Novel On-Road Vehicle Detection System Using Multi-Stage Convolutional Neural Network

JISU KIM1, SUNGJUN HONG2, AND EUNTAI KIM1

1School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea
2School of Information Technology, Sungkonghoe University, Seoul 08359, South Korea

Corresponding author: Euntai Kim (etkim@yonsei.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant through the Korea Government [Ministry of Science and ICT (MSIT)], Development of Driving Environment Data Transformation and Data Verification Technology for the Mutual Utilization of Self-Driving Learning Data for Different Vehicles under Grant 2021-0-00800.

ABSTRACT General object-detection methods based on deep learning have received considerable attention in the field of computer vision. However, when they are applied to vehicle detection (VD) in a straightforward manner to realize an intelligent vehicle (IV), a graphics processing unit (GPU) is required for their real-time implementation. The use of GPUs is unacceptable in commercial VD systems. A novel on-road VD method comprising the use of a multi-stage convolutional neural network (MSCNN) is proposed to solve this problem. In the MSCNN, the properties of the vehicles are exploited, and an efficient region proposal specialized for vehicles is developed. The proposed MSCNN comprises four stages: vehicle lower-boundary detection, vehicle upper-boundary detection, region proposal network (RPN), and vehicle classification. Effective anchor boxes are generated in the lower- and upper-boundary-detection stages with appropriate sizes for vehicles of all scales. The bounding box of the vehicle within the anchor box is determined in the RPN stage. In the last stage, the predicted bounding boxes are classified as vehicles or non-vehicles. Finally, the proposed method is applied to the KITTI, CrowdAI, and AUTTI datasets, and its advantages are demonstrated by comparing its performance with those presented in previous studies. The proposed MSCNN realizes an average precision (72.1%) on the KITTI dataset while running on a central processing unit (CPU).

INDEX TERMS Vehicle detection, convolutional neural network, region proposal network, Bayesian approach.

I. INTRODUCTION

Autonomous driving is considered one of the most attractive emerging technologies in the industry and academia. Various technologies are being developed for the reliable implementation of autonomous driving. Object detection [1]–[3] around an autonomous vehicle is an important technology that is required for realizing autonomous driving, and popular examples of object detection (OD) include pedestrian detection, vehicle detection (VD), and cyclist detection. Several sensors and combinations of sensors, as well as monocular cameras owing to their low cost, have been widely used in the implementation of OD. This study focuses on VD alone with the use of a monocular camera [4]–[7].

The majority of studies on VD using a monocular camera can be classified into two categories: 1) the use of hand-crafted features and 2) the use of deep learning methods. In the former, hand-crafted features are used. The most popular hand-crafted features used for VD are the histogram of oriented gradient (HOG) [8], Haar-like features [9], Gabor features [10], aggregate channel features (ACFs) [11], and position and intensity-included HOG (πHOG) [12]. The Haar-like features require less computation time than the HOG, and Gabor features; however, the detection performance obtained using Haar-like features is inferior to that obtained using HOG or Gabor features. The ACF is a popular hand-crafted feature that includes HOG, normalized gradient magnitudes, and LUV [11]. πHOG compensates for the information loss that occurs when computing a histogram with a combination of position and intensity information [12]. Fig. 1(a)–(d) illustrate the basic structures obtained from classical studies wherein hand-crafted features are used. Fig. 1(a)–(c) depict the exhaustive searches. Fig. 1(d) depicts an image to which the region proposal method was
FIGURE 1. Various strategies for VD: (a)–(c) exhaustive searches with hand-crafted features, (d) region proposal method with hand-crafted features, and (e)–(g) region proposal method with CNN.

applied [13]–[15]. Fig. 1(a) illustrates a simple sliding window wherein a search is performed for vehicles at all possible locations and scales [16], [17]. In Fig. 1(b), the cost of generating multi-scale feature maps is reduced by approximating the feature map [11]. Fig. 1(c) depicts the structure wherein different classifiers are assigned to each scale, thus realizing a considerable speed-up [18]. In Fig. 1(d), the region proposal method is employed to avoid the implementation of an exhaustive search and reduce the number of applications of the classifiers.

Deep learning methods based on convolutional neural networks (CNNs) have received significant attention as a general tool for OD in the field of computer vision. The learned features obtained from the deep learning method have a considerably higher discriminatory power for general OD than classical hand-crafted features [19]. Fig. 1(e)–(g) depict some of the structures of the general OD based on the CNN. The structure illustrated in Fig. 1(e) is similar to that in Fig. 1(d), where regions with a CNN belong to this type [20]. In Fig. 1(f), the feature map is computed for the entire image only once to extract the features of all the region proposals [21], [22], thus reducing the computation time.

In Fig. 1(g), a region proposal network (RPN) is used to share the feature map with the classifier [23].

Although the above method [23] shares the feature map between the RPN and classifier, an alternative training is still required. This learning method has become a bottleneck in real-time applications. To address this problem, one-step frameworks based on regression and classification methods have been proposed. You only look once (YOLO) [24] and single shot multibox detector [25] are representative one-step frameworks. The YOLO divides the input image into a grid, and each grid is used to predict the bounding box and object class. This network can process images at 45 fps with a better detection performance than other real-time systems. The second improved version, YOLOv2 [26], was later proposed for improving the detection performance and speed by using anchor boxes, dimension clusters, and multi-scale training. YOLOv2 can process images at 67 fps with a better detection performance than YOLO. A third improved version, YOLOv3 [27], was also presented. YOLOv3 adopted predictions across scales to detect small objects. In terms of accuracy and speed, YOLOv3 demonstrates significant benefits over YOLOv2. However, YOLOv3 incurs a high computational cost and requires a large amount of time without a GPU. For the convenience of deployment, a simplified version called YOLOv3-tiny [27] was also presented. The structure of YOLOv3-tiny is similar to that of YOLOv3; however, the number of layers and filters is less than that of YOLOv3.

The above methods [20]–[27] based on deep learning demonstrate excellent detection performance in comparison with those in other studies [8]–[12]. However, the straightforward application of general OD to VD presents some problems. First, because the entire image is used as an input to deep learning and millions of convolution operations are used to perform OD, a GPU is required to meet the acceptable computation time. However, the use of GPUs is unacceptable in commercial VD systems. Second, the VD is likely to fail in detecting small vehicles at a large distance from the camera. As the layer becomes deeper, the output size of the layer becomes smaller due to the pooling operation, and thus, the output from the last layer does not fit small vehicles. To solve the above problems, a new VD method called multi-stage CNN (MSCNN) based on deep learning is proposed in this study. The proposed MSCNN comprises four stages: vehicle lower-boundary detection, vehicle upper-boundary detection, RPN, and vehicle classification. The first three stages correspond to the new region proposal method, which is specialized for VD. In the last stage, a shallow CNN is trained to classify the given proposal into vehicles and non-vehicles. The contributions of this study can be summarized as follows:

1) In contrast to the general OD methods comprising the use of the deep learning method, the properties of the vehicles are exploited, and a new region proposal method specialized for VD is proposed.

2) A cascade of shallow CNNs is used to detect vehicles.
3) A new method for using a new image strip in the second stage is developed. The new method is based on the position-wise vehicle-height predictor (PVHP), as described in Section IV. Thus, the proposed MSCNN has the following advantages over the general OD applied to the VD:

1) The MSCNN outperforms general OD methods, such as YOLOv3 and YOLOv3-tiny in terms of the log-average miss rate, average precision, and computation time.

2) As the MSCNN consists of a few shallow networks and does not use deep networks, it can be implemented on a central processing unit (CPU).

3) The MSCNN demonstrates a higher detection performance for small vehicles than a general OD, such as YOLOv3 and YOLOv3-tiny. This is because, as the MSCNN is implemented using shallow networks, a considerably smaller number of pooling layers is used in the MSCNN than that in the general OD, thus preventing the loss of information.

The MSCNN may appear to be slightly similar to the CNN cascade methods, such as those presented in [33]; however, the philosophy of the MSCNN is different from that in [33]. Fig. 2 presents a comparison of the previous CNN cascade method and the proposed MSCNN. In Fig. 2(a), the CNN cascade method considers the detections by the CNN in the earlier stage as region proposals. As the stage increases, deeper CNNs are used to detect vehicles, and the number of false positives is reduced. Therefore, the same region can be tested several times, which is time-consuming. However, in Fig. 2(b), each stage in the MSCNN plays its semantic role, such as lower-boundary or upper-boundary detection, and generates proposals efficiently and effectively.

The remainder of this paper is organized as follows. In Section II, the proposed VD system based on the MSCNN is briefly outlined. In Section III, the proposed VD method comprising the use of the MSCNN and the network structure for each stage are explained in detail. A new method of using a new image strip based on PVHP to improve the detection performance is developed in Section IV. In Section V, the experimental results of the proposed methods are presented and compared with those of previous methods, and the merits of the proposed methods are discussed. In Section VI, the conclusions of this study are presented.

II. SYSTEM OVERVIEW

The proposed MSCNN comprises four stages, as shown in Fig. 3. The four stages are 1) vehicle lower-boundary detection, 2) vehicle upper-boundary detection, 3) RPN, and 4) vehicle classification. Fig. 3 depicts an outline of the MSCNN.

In the first stage, the lower bounds of the vehicles are detected using a CNN. This CNN is based on a previous study [28], and is similar to that used in detecting the road boundary because all vehicles in an urban environment are on the road. In Fig. 3, the red bars represent the lower bounds of the vehicles. In the second stage, the upper bounds of the vehicles are detected using another CNN, and the input of the CNN is an image cut below the lower boundary generated using the first CNN. The architecture of the CNN in the second stage is the same as that in the first stage. In Fig. 3, the blue bars represent the upper bounds of the vehicles. In the third stage, pre-defined boxes called anchor boxes [23] are generated using the lower and upper bounds of the vehicles. The anchor boxes are indicated in green in Fig. 3. To consider various views, such as front, rear,
or side views of the vehicles, anchor boxes of two different aspect ratios (1:1 and 2:1) are generated. These anchor boxes are refined using RPN [23]. The refined anchor boxes are termed object proposals. In Fig. 3, the green and cyan boxes represent the initial anchor boxes and corresponding refined object proposals, respectively. In the fourth stage, the object proposals are classified using the last CNN. The compactness of the shallow CNN facilitates the implementation of the proposed MSCNN in an embedded system without a GPU.

### III. PROPOSED METHOD

#### A. LOWER BOUNDARY DETECTION OF VEHICLES

In the first stage, the CNN returns the lower boundaries of the vehicles in an urban environment. To obtain the lower boundaries, an RGB image \( I(x, y, c) \in \mathbb{R}^{W \times H \times 3} \) is divided into \( N \) multiple strips,

\[
S_i^j(x, y, c) = I(x, y, c) \mid (i-1)w_s \leq x \leq iw_s, 1 \leq y \leq H, c \in \{R, G, B\}
\]

(1)

where \( i = 1, 2, \cdots, N \), \( W \) and \( H \) are the width and height of the image, respectively, and \( w_s = \lfloor \frac{W}{M} \rfloor \) is the width of the image strip. Fig. 4 presents the main objectives of the first stage. When the \( i \)-th image strip \( S_i^j(x, y, c) \) is given, the lower boundary detection problem for the strip becomes a regression problem as shown in Fig. 4(b), where the input is the image strip and the output \( y_i(i) \) comprises the lower boundaries of the vehicles in the strip. To simplify this regression problem, we convert the lower boundary detection of the vehicles into a multiple class classification problem for each image strip. In particular, we divide (or quantize) the height of an image into \( M \) intervals. The regression problem is approximated using a multiple-class classification problem:

\[
\hat{y}_i(i) = \arg \min_{\hat{y}_i \in [0, 1, \cdots, M]} \frac{\hat{y}_i}{L}
\]

(2)

as shown in Fig. 4(c), where \( L = \lfloor \frac{H}{M} \rfloor \) is the length of the interval and \( I([\cdot]) \) is an indicator function that returns a value of 1 if the argument is true and 0 otherwise. The classification problem is solved using the CNN. The classification result \( \hat{y}_i(i) \in [0, 1, \cdots, M] \) denotes the label corresponding to the vertical position of the vehicle, and \( \hat{y}_i(i) = 0 \) corresponds to the case in which there is no vehicle in the image strip. The non-vehicle class aids in eliminating unnecessary region proposals and reducing the subsequent computation time. Using the image strips and lower boundaries of the vehicles, the first CNN is trained to estimate the quantized vertical position of the lower boundary of the vehicle \( \hat{y}_i \). The last layer of the first CNN comprises \( M + 1 \) nodes with a softmax function, and each node returns the probability of each interval being the lower boundary of the vehicle. Using these definitions, we minimize the cross-entropy loss as follows:

\[
L_{cls}(n_i, \hat{y}_i) = -\log \left( \frac{\exp(n_i(\hat{y}_i))}{\sum_{i=0}^{M} \exp(n_i(i))} \right)
\]

(3)

where \( n_i \) denotes the \( M + 1 \) outputs of the last layer of the first CNN. Among the \( M + 1 \) nodes, the node with the maximum probability denotes the prediction \( p_i \) for the first CNN as follows

\[
p_i = \arg \max_{k \in [0, 1, \cdots, M]} \left( \frac{\exp(n_i(k))}{\sum_{i=0}^{M} \exp(n_i(i))} \right).
\]

(4)

If no vehicle is present, the first node has the maximum probability \( (p_i = 0) \). Based on the \( p_i \) estimated by the first CNN, the lower boundary of the vehicle \( \hat{y}_i = p_i \times \lfloor \frac{H}{M} \rfloor \) is predicted as shown in Fig. 4(d).

Fig. 5 presents a summary of the workflow. Fig. 5(a) presents two deep-learning modules: C_M and F_M. The C_M module comprises four deep-learning layers: a convolution layer, batch normalization layer, max pooling layer, and rectified linear activation function. The F_M module comprises three deep-learning layers: a fully connected convolution layer, batch normalization layer, and rectified linear activation function. These modules use shallow layers to reduce computation time. Fig. 5(b) presents a summary of the workflow. In this study, the input to the first CNN is set to a size of \( 10 \times 240 \) (\( w_s = 10 \), and \( H = 240 \)), and all the image strips of different sizes are resized to a size of \( 10 \times 240 \). The height \( H \) of the strips is quantized into \( M = 48 \) intervals of the same length \( \lfloor \frac{H}{M} \rfloor \) = 5 pixels. On combining the outputs from the image strips, the lower boundaries of the vehicles in the image are predicted, as indicated by the red bars in Fig. 5.

#### B. UPPER BOUNDARY DETECTION OF VEHICLES

In the second stage, the CNN returns the upper boundaries of the vehicles in an urban environment. Fig. 6 presents the main objective of the second stage. The input to the second CNN is an image strip cut below the lower boundary,

\[
S_u^j(x, y, c) = \left\{ S_i^j(x, y, c) \mid 1 \leq y \leq \hat{y}_i(i) \right\}
\]

(5)
where $y_\bar{r}$ is the upper boundary of the vehicle in the resized image strip.

as shown in Figs. 6(a) and (b). Before being applied to the CNN, the cut strip $S^i_r(x, y, c)$ is resized to $10 \times 240$.

The resized image of $S^i_r(x, y, c)$ is denoted by $S^i_{r,r}(x, y, c)$. The detection of the upper boundary of a vehicle is formulated as a multiple-class classification problem,

$$\hat{y}_{u,r}(i) = \arg \max_{j \in \{0, 1, 2, \ldots, M-1\}} \frac{y_{u,r} \cdot \exp(n_u(\hat{y}_{u,r}))}{L}$$  \hspace{1cm} (6)

where $y_{u,r}$ is the upper boundary of the vehicle in the resized image strip.

The second CNN is trained to predict the quantized vertical position of the upper boundary $\hat{y}_{u,r}$ of a vehicle in the resized image strip $S^i_{u,r}(x, y, c)$. Using these definitions, we minimize the cross-entropy loss as follows

$$L_{cls}(n_u, \hat{y}_{u,r}) = - \log \left( \frac{\exp(n_u(\hat{y}_{u,r}))}{\sum_{i=0}^{M-1} \exp(n_u(i))} \right)$$  \hspace{1cm} (7)

where $n_u$ denotes the $M$ outputs of the last layer of the second CNN. As in the case of the first CNN, the prediction $p_{u,r}$ for the second CNN is as follows:

$$p_{u,r} = \arg \max_{k \in \{0, 1, 2, \ldots, M-1\}} \left( \frac{\exp(n_u(k))}{\sum_{i=0}^{M-1} \exp(n_u(i))} \right)$$  \hspace{1cm} (8)

Using $p_{u,r}(i)$, the upper boundary of the vehicle of the $i$-th image strip is computed as follows

$$\hat{y}_{u}(i) = \frac{240}{M} \cdot \frac{\hat{y}_{u,r}(i) \cdot \hat{y}_{r}(i)}{240} = \hat{y}_{u,r}(i) \cdot \hat{y}_{r}(i)$$  \hspace{1cm} (9)

as shown in Figs. 6(c) and (d), where $\hat{y}_{u,r}(i) = p_{u,r}(i) \left[ \frac{240}{M} \right]$. Fig. 7 depicts a summary of the workflow. The workflow is similar to that of the first stage, and the architecture of the second CNN is the same as that of the first CNN; however, different weights are used. $\hat{y}_{u}$ is indicated by the blue bars in Fig. 7.

C. REGION PROPOSAL NETWORK

In the third stage, two types of anchor boxes of different aspect ratios are generated for each vehicle candidate region, and an RPN is used to refine the anchor boxes to provide region proposals for the vehicles. Fig. 8 depicts the workflow of the RPN. As vehicles have varied width-to-height ratios, we generated two types of anchor boxes of aspect ratios 1:1 and 2:1 to represent all possible width-to-height ratios. For each vehicle candidate region characterized by a pair of upper and lower boundaries, the center position of the
anchor box is initialized as the mean of the upper and lower boundaries. For the \( j \)-th anchor box in the \( i \)-th image strip, the anchor box is parameterized as follows.

\[
a_{ij} = \left( a_{ij}^x, a_{ij}^y, a_{ij}^w, a_{ij}^h \right) = \left( \frac{(2i - 1) w_2}{2}, \frac{y_{ij}^u + y_{ij}^l - y_{ij}^l}{R_j}, \frac{y_{ij}^l - y_{ij}^l}{R_j} \right)
\]

where \( \left( a_{ij}^x, a_{ij}^y \right) \) denotes the center position of the \( j \)-th anchor box, \( a_{ij}^w \) denotes the width of the \( j \)-th anchor box, \( a_{ij}^h \) denotes the height of the \( j \)-th anchor box, \( R_j \) is the aspect ratio of the \( j \)-th anchor box, and \( y_{ij}^u \) and \( y_{ij}^l \) are the upper and lower boundaries of the \( j \)-th image strip, respectively.

In Fig. 8, two anchor boxes of different aspect ratios are indicated by green boxes for an image strip. After generating all the anchor boxes, the RPN is used to estimate the bounding box via the parameterizations of the four coordinates as follows [23]:

\[
\tilde{p}_{ij} = f_N \left( p_{ij}, a_{ij} \right) = \left( p_{ij}^x - a_{ij}^x, p_{ij}^y - a_{ij}^y, \log \left( \frac{p_{ij}^w}{a_{ij}^w} \right), \log \left( \frac{p_{ij}^h}{a_{ij}^h} \right) \right)
\]

where \( p_{ij} = \left( p_{ij}^x, p_{ij}^y, p_{ij}^w, p_{ij}^h \right) \) is the prediction for the region proposal, and \( \tilde{p}_{ij} = \left( \tilde{p}_{ij}^x, \tilde{p}_{ij}^y, \tilde{p}_{ij}^w, \tilde{p}_{ij}^h \right) \) denotes the normalized value for \( p_{ij} \). The parameter \( g_{ij} \) for the \( j \)-th anchor box is considered to be a nearby ground truth of the vehicle from the \( j \)-th anchor box. Using these definitions, we minimize the regression loss as follows.

\[
L_{reg} \left( \tilde{p}_{ij}, \tilde{g}_{ij} \right) = \| \tilde{p}_{ij} - \tilde{g}_{ij} \|_2^2,
\]

where \( \tilde{g}_{ij} \) denotes the normalized value for \( g_{ij} \) calculated using the function \( f_N \left( g_{ij}, a_{ij} \right) \) in (11).

**D. VEHICLE CLASSIFIER**

In the last stage, the vehicle classifier network is applied to the region proposals, and the vehicles are then detected. The system illustrated in Fig. 9 is in the fourth stage of vehicle classification. The architecture of the fourth CNN is similar to that of the third one; however, different weights are used. The input to the fourth CNN is the region proposals returned from the third CNN. As in the third stage, two vehicle classifier networks are implemented separately depending on which anchor box generated region proposals in the last stage. One vehicle classifier network related to the anchor box (1:1) is suitable for detecting the front or rear views of the vehicles, whereas the other RPN related to the aspect ratio of 2:1 is focused on the region proposals for the side view of the vehicles in this study.
The input to the fourth CNN is set to an image size of 64 × 64, and each region proposal from the third CNN is resized to 64 × 64. The fourth CNN is trained as a binary classification, and the classification loss is in the form of a cross-entropy loss over the two classes (vehicle versus non-vehicle).

The proposed RPN makes some overlapping region proposals for each vehicle, as shown in Fig. 9. The detected bounding boxes are accordingly overlapped, thereby deteriorating the detection performance. To prevent performance degradation, NMS [29] is used for the post-processing. NMS is used to eliminate the detection results having low confidence scores, thus retaining only the detections having a maximum confidence score among the overlapping detections. In Fig. 9, the VD results are indicated by blue bounding boxes, and the NMS results are indicated by red bounding boxes.

### E. TRAINING PROCESS

In this study, we adopt a four-step training algorithm to learn the CNN in each stage. Fig. 10 presents the training process for the proposed MSCNN. In the first step, we train the first CNN for the lower-boundary detection of vehicles. In the second step, we collect the training dataset using image strips cut below the lower boundary generated by the first CNN. We then train the second CNN for upper-boundary detection of the vehicles using this dataset. In the third step, we collect two training datasets using anchor boxes generated by the first and second CNNs. The two datasets consist of anchor boxes of aspect ratios of 1:1 and 2:1. Then, we train two RPNs using each dataset. In the fourth step, we separately train two vehicle classifiers using the region proposals generated by each RPN.

### IV. POSITION-WISE VEHICLE HEIGHT PREDICTOR

In the second stage for the upper-boundary detection of vehicles, the image strip from the lower boundary to the top is provided as the input to the second CNN. Because the image strip cut below the lower boundary is used, it is possible to prevent false upper-boundary detection of vehicles below the lower boundary. However, this approach still involves the scanning of the top of the image, including inefficient areas such as the sky or buildings. In this paper, a new approach using a PVHP is presented to generate a new image strip with the inefficient areas removed and enhance the upper-boundary detection performance. The main concept used herein is based on our previous work [12]. In our previous work, it was reported that the bottom position of the vehicle is highly correlated with the vehicle size. More specifically, large vehicles tend to be detected in the lower region of the image, whereas small vehicles tend to be detected in the upper region of the image. The vehicle-bottom position along the y-axis is plotted against the vehicle height in Fig. 11, wherein each dot corresponds to a vehicle in the AUTTI dataset [32]. In this manner, we defined the PVHP, which is given by the following linear statistical model.

\[
\begin{align*}
    w_h &= \beta_0 + \beta_1 w_y + \epsilon \\
    &= w_y^T \beta + \epsilon \\
    \epsilon &\sim \mathcal{N}(0, \frac{1}{\phi})
\end{align*}
\]

(13)

where \( \beta = (\beta_0, \beta_1)^T \), \( w_y = (1, w_y)^T \), \( w_y \) is the bottom position of the vehicle, \( w_h \) is the vehicle height, \( \mathcal{N}(\cdot) \) is a Gaussian distribution, and \( \phi \) denotes the precision (inverse of the variance). As shown in (13), \( \beta \) and \( \phi \) are the model parameters of the PVHP. We then restrict the vehicle height to \( k \) standard deviations from the mean \( w_y^T \beta \) using

\[
w_h \in \left[ w_y^T \beta - \frac{k}{\sqrt{\phi}}, \ w_y^T \beta + \frac{k}{\sqrt{\phi}} \right]
\]

(14)

In this study, we use \( k = 3 \) because three standard deviations from the mean accounts for 99.7% of the samples. Fig. 12 presents the results of the PVHP in the AUTTI dataset. In Fig. 11, the red line denotes the mean \( w_y^T \beta \) for the vehicle height and the red dotted lines denote the minimum and maximum values \( w_y^T \beta - k/\sqrt{\phi} \) and \( w_y^T \beta + k/\sqrt{\phi} \), respectively.

To explain the update of the PVHP parameters in detail, we denote the \( n \)-th detected vehicle as \( w_n = (w_n^y, w_n^h)^T \).
and the accumulated vehicles as $W^n = \{w^1, w^2, \ldots, w^n\}$. We denote the parameters of the PVHP after detecting $n$ vehicles as $(\beta, \phi) | W^n$. If we assume that

$$(\beta, \phi) | W^{n-1} \sim N_G \left( \mu^{n-1}, S^{n-1}, \alpha^{n-1}, \lambda^{n-1} \right)$$

(15)

and that the $n$-th vehicle $w^n = (w^n_y, w^n_h)^T$, which respects the PVHP,

$$w^n_h \sim N \left( \cdot | w^n_y^T \beta, \frac{1}{\phi} \right)$$

(16)

is detected, the parameters of the PVHP can then be updated using the Bayesian rule as

$$(\beta, \phi) | W^n \sim N_G \left( \mu^n, S^n, \alpha^n, \lambda^n \right)$$

(17)

and

$$\alpha^n = \alpha^{n-1} + \frac{1}{2}, \quad (S^n)^{-1} = \left\{ (S^{n-1})^{-1} + w^n_y w^n_y^T \right\}$$

$$\mu^n = S^n \left\{ (S^{n-1})^{-1} \mu^{n-1} + w^n_y w^n_y^T \right\}$$

$$\lambda^n = \lambda^{n-1} + \frac{1}{2} \left\{ (\mu^n)^T (S^n)^{-1} \mu^n - (w^n_y)^2 \right\} + (w^n_h)^2 - (\mu^n)^T (S^n)^{-1} (\mu^n)$$

(18)

where $N(\cdot)$ and $N_G(\cdot)$ denote the normal and normal-gamma distributions, respectively. The derivation of (18) is presented in our previous work [12]. Using the PVHP parameterized by $(\beta, \phi) | W^n$, we replace $\beta$ and $\phi$ in (14) with their means $\mathbb{E}(\beta | W^n)$ and $\mathbb{E}(\phi | W^n)$, respectively, and then obtain

$$w_y \in \left[ \begin{array}{c} w_y^T \mathbb{E}(\beta | W^n) - \frac{k}{\sqrt{\mathbb{E}(\phi | W^n)}} \\ \frac{k}{\sqrt{\mathbb{E}(\phi | W^n)}} \end{array} \right]$$

(19)

where

$$\mathbb{E}(\beta | W^n) = \mu^n, \quad (\phi | W^n) = \frac{\alpha^n}{\lambda^n}$$

(20)

The proposed MSCNN with the PVHP (MSCNN+PVHP) in Fig. 13 is outlined as follows:

1) Let us assume that $n-1$ vehicles are already detected. The first stage of the proposed method (MSCNN+PVHP) is the same as that of the MSCNN. When the lower boundary predicted using the first CNN is given, the PVHP is parameterized by $(\beta, \phi) | W^{n-1}$, which estimates the maximum value of the vehicle height $w_{h_{\text{max}}} = w^T_y \beta + k / \sqrt{\phi}$.

2) Using the maximum value of the vehicle height, we refine the image strip which is the input of the second CNN. The refined image strip includes a more efficient area than the previous one.

3) In the third and last stages, the workflow of the proposed method (MSCNN+PVHP) is the same as that of the MSCNN. If the $n$-th vehicle is detected, the parameters $\mu^n, S^n, \alpha^n, \lambda^n$ are updated using (18).

4) Steps 1 to 3 are repeated. Whenever the PVHP is updated by the current detected vehicle, the performance of the PVHP is more stable than that of the previous one.

V. EXPERIMENTAL RESULTS

In this section, the proposed MSCNN is applied to three datasets: KITTI [30], AUTTI [31], and CrowdAI [31]. Each dataset is built using a fixed monocular camera installed at the front of the host vehicle. The performance of the proposed method is compared with that of previous methods. All the algorithms are trained on a GTX TITAN X GPU using the Pytorch package [32] in Python, and the trained network is deployed on an Intel CPU i5-4670 for testing. The following four algorithms are compared in this section:

1) The vehicle detection based on the YOLOv3 (single class).
2) The vehicle detection based on the YOLOv3-tiny (single class).
3) The vehicle detection based on the MSCNN.
4) The vehicle detection based on the MSCNN with PVHP (MSCNN_PVHP).

The first and second algorithms are state-of-art detection methods, whereas the third and fourth algorithms are the ones proposed in this study. In the fourth algorithm, the PVHP is employed to accelerate the CNN of the upper boundary detector. YOLOv3 and YOLOv3-tiny generally consist of multiple outputs in each YOLO layer to achieve multiple classifications. For a fair experimental comparison, YOLOv3 and YOLOv3-tiny are implemented for a single class (vehicle) and not a multi-class. Thus, for a single class, YOLOv3 and YOLOv3-tiny require fewer convolution filters than those for multiple classes.

Table 1 lists the experimental environments for the training and testing of each algorithm. To meet the real-time performance requirements in a CPU-based test environment, YOLOv3 is trained using a small image $(160 \times 160)$. To obtain computation times similar to that of YOLOv3, the other algorithms are trained using the corresponding appropriate image sizes.
FIGURE 13. System for VD using MSCNN with PVHP ($N_V$ is the number of detected vehicles and $W_{h_{\text{max}}}$ is the maximum value of PVHP.)

TABLE 1. Image size and computation time for testing algorithms on CPU.

| DB     | Algorithm | Image width | Image height | Computation time (ms) |
|--------|-----------|-------------|--------------|-----------------------|
| KITTI  | YOLOv3    | 160         | 160          | 81                    |
|        | YOLOv3-tiny | 800         | 800          | 73                    |
|        | MSCNN     | 800         | 240          | 74                    |
|        | MSCNN_PVHP | 800         | 240          | 74                    |
| CrowdAI| YOLOv3    | 160         | 160          | 89                    |
|        | YOLOv3-tiny | 416         | 416          | 44                    |
|        | MSCNN     | 384         | 240          | 54                    |
|        | MSCNN_PVHP | 384         | 240          | 54                    |
| AUITI  | YOLOv3    | 160         | 160          | 78                    |
|        | YOLOv3-tiny | 416         | 416          | 43                    |
|        | MSCNN     | 384         | 240          | 46                    |
|        | MSCNN_PVHP | 384         | 240          | 46                    |

A. KITTI DATASET

The KITTI set comprises the detection and tracking DB, and the size of the images in the KITTI datasets is $1250 \times 375$. The KITTI detection DB comprises 7481 training images and 7518 testing images. The KITTI tracking DB comprises 21 training sequences (8008 images) and 29 test sequences (11095 images). The proposed and previous methods are trained using the training images in the KITTI detection DB and evaluated using the training images in the KITTI tracking DB because the testing set in the KITTI detection DB is not annotated. Figure 14 depicts the VD results for the KITTI dataset.

Figs. 14(a) and (b) are obtained using the YOLOv3 and YOLOv3-tiny models. For real-time performance using only the CPU, the two models are trained by reducing the image size. In this test environment based on the CPU, the YOLOv3 method has more false negatives than YOLOv3-tiny because YOLOv3 is trained by an image that is too small to satisfy the real-time performance. Figs. 14(c) and (d) are obtained using the proposed methods (MSCNN and MSCNN_PVHP). As shown in the figure, the proposed methods have fewer false positives and false negatives than the previous methods. In particular, the PVHP contributes to improving the VD rate of the MSCNN.

B. CrowdAI DATASET

The CrowdAI set includes driving in Mountain View, California, and neighboring cities during daylight. The size of the images in the CrowdAI dataset is $1920 \times 1200$. In the CrowdAI DB, there are 9400 images, and we randomly divide them into 4800 training images and 4600 testing images. Fig. 15 depicts some of the VD results for the CrowdAI dataset.

Figs. 15(a) and (b) are obtained using the YOLOv3 and YOLOv3-tiny models. As in the KITTI test environment,
the two models are trained by reducing the image size. In this CPU-based test environment, the YOLOv3 method has more false negatives than YOLOv3-tiny because YOLOv3 is trained using an image that is too small to satisfy real-time performance requirements. Figs. 15(c) and (d) are obtained using the proposed methods (MSCNN and MSCNN_PVHP). As shown in the figure, the proposed methods have fewer false positives and false negatives than previous methods. In particular, the PVHP improves the VD performance in the cases of small far-away vehicles and occlusions.

C. AUTTI DATASET
The AUTTI set is similar to the CrowdAI set. The size of the images in the AUTTI dataset is 1920 × 1200. The AUTTI DB comprises 10416 images, and we randomly divided them into 6400 training images and 4000 testing images. Fig. 16 depicts the VD results for the AUTTI dataset.

As shown in Fig. 16(a), a number of false positives and false negatives are obtained using YOLOv3. In particular, small vehicles far from the camera or vehicles...
that are seen from the side tend to remain undetected. In Fig. 16(b), YOLOv3-tiny exhibits more robustness than YOLOv3 and detects the vehicles viewed from various angles in a CPU-based test environment; however, small far-away vehicles are sometimes missed in this case. In Fig. 16(c), the MSCNN demonstrates an improved detection performance for small far-away vehicles. This implies that the MSCNN generates better region proposals than YOLOv3-tiny. Figure 16(d) depicts the results of the MSCNN with PVHP. It is demonstrated that the MSCNN with PVHP realizes a better detection performance than that without the PVHP.

D. ANALYSIS OF THE EXPERIMENTAL RESULTS

Fig. 17 depicts the VD results for special cases. When two vehicles appear in the same image strip, the first CNN predicts only one vehicle’s lower boundary with the maximum probability, as shown in Fig. 17(a). Although the first CNN misses the vehicle lower boundaries on some image strips (left image in Fig. 17(a)), the vehicle can be detected (left image in Fig. 17(c)) because one vehicle typically includes multiple image strips where the vehicle’s lower boundaries are correctly detected. However, if the first CNN misses all the lower boundaries of the vehicle (right image in Fig. 17(a)), the vehicle cannot be detected (right image in Fig. 17(c)). When the vehicle is hidden by some obstacles, the first CNN predicts a false lower boundary of the vehicle near obstacles, as shown in Fig. 17(b). As in the previous case, other lower-boundary detections of the vehicle are used to detect vehicles correctly, as shown in Fig. 17(d).

When the vehicle is hidden by some obstacles, the first CNN predicts a false lower boundary of the vehicle near obstacles, as shown in Fig. 17(b). As in the previous case, other lower-boundary detections of the vehicle are used to detect vehicles correctly, as shown in Fig. 17(d).

Tables 2, 3, and 4 present the comparisons of the log-average miss rates for each KITTI tracking DB sequence for all the methods under consideration. In the KITTI tracking DB, the ground truth of the vehicles has three occlusion levels: fully visible (level 0), partly occluded (level 1), and difficult to see (level 2). Fig. 18 illustrates examples of vehicles for each occlusion level. Tables 2, 3, and 4 present the detection performances for the three occlusion levels mentioned above. In Tables 2, 3, and 4, the best performance is indicated in red boldface for each KITTI tracking DB sequence. The proposed methods MSCNN and MSCNN_PVHP have advantages in considering all the occlusion levels.

Fig. 19 depicts the precision–recall curves for the KITTI tracking, CrowdAI, and AUTTI datasets. The precision–recall curve plots the precision versus recall. The value of the algorithm in the legend denotes the average precision. The
TABLE 2. Log Average Miss Rate In Case Of Occlusion Level 0 For Each KITTI Tracking DB Sequence.

| Sequence | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|----------|-----|-----|-----|-----|-----|-----|-----|
| YOLO3    | 0.7275 | 0.9229 | 0.9543 | 0.803 | 0.9414 | 0.941 | 0.9038 |
| YOLO3-tiny | 0.5848 | 0.5326 | 0.6405 | 0.3711 | 0.1757 | 0.339 | 0.6401 |
| MSCNN    | 0.213 | 0.1561 | 0.3691 | 0.1031 | 0.0349 | 0.1072 | 0.1375 |
| MSCNN_PVHP | 0.2091 | 0.1773 | 0.3413 | 0.089 | 0.0252 | 0.0875 | 0.1319 |

TABLE 3. Log Average Miss Rate In Case Of Occlusion Level 1 For Each KITTI Tracking DB Sequence.

| Sequence | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|----------|-----|-----|-----|-----|-----|-----|-----|
| YOLO3    | 0.7702 | 0.9649 | 0.8272 | 0.9558 | 0.7861 | 0.8795 |
| YOLO3-tiny | 0.4806 | 0.3758 | 0.573 | 0.2526 | 0.5376 | 0.3487 | 0.685 |
| MSCNN    | 0.118 | 0.1352 | 0.1334 | 0.0775 | 0.1801 | 0.1781 | 0.3146 |
| MSCNN_PVHP | 0.1181 | 0.1076 | 0.1364 | 0.0643 | 0.1923 | 0.0789 | 0.3117 |

TABLE 4. Log Average Miss Rate In Case Of Occlusion Level 2 For Each KITTI Tracking DB Sequence.

| Sequence | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|----------|-----|-----|-----|-----|-----|-----|-----|
| YOLO3    | 0.8738 | 0.6152 | 1 | 0.0000 | 0.6121 | 0.8257 | 0.8005 |
| YOLO3-tiny | 0.478 | 0.3289 | 0.5042 | 0.0000 | 0.3398 | 0.7039 | 0.4731 |
| MSCNN    | 0.1659 | 0.0805 | 0.0001 | 0.0000 | 0.0099 | 0.2378 | 0.1261 |
| MSCNN_PVHP | 0.1559 | 0.088 | 0.0702 | 0.0000 | 0.0081 | 0.2313 | 0.1353 |

The goal of VD is to increase the precision, recall, and average precision simultaneously. Figs. 19(a)–(c) correspond to the KITTI tracking, CrowdAI, and AUTTI datasets, respectively. In addition, YOLOv5 [34] is also compared in this figure. It is proved that YOLOv5 exhibits the best performance among YOLO series. For all three datasets, it can be observed that the MSCNNs outperform YOLOv3, YOLOv3-tiny, and YOLOv5. In particular, the MSCNN_PVHP exhibits the best performance among the five compared methods. To demonstrate the ability of the MSCNN in detecting small vehicles, the precision–recall curve is plotted in Fig. 20 when only small vehicles are considered. In the figure, small vehicles are defined as those having widths lesser than or equal to 50 (KITTI tracking dataset) or 200 (CrowdAI and AUTTI dataset) pixels. As shown in Fig. 20, the proposed MSCNNs also present a better detection performance than those of the other methods, and the performance of MSCNN_PVHP is the best. In particular, YOLOv3 and YOLOv3-tiny demonstrate significant performance degradation when only small vehicles are considered.
All the experiments conducted thus far have been focused on the detection performance (log-average miss rate, precision, recall, and average precision) without considering the computation time analysis. Considering only the previous evaluation criterion, it is difficult to determine which algorithm is better (e.g., one algorithm exhibits a high detection...
To consider both detection performance and computation time, the use of a new evaluation criterion, AP-FPS, is proposed in this study. AP-FPS is defined as

$$APFPS = \frac{1}{11} \sum_{t \in \{0, 4, \ldots, 40\}} p_{\text{interp}}(t)$$

(21)

where $t$ is the FPS, and $p(t)$ is the average precision (AP) value for the input in the AP-FPS curve. Fig. 21 depicts the AP-FPS curves in the KITTI tracking, CrowdAI, and AUTTI datasets using only the CPU. In the region below 5 fps, YOLOv3 exhibits a high detection performance, but for a high FPS, the detection performance decreases significantly. Therefore, YOLOv3 is unsuitable for use with only the CPU. As shown in Fig. 21, the proposed MSCNN_PVHP outperforms all compared algorithms above 5 fps. Table 5 lists the AP-FPS values defined by (21) for the three datasets. Considering the detection performance and computation time, the proposed MSCNN_PVHP incurs no additional computation cost (below 1 ms) compared to that of the MSCNN, although the detection performance of the MSCNN_PVHP is superior. In conclusion, our MSCNN demonstrates a higher AP-FPS value than the algorithms being compared. Thus, the proposed VD system is effective for real-time systems using only a CPU (without using a GPU).

### E. CONCLUSION

In this study, the MSCNN and PVHP was proposed to improve VD performance and reduce the corresponding computation time. The proposed methods demonstrated high performance in detecting vehicles while considering the KITTI, CrowdAI, and AUTTI datasets. The experimental results revealed that the MSCNN outperformed the state-of-the-art methods YOLOv3 and YOLOv3-tiny. Furthermore, the MSCNN with PVHP improved the VD performance without incurring an additional computation cost. Without the GPU, the proposed method is useful for obtaining the best performance. The proposed approach is expected to facilitate the implementation of the system in embedded environments.

### REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 3354–3361.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, “Towards fully autonomous driving: Systems and algorithms,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2011, pp. 163–168.
J. Kim et al.: Novel On-Road Vehicle Detection System Using MSCNN

[3] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, “Towards a viable autonomous driving research platform,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2013, pp. 763–770.

[4] S. Sivaraman and M. M. Trivedi, “Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis,” IEEE Trans. Intell. Transp. Syst., vol. 14, no. 4, pp. 1773–1795, Dec. 2013.

[5] S. Sivaraman and M. M. Trivedi, “A review of recent developments in vision-based vehicle detection,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2013, pp. 310–315.

[6] A. Mukhtar, L. Xia, and T. B. Tang, “Vehicle detection techniques for collision avoidance systems: A review,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 5, pp. 2318–2338, Oct. 2015.

[7] E. Ohn-Bar and M. M. Trivedi, “Learning to detect vehicles by clustering appearance patterns,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 5, pp. 2511–2521, Oct. 2015.

[8] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, “Learning a family of detectors via multiplicative kernels,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 3, pp. 514–530, Mar. 2011.

[9] W.-C. Chang and C.-W. Cho, “Online boosting for vehicle detection,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 40, no. 3, pp. 892–902, Jun. 2010.

[10] Z. Sun, G. Bebis, and R. Miller, “On-road vehicle detection using evolutionary Gabor filter optimization,” IEEE Trans. Intell. Transp. Syst., vol. 6, no. 2, pp. 125–137, Jun. 2005.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2014, pp. 346–361.

[12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1–9.

[13] D. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 798–807.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in Proc. Eur. Conf. Comput. Vis., Oct. 2016, pp. 21–37.

[15] J. Redmon and A. Farhadi, “YOLOv3: Better, faster, stronger,” 2016, arXiv:1804.02767. [Online]. Available: http://arxiv.org/abs/1804.02767

[16] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” 2016, arXiv:1612.08242. [Online]. Available: http://arxiv.org/abs/1612.08242

[17] J. Redmon, A. Farhadi, and D. Anguelov, “YOLOv3: An incremental improvement,” 2018, arXiv:1804.02767. [Online]. Available: http://arxiv.org/abs/1804.02767

[18] D. Levi, N. Garnett, and E. Fetaya, “StixelNet: A deep convolutional network for obstacle detection and road segmentation,” in Proc. Brit. Mach. Vis. Conf., 2015, p. 4.

[19] A. Neubeck and L. Van Gool, “Efficient non-maximum suppression,” in Proc. 18th Int. Conf. Pattern Recognit. (ICPR), 2006, pp. 850–855.

[20] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” Int. J. Robot. Res., vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[21] Uadcity Open-Source Data Sets. [Online]. Available: https://github.com/udacity/self-driving-car/tree/master/annotations

[22] P. V. Pachké et al., “PyTorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8026–8037.

[23] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 5325–5334.

[24] G. Jocher, A. Stoken, J. Borovec, and P. Rai. (Oct. 2020). Ultralytics/YOLOv5: V3.1- Bag Fixes and Performance Improvements. [Online]. Available: https://doi.org/10.5281/zenodo.4154370

JISU KIM received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2011, where he is currently pursuing the joint master’s and Ph.D. degrees. He has studied machine learning and pattern recognition for vehicle detection.

SUNGJIN HONG received the B.S. degree in electrical and electronic engineering and computer science and the Ph.D. degree in electrical and electronic engineering from Yonsei University, in 2005 and 2012, respectively. Upon his graduation, he worked for LG Electronics, a connected car industry, as a Senior Researcher, from 2012 to 2013. He worked for three years as a Lead Software Engineer with SMARTSTUDY, from 2013 to 2016. He was a Research Professor with the School of Electrical and Electronic Engineering, Yonsei University, from 2016 to 2020, prior to his current appointment. He is currently an Assistant Professor with the School of Information Technology, Sungkonghoe University, Seoul, South Korea. His research interests include machine learning, deep learning, computer vision, and their various applications. He received the IET Computer Vision Premium Award from the Institution of Engineering and Technology (IET), U.K., in 2015.

EUNTAI KIM received the B.S., M.S., and Ph.D. degrees in electronic engineering from Yonsei University, Seoul, South Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a full-time Lecturer with the Department of Control and Instrumentation Engineering, Hankyong National University, Kyonggi-do, South Korea. Since 2002, he has been with the faculty of the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was also a Visiting Researcher with the Berkeley Initiative in Soft Computing, University of California at Berkeley, CA, USA, in 2008. His current research interests include computational intelligence, statistical machine learning and deep learning, and their application to intelligent robotics, autonomous vehicles, and robot vision.