$_{\mathrm{WOD}}$)	**4.81**	**157.23**	71.64 63.63	69.20 61.56	53.57 51.55	64.80 58.91

TABLE 4 Comparison of results for PandaSet validation. L1 and L2 represent LEVEL_1 and LEVEL_2, respectively.

Model	Vehicle AP/L1 AP/L2	Pedestrian AP/L1 AP/L2	Cyclist AP/L1 AP/L2	All mAP/L1 mAP/L2
Model$_{\mathrm{WOD}}$	60.02 46.01	26.54 19.23	13.42 10.28	33.33 25.17
Model$_{\mathrm{P\&E}}$	68.02 59.67	16.28 13.40	10.06 8.51	31.45 27.19
Model$_{\mathrm{P\&E\_F}}$	**75.12 67.70**	**37.62 31.33**	**24.06 20.27**	**45.60 39.77**

to improve the generalization performance of the model. We applied two techniques to augment data: random resizing and cropping of an input image as well as horizontal flipping with a probability of 50%. The model was trained for 100 epochs using a stochastic gradient descent optimizer [44] with a learning rate of 0.1, momentum of 0.9, and a weight decay of 0.0001. The learning rate decreased by a factor of 10 at the 50th and 75th epochs. Training was performed on eight NVIDIA RTX 3090 GPUs with a batch size of 16 per GPU.

5.3.2 | Results

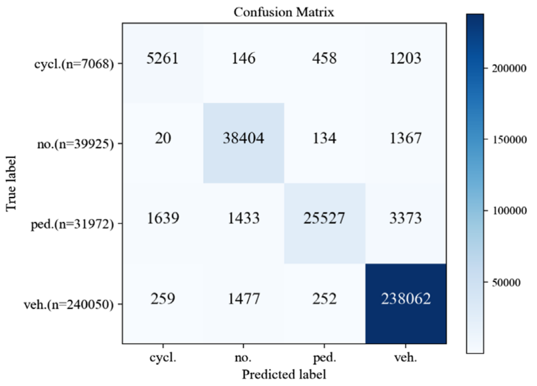
To evaluate the trained model on the AIHub validation set, we used two evaluation metrics: the accuracy and balanced accuracy [45]. The accuracy was calculated as follows:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \text{TP}_i}{\sum_{i=1}^{N} (\text{TP}_i + \text{FP}_i)}. \tag{10}$$

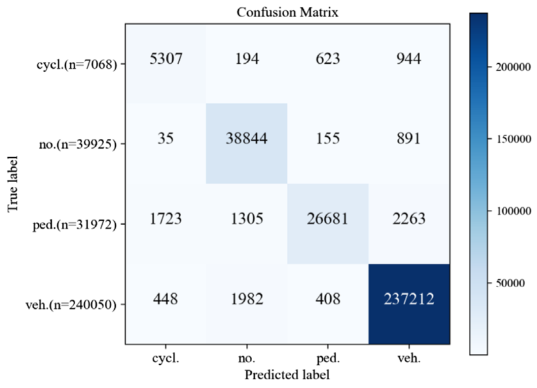
The balanced accuracy was calculated as follows:

$$\text{Balanced accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}, \tag{11}$$

where N is the number of classes. The evaluation results of the ResNet50-Tiny model trained using repeat factor sampling are listed in Table 5. Comparing the performance of the ResNet50-Tiny model with and without repeat factor sampling, it was observed that setting hyperparameter **t** to 0.2 resulted in a statistically significant improvement in the balanced accuracy (1.25%) compared with the case where **t** was set to 0 (that is, no resampling training). However, the improvement in accuracy was modest (0.24%). These results suggest that repeat factor sampling with an appropriate value of **t** can improve the model performance, particularly in terms of balanced accuracy. Furthermore, the confusion matrix in Figure 5 indicates that the ResNet50-Tiny model exhibited a higher classification performance for all nonvehicle classes for **t** = 0.2 compared with the case for **t** = 0.



(A)



(B)

FIGURE 5 Confusion matrices of proposed ResNet50-Tiny model based on hyperparameter **t** for repeat factor sampling: (A) **t** = 0 and (B) **t** = 0.2.

TABLE 5 Performance comparison of proposed ResNet50-Tiny model based on hyperparameter **t** for repeat factor sampling. Threshold **t** is a hyperparameter that determines the extent to which rare classes are resampled during training. Value **t** = 0 indicates no resampling.

Model	t	Image size	Params. (M)	FLOPs (G)	ACC.	Balanced ACC.
ResNet50-Tiny	0	32^2	23.51	1.31	96.31	87.41
ResNet50-Tiny	0.1	32^2	23.51	1.31	96.34	87.3
ResNet50-Tiny	0.2	32^2	23.51	1.31	**96.56**	**88.66**
ResNet50-Tiny	0.3	32^2	23.51	1.31	96.48	87.86
ResNet50-Tiny	0.4	32^2	23.51	1.31	96.43	88.21

5.4 | Evaluation on ETRI 2D MOC dataset

5.4.1 | Dataset

We constructed an ETRI 2D MOC dataset using data collected from real environments to validate the performance of the proposed method. We used the same AV to collect the ETRI 3D MOD dataset but chose different locations and times for data collection. The objects from the three classes were detected using the LiDAR-based 3D object detection model developed and evaluated in this study, $\text{Model}_{\text{P\&E\_F}}$. For objects with a confidence score of 0.2 or higher, we cropped and saved the corresponding parts of the objects within the image using camera calibration information. As shown in Figure 4B, the annotators classified the saved images into four classes: vehicles, pedestrians, cyclists, and noise.

5.4.2 | Results

We evaluated a fusion approach for object detection by combining the predictions of two different models using the prediction refinement method outlined in Section 3.3. The evaluation results are listed in Table 6 and indicate that the best overall performance was achieved using hyperparameters (0.7, 0.3), reaching an accuracy of 91.48% and balanced accuracy of 78.1%. $\text{Model}_{\text{P\&E\_F}}$ was fine-tuned using both the PandaSet and ETRI 3D MOD datasets, whereas ResNet50-Tiny was trained only on the public AIHub dataset [39]. Consequently, different camera models were used for training and testing, leading to a decrease in performance owing to the domain gap. Despite the lower performance of the ResNet50-Tiny model compared with that of $\text{Model}_{\text{P\&E\_F}}$, we demonstrated the practicality of our proposed method by showing improved performance when fusing the classification information of the two models.

5.4.3 | Runtime

To assess the runtime performance of the proposed method, we used the same server (Intel Xeon E5-2690 v4 central processing unit operating at 2.6 GHz and an NVIDIA RTX 3090 GPU) and the PyTorch framework [18] for training to assess the runtime performance of the proposed method. We evaluated the performance of $\text{Model}_{\text{P\&E\_F}}$ using point-cloud data consisting of the 9831 frames previously used to generate the ETRI 2D MOC dataset with a batch size of 1. The performance evaluation included both preprocessing (voxelization) and post-processing (non-maximum suppression) steps. The experimental results showed a processing speed of 32.8 ms per frame with an average detection rate of 74.5 objects per frame.

Furthermore, we measured the performance of the ResNet50-Tiny model with the ETRI 2D MOC dataset and batch size of 32. The measurement time included preprocessing steps such as image resizing. According to the experimental results, the inference time was 11.99 ms per batch, corresponding to 0.36 ms per image. We demonstrated that objects could be detected and classification results could be refined in real time using the two methods mentioned above. When applied to autonomous driving systems, a further reduction in the processing time may be achieved using optimizers such as TensorRT [19].

TABLE 6 Performance comparison of proposed method for different hyperparameters (λ_1, λ_2) used for prediction refinement. Only the object classification results from the LiDAR- and camera-based detection models are used when λ_1 and λ_2 are set to 1.0, respectively.

Model	λ_1	λ_2	ACC.	Balanced ACC.
$\text{Model}_{\text{P\&E\_F}}$	1.0	0	88.58	66.03
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.9	0.1	88.58	66.03
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.8	0.2	89.99	70.49
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.7	0.3	**91.48**	78.1
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.6	0.4	89.88	**78.78**
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.5	0.5	86.56	76.27
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.4	0.6	84.46	74.15
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.3	0.7	82.97	72.63
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.2	0.8	82.02	71.5
$\text{Model}_{\text{P\&E\_F}}$ + ResNet50-Tiny	0.1	0.9	81.34	70.82
ResNet50-Tiny	0	1.0	80.8	70.33

TABLE 7 Results of 3D object detection on KITTI validation set. For the OT method, we used a scale factor of 0.9. To fuse Model$_{\text{WOD}}$ (w/OT) and ResNet50-Tiny models, we set λ_1 and λ_2 to 0.7 and 0.3, respectively.

	Car		Pedestrian		Cyclist		All	
Model	**AP$_{\text{BEV}}$**	**AP$_{\text{3D}}$**	**AP$_{\text{BEV}}$**	**AP$_{\text{3D}}$**	**AP$_{\text{BEV}}$**	**AP$_{\text{3D}}$**	**mAP$_{\text{BEV}}$**	**mAP$_{\text{3D}}$**
PointRCNN (w/SN) [46]	71.30	47.10	–		–		–	
SECOND-IoU (w/SN) [47]	**78.96**	59.20	**53.72**	**50.44**	44.61	41.43	59.10	50.36
Model$_{\text{WOD}}$	53.31	13.65	36.72	32.79	55.21	50.84	48.41	32.43
Model$_{\text{WOD}}$ (w/OT)	76.76	59.43	40.74	34.90	60.24	**56.09**	59.25	50.14
Model$_{\text{WOD}}$ (w/OT) + ResNet50-Tiny	77.48	**60.06**	46.44	40.42	**61.05**	55.90	**61.66**	**52.13**

5.5 | Evaluation on KITTI dataset

We aimed to demonstrate the generalization ability of the proposed EMOS method by evaluating its performance on data collected from an environment different from that used to train the deep learning model. To this end, we evaluated the performance of 3D object detection on the KITTI dataset, which is widely used and publicly available.

5.5.1 | Implementation details

To assess the performance of 3D object detection on the KITTI dataset and preprocessed point-cloud data, we employed the OpenPCDet toolbox [48]. Furthermore, we conducted experiments using Model$_{\text{WOD}}$ trained on the Waymo Open dataset for comparison with the results in [46, 47]. During detection of 3D objects using point-cloud data from Model$_{\text{WOD}}$ and the KITTI dataset, the predicted object sizes were slightly larger than those obtained from the Waymo Open data, despite accurately predicting the object center coordinates. This issue may stem from the differences in the average object sizes between datasets or variations in the point-cloud density caused by different LiDAR sensors [46]. Moreover, variations in annotation guidelines may contribute to this problem. To overcome this issue, we applied a straightforward approach called output transformation (OT) [46]. Instead of adding a fixed value, we multiplied the predicted sizes by a scale factor.

5.5.2 | Results

To evaluate the performance of 3D object detection on the KITTI validation set, we used the metrics employed in the KITTI benchmark. We reported the average precision (AP) for both BEV and 3D evaluations using 40 recall positions. For cars, we applied an IoU threshold of 0.7,

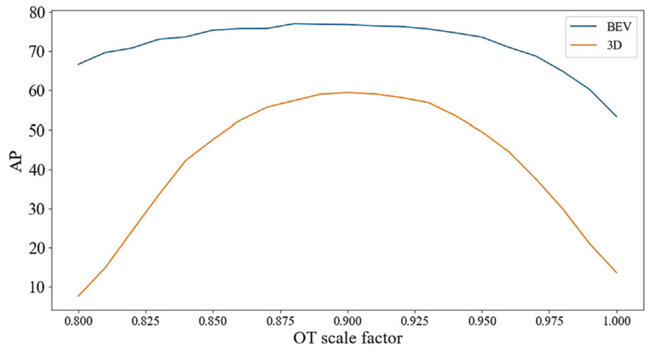


FIGURE 6 Average precision of cars detected by Model$_{\text{WOD}}$ according to OT scale factor.

whereas for the other objects, the threshold was set to 0.5. Our evaluation specifically focused on the moderate difficulty level among the available levels of easy, moderate, and hard. Table 7 lists the results of the proposed method and recent studies [46, 47]. All the models were trained on the Waymo Open training set for 3D object detection without using KITTI training data. Model$_{\text{WOD}}$ exhibited subpar performance because of the domain disparity between the training and test sets. To address this issue, we used the OT method with a scale factor of 0.9. Figure 6 shows the AP of car detection using Model$_{\text{WOD}}$ according to the scale factor. The AP was improved for all classes by approximately 45.8%, specifically in AP$_{\text{3D}}$ for cars. By fusing Model$_{\text{WOD}}$ (w/OT) and ResNet50-Tiny, we observed improvements in mAP$_{\text{BEV}}$ and mAP$_{\text{3D}}$ by 2.41% and 1.99%, respectively, indicating the best performance. Thus, we demonstrated the generalization ability of the proposed method by its superior performance, even when tested in environments with sensors different from those used for model training.

6 | CONCLUSIONS

We propose a simple yet effective LiDAR–camera fusion method to support autonomous driving. We validated the performance of the proposed method using both our constructed ETRI 2D MOC dataset and the publicly available KITTI dataset. The experimental results demonstrate that the proposed method performs well in both real-world and previously unseen environments, demonstrating its robustness and generalization ability. Our study may contribute to the field of autonomous driving by creating a new 3D object detection dataset called ETRI 3D MOD, which we believe will facilitate future research in this area. Overall, we expect that our study and findings promote the development of safe and reliable AVs.

ACKNOWLEDGEMENTS

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) under grant funded by the Korean government (MSIT) (No. 2021-0-00891, Development of AI Service Integrated Framework for Autonomous Driving) and the Korean government (MSIP) (No. 2020-0-00002, Development of Standard SW Platform-Based Autonomous Driving Technology to Solve Social Problems of Mobility and Safety for Marginalized Public Transport Communities).

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Dongjin Lee https://orcid.org/0000-0003-4527-6214
Seung-Jun Han https://orcid.org/0000-0002-9594-6047

REFERENCES

1. D. Choi, S.-J. Han, K.-W. Min, and J. Choi, *PathGAN: Local path planning with attentive generative adversarial networks*, ETRI J. **44** (2022), no. 6, 1004–1019.
2. S.-J. Han, J. Kang, K.-W. Min, and J. Choi, *DiLO: Direct light detection and ranging odometry based on spherical range images for autonomous driving*, ETRI J. **43** (2021), no. 4, 603–616.
3. J. Kang, S.-J. Han, N. Kim, and K.-W. Min, *ETLi: Efficiently annotated traffic Lidar dataset using incremental and suggestive annotation*, ETRI J. **43** (2021), no. 4, 630–639.
4. Y. Zhou and O. Tuzel, *VoxelNet: End-to-end learning for point cloud based 3D object detection*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA), 2018, pp. 4490–4499.
5. H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, *nuScenes: A multimodal dataset for autonomous driving*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA), 2020, pp. 11621–11631.
6. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, *Vision meets robotics: The KITTI dataset*, Int. J. Robot. Res. **32** (2013), no. 11, 1231–1237.
7. P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, *Scalability in perception for autonomous driving: Waymo open dataset*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA), 2020, pp. 2446–2454.
8. B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, *Argoverse 2: Next generation datasets for self-driving perception and forecasting*, arxiv preprint, 2023. DOI 10.48550/arXiv.2301.00493
9. J. Lambert, A. Carballo, A. M. Cano, P. Narksri, D. R. Wong, E. Takeuchi, and K. Takeda, *Performance analysis of 10 models of 3D LiDARs for automated driving*, IEEE Access **8** (2020), 131699–131722.
10. Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov, *SPG: Unsupervised domain adaptation for 3D object detection via semantic point generation*, (IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, Canada), 2021, pp. 15446–15456.
11. X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, *TransFusion: Robust Lidar-camera fusion for 3D object detection with transformers*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA), 2022, pp. 1090–1099.
12. X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, *Multi-View 3D object detection network for autonomous driving*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA), 2017, pp. 1907–1915.
13. Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, *BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation*, (IEEE International Conference on Robotics and Automation (ICRA)), London, UK), 2023.
14. AIHub, 2023. Available from: https://aihub.or.kr/ [last accessed February 2023].
15. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *PointNet: Deep learning on point sets for 3D classification and segmentation*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA), 2017, pp. 652–660.
16. Y. Yan, Y. Mao, and B. Li, *SECOND: Sparsely embedded convolutional detection*, Sensors **18** (2018), no. 10, 3337.
17. A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, *PointPillars: Fast encoders for object detection from point clouds*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA), 2019, pp. 12697–12705.
18. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: An imperative style, high-performance deep learning library*, (International Conference on Neural Information Processing Systems (NIPS)), Vancouver, Canada, 2019, pp. 8026–8037.
19. NVIDIA TensorRT, 2023. Available from: https://developer.nvidia.com/tensorrt [last accessed February 2023].
20. Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Y. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, *End-to-end multi-view*

fusion for 3D object detection in LiDAR point clouds, (Conference on Robot Learning (CoRL)), Osaka, Japan), 2019, pp. 923–932.

21. T. Yin, X. Zhou, and P. Krahenbuhl, *Center-based 3D object detection and tracking*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA), 2021, pp. 11784–11793.
22. A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet classification with deep convolutional neural networks*, Commun. ACM **60** (2017), no. 6, 84–90.
23. K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, (International Conference on Learning Representations (ICLR), CA, USA), 2015.
24. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA), 2015, pp. 1–9.
25. K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA), 2016, pp. 770–778.
26. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A large-scale hierarchical image database*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA), 2009, pp. 248–255.
27. M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, *The Pascal Visual Object Classes (VOC) challenge*, Int. J. Robot. Res. **88** (2010), no. 2, 303–338.
28. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft COCO: Common objects in context*, *European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, (eds.), Springer International Publishing, Zurich, Switzerland, 2014, pp. 740–755.
29. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA), 2017, pp. 4700–4708.
30. Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A ConvNet for the 2020s*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA), 2022, pp. 11976–11986.
31. S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, *Aggregated residual transformations for deep neural networks*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA), 2017, pp. 1492–1500.
32. M. Tan and Q. V. Le, *EfficientNet: Rethinking model scaling for convolutional neural networks*, (International Conference on Machine Learning(ICML), California, USA), 2019, pp. 6105–6114.
33. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *MobileNets: Efficient convolutional neural networks for mobile vision applications*, arXiv Preprint, 2017. http://arxiv.org/abs/1704.04861
34. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An image is worth 16×16 words: Transformers for image recognition at scale*, (International Conference on Learning Representations (ICLR), Vienna, Austria), 2021.
35. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, *Swin transformer: Hierarchical vision transformer using shifted windows*, (IEEE/CVF International Conference on Computer Vision (ICCV)), 2021, pp. 10012–10022.
36. D. Lee, D.-W. Kang, J. Kang, J.-Y. Kim, K.-W. Min, J.-H. Park, K.-B. Sung, Y.-S. Song, T.-H. An, Y.-W. Jo, D. Choi, J.-D. Choi, and S.-J. Han, *Apparatus for recognizing object of automated driving system using error removal based on object classification and method using the same*, Tech. Report US11507783B2. U.S. Patent. USA, 2022.
37. P. Xiao, S. Shao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang, and D. Yang, *PandaSet: Advanced sensor suite dataset for autonomous driving*, (IEEE Intelligent Transportation Systems Conference (ITSC), Indianapolis, USA), 2021, pp. 3095–3101.
38. E. Li, S. Wang, C. Li, D. Li, X. Wu, and Q. Hao, *SUSTech POINTS: A portable 3D point cloud interactive annotation platform system*, (IEEE Intelligent Vehicles symposium (IV)), Nevada, USA), 2020, pp. 1108–1115.
39. H. Jung, Sensor fusion multi-object tracking and prediction data, 2021. Available from: https://aihub.or.kr/ [last accessed February 2023].
40. M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, *The Pascal Visual Object Classes challenge: A retrospective*, Int. J. Comput. Vis. **111** (2014), 98–136.
41. L. Du, X. Ye, X. Tan, E. Johns, B. Chen, E. Ding, X. Xue, and J. Feng, *AGO-Net: Association-guided 3D point cloud object detection network*, IEEE Trans. Pattern Anal. Machine Intell. (PAMI) **44** (2022), no. 11, 8097–8109.
42. I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, (International Conference on Learning Representations (ICLR), LA, USA), 2019.
43. A. Gupta, P. Dollar, and R. Girshick, *LVIS: A dataset for large vocabulary instance segmentation*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), California, USA), 2019, pp. 5356–5364.
44. I. Sutskever, J. Martens, G. Dahl, and G. Hinton, *On the importance of initialization and momentum in deep learning*, (International Conference on Machine Learning (ICML), Georgia, USA), 2013, pp. 1139–1147.
45. K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, *The balanced accuracy and its posterior distribution*, (International Conference on Pattern Recognition (ICPR), Istanbul, Turkey), 2010, pp. 3121–3124.
46. Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, *Train in Germany, Test in the USA: Making 3D object detectors generalize*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, USA), 2020, pp. 11713–11723.
47. J. Yang, S. Shi, Z. Wang, H. Li, and X. Q, *ST3D++: Denoised self-training for unsupervised domain adaptation on 3D object detection*, IEEE Trans. Pattern Anal. Machine Intell. **45** (2023), no. 5, 6354–6371.
48. OpenPCDet: An open-source toolbox for 3D object detection from point clouds, 2020. Available from: https://github.com/open-mmlab/OpenPCDet [last accessed July 2023].

49. E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, *Randaugment: Practical automated data augmentation with a reduced search space*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Seattle, USA), 2020, pp. 702–703.
50. J. Ngiam, B. Caine, W. Han, B. Yang, Y. Chai, P. Sun, Y. Zhou, X. Yi, O. Alsharif, P. Nguyen, Z. Chen, J. Shlens, and V. Vasudevan, *StarNet: Targeted computation for object detection in point clouds*. arXiv Preprint, 2019, https://arxiv.org/abs/1908.11069
51. Z. Yang, Y. Zhou, Z. Chen, and J. Ngiam, *3DMAN: 3D multi-frame attention network for object detection*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Virtual)), 2021, pp. 1863–1872.

AUTHOR BIOGRAPHIES

Dongjin Lee received his BS degree from Dankook University, Republic of Korea, in 2008, and his MS degree from the University of Science and Technology, Republic of Korea, in 2013. From 2008 to 2010, he was an engineer at Samsung S1 Corporation, Republic of Korea. He has been a researcher at ETRI since 2013. He is currently working toward the PhD degree at Chungnam National University, Republic of Korea. His research interests include computer vision, machine learning, affective computing, and autonomous driving.

Seung-Jun Han received his BE degree in Control and Instrumentation Engineering from Pukyong National University, Busan, Republic of Korea, in 1998. He received his MS degree in Electronics Engineering from Pusan National University, Busan, Republic of Korea in 2000. From 2000 to 2010, he worked as a principal researcher at Sane System Co., Ltd., Anyang, Republic of Korea. In 2011, he was a researcher in the Department of Aerospace Engineering at KAIST, Daejeon, Republic of Korea. Since 2012, he has been working as a senior researcher at ETRI, Daejeon, Republic of Korea. His primary research interests include computer vision, structure from motion, simultaneous localization and mapping, and machine learning.

Kyoung-Wook Min received his BS and MS degrees in Computer Engineering from Pusan National University, Republic of Korea, in 1996 and 1998, respectively. He received his PhD degree in Computer Engineering from Chungnam National University, Republic of Korea, in 2012. Since 2001, he has been a principal researcher at ETRI, Republic of Korea, and is the director of the Autonomous Driving Intelligence Research Section. His main research area is autonomous driving.

Jungdan Choi received her PhD degree in Image Processing from Chungnam National University, Daejeon, Republic of Korea, in 2006 and her BSc and MSc degrees in Computer Graphics at Chung-Ang University, Seoul, Republic of Korea, in 1993 and 1995, respectively. Since 1995, she has worked as a principal researcher at ETRI. She is also an assistant vice-president of the Mobility Robot Research Division, Superintelligence Creative Research Laboratory, ETRI. Her research interests include automotive image processing and recognition, 3D modeling, and rendering.

Cheong Hee Park received her PhD degree in Mathematics from Yonsei University, Republic of Korea, in 1998. She received her MS and PhD degrees in Computer Science from the University of Minnesota, USA, in 2002 and 2004, respectively. Since 2005, she has been a professor at the Department of Computer Science and Engineering, Chungnam National University, Republic of Korea. Her research interests include machine learning, pattern recognition, and data mining.

How to cite this article: D. Lee, S.-J. Han, K.-W. Min, J. Choi, and C. H. Park, *EMOS: Enhanced moving object detection and classification via sensor fusion and noise filtering*, ETRI Journal **45** (2023), 847–861. DOI 10.4218/etrij.2023-0109