Received 25 July 2022, accepted 24 September 2022, date of publication 10 October 2022, date of current version 27 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3213328

APPLIED RESEARCH

An Intelligent Driving Assistance System Based on Lightweight Deep Learning Models

KO-FENG LEE[1], XIU-ZHI CHEN[1], CHAO-WEI YU[1], KAI-YI CHIN[2], (Member, IEEE), YIH-CHEN WANG[1], CHIA-YU HSIAO[1], AND YEN-LIN CHEN[1], (Senior Member, IEEE)
[1]Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei 10608, Taiwan
[2]Department of Big Data Management, Soochow University, Taipei 111, Taiwan

Corresponding author: Yen-Lin Chen (ylchen@mail.ntut.edu.tw)

This work was supported by the National Science and Technology Council of China under Contract MOST-109-2628-E-027-004-MY3, Contract MOST-110-2622-8-027-006, and Contract MOST-111-2218-E-027-003.

ABSTRACT An intelligent driver assistance system is developed in this study, which is able to remind the drivers to turn on the head lights or wipers through situation recognition method when driving at night or on rainy days. Furthermore, the object detection results from multiple perspective views are integrated, and the surrounding object detection results are produced for collision avoidance. The system is able to alarm the drivers based on the lightweight deep learning model and the distance estimation method when surrounding vehicles are too close. Experimental results show that the proposed methods and the chosen lightweight model in our proposed system obtain reliable performance and sufficient computational efficiency under limited computing resource. In conclude, our proposed system obtains high probability to be adopted for the development of advanced driver assistance systems (ADAS). The proposed system can not only assist the driver in determining the vision ahead, but also provide an instant overview of the vehicle's surrounding conditions to enhance driving safety.

INDEX TERMS Advanced driver assistance systems, distance estimation method, lightweight deep learning model, situation recognition method, vehicle detection method.

I. INTRODUCTION

Due to the rapid economic development, people gradually have the ability to own their own cars and begin to pay attention to their personal convenience, therefore, the number of private transports has increase. Apart from the traffic congestion, traffic accidents happen more frequently. According to the statistics of the Ministry of Transport of Taiwan government, the average annual number of traffic accidents was approximately 300,000, including nearly 1,600 deaths and about 400,000 persons injured in the past five years, moreover, a research shows that about 77% of the accidents are cause by the drivers themselves [1]. Based on the above statistical reports, this study will propose an intelligent driver assistance system which is able to assist drivers to perceive the traffic objects surrounding the host car, to turn on the head lights or wipers, and achieve collision prevention, which reduce the occurrence of accidents and improve traffic safety.

Considerable researches about safety improvement have been published in various countries, and major car manufacturers around the world have also displayed Advanced Driver Assistance System (ADAS) on their cars, which shows that the ADAS system is a major developing trend and has become an important system related to active safety. The ADAS has been developed for a period of time, the computing units have greatly increased and the price of environmental detection sensors has decreased with the advancement of technology, which driven the performance of auxiliary systems to be improved while the costs gradually being decreased. Currently, most ADAS systems use cameras and radars to sense the environment, and each has its own advantages and disadvantages, so most complete ADAS systems use both sensors to complement each other to build a complete environment around the vehicle.

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

Since careless driving behavior is an important factor in causing traffic incidents, the main goal of driving assistance system is to efficiently and rapidly warn the drivers before possible collisions through object detections. The traffic object detection techniques can roughly be divided into two categories. One category is the Depth sensor based systems, these methods adopted depth sensor information for object detection, such as the radar and light detection and ranging (LIDAR). The methods can obtain accurate object distances for rapid driver warning, however, the costs of such depth sensors are relatively expensive comparing to the optical camera systems [2]. The other category is the Camera based systems, these methods detect objects from the grabbed image sequences based on the machine learning techniques, and the consumer high-resolution cameras are relatively low-cost and more popular in the ADAS markets [3].

Therefore, the proposed system adopted the color features obtained by the optical camera for object detection. The proposed system can accurately and computational effectively recognize the traffic objects surrounding the host vehicle by the cameras.

For example, in 2001, researchers in [4] developed the use of the Haar-Like fast object features and made use of integrated images to speed up the cascade classifier features during detection phase in order to filter out areas of non-target objects quickly. In 2005, researchers in [5] proposed a brand-new feature, the Histogram of Oriented Gradients (HOG). This feature cut the object into several small cells to calculate the edge characteristics of the object and the histogram of its direction in each small cell and weight. After combining these cells with their own weights, the object can be detected. In 2019, researchers developed the use of the lane change warning system to improve the ADAS based on self-learning of the individual driving characteristics [6].

Methods mentioned above detect objects based on images, however, they are all using machine learning technology with supervised learning method in the artificial intelligence to detect objects. Machine learning allow computers to learn a set of skills from the data by itself and gradually improve the performance [7].

Artificial intelligence is now a popular topic which is closely related to many research and industrial areas. Generally, the artificial intelligence refers to the science and engineering technology for creating intelligent machines, especially computer program. Computers, robots controlled by the computers and software created by this technic are able to think similarly to humans with higher speed and stronger power. These techniques include supervised learning and unsupervised learning [8].

The Supervised learning requires manually filtered features, which is more prone to human bias or errors [9], therefore, this study used deep learning in machine learning. Deep learning models have the ability of self-learning, which means it can achieve the recognition effect without providing specific data set of training. FIGURE 1 is a schematic diagram of the neural network architecture. The Deep learning networks is the concept extended from neural networks. The training data are adopted into the convolutional neural network training to obtain the best classification model.

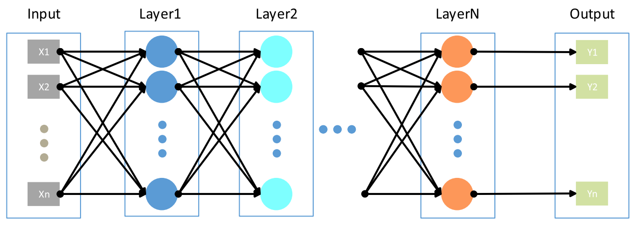


FIGURE 1. The schematic diagram of the neural network architecture [10].

A famous example of deep learning applications is AlphaGo [11] which is developed by Google. AlphaGo is an AI Go software based on deep learning. The AlphaGo's learning technology used the deep neural networks (DNN) that imitates the biological nervous system. Therefore, the DNN techniques are then widely applied in the field of artificial intelligence, including the perception issues in ADAS.

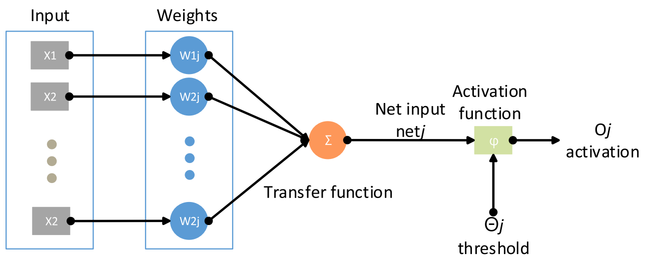


FIGURE 2. The diagram of single neuron operation of similar neural network.

FIGURE 2 is the diagram of the single neuron operation of similar neural network. In the neural network, there are usually several levels and each level has tens of hundreds of neurons. The neuron sums up the inputs of the neurons in the previous layer and converts the activation function as the output of the neuron. Each neuron will have a special connection with the neuron of the next layer so that the output value of the neuron of the upper layer passed to the neuron of the next layer after weight calculation [12].

Based on the concept mentioned above, this study developed an intelligent driving assistance system. The contributions of this study are listed as following:

1. This study had collected road vehicles and pedestrian image samples in both daytime and nighttime under multiple perspectives to build up the comprehensive dataset for model training.

2. A lightweight deep learning model with effective computation on the embedded platform, which can recognize seven kinds of objects, including cars, motorcycles, bicycles, trucks, buses, vans, and pedestrians, in multiple perspectives had chosen, based on the comparison results.

3. Based on the vehicle detection results, a distance estimation method had proposed for alarming drivers when there is a high risk of collisions with the surrounding vehicles.

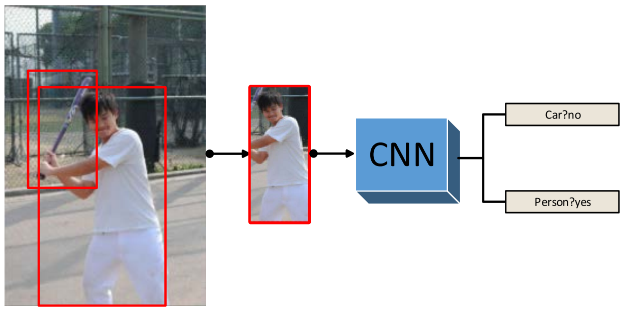


FIGURE 3. The architecture diagram of RCNN.

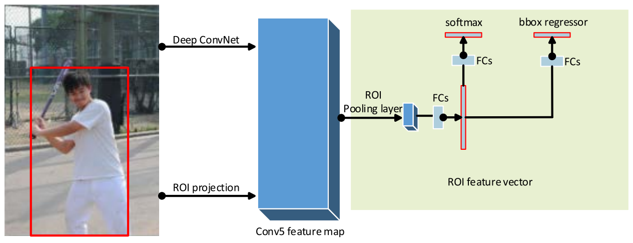


FIGURE 4. The architecture diagram of Fast-RCNN [17].

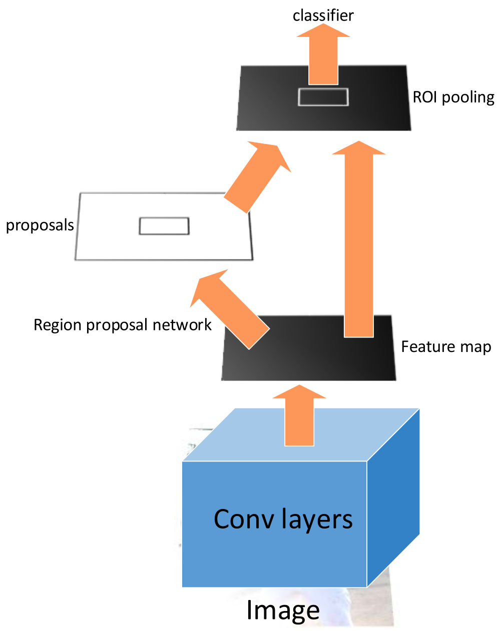


FIGURE 5. The architecture diagram of Faster-RCNN [18].

4. A situation recognition method based on machine learning techniques for reminding drivers to turn on head lights, wipers, etc., was proposed.

II. RELATED WORKS

This study is developed based on the deep learning techniques by using the deep convolution structure in the MobileNet [13] convolutional neural network to build a lightweight convolutional neural network. The classical convolutional neural network is the detection and classification algorithm of the RCNN [14] proposed by scholars. The RCNN series techniques can detect and classify objects without inputting the entire picture, therefore, it is broadly used on the newly developed object detection algorithms. This subsection will introduce the methods of object detection and classification developed based on the convolutional neural networks.

A. RCNN SERIOUS

In 2014, a research proposed the RCNN architecture [14], the architecture diagram is shown as FIGURE 3. First, the picture is divided into many small regions to extract frames with the selective search [15] in the RCNN. Next, the features are extracted and passed to the SVM [16] classifier by the convolutional network. Last, the SVM classifier determines whether the classification is correct and the regression corrects the position of the candidate frame.

As mentioned above, it is the relatively simple by using RCNN, but it also has two disadvantages. 1). It uses selective search [15] to divide the picture into many small areas. The extraction of the candidate frames is performed by using CPU, but it still consumes a lot of efficacy. 2). When the candidate frame is applied for the convolution calculation, it uses the amount of complex calculations which increases the computational complexity.

The Fast-RCNN improves the problem of the RCNN training and the prediction speed. FIGURE 4 shows the architecture diagram of the Fast-RCNN. The Fast-RCNN is modified with the RCNN architecture diagram and added a ROI pooling layer to the last convolutional layer. The multi-task in the loss function is applied to make the training and testing of the neural network easier and more convenient. Although the Fast-RCNN improves the training and prediction problems of the RCNN, the selective search is still applied to extract candidate frames, which is the same as the RCNN, that consumes most of efficacy and causes slow detection speed. Therefore, there is a Faster-RCNN.

In 2014, the research shows that the Faster-RCNN architecture has proposed [18]. The candidate frame extraction and consumption of computational performs are applied to solve the problem of the traditional Fast-RCNN and the RCNN. The Faster-RCNN integrates the extracts candidate frames into the network architecture, the architecture diagram of the Faster-RCNN is shown in FIGURE 5. The Faster-RCNN network architecture has more region proposal network layers than the traditional Fast-RCNN and the RCNN. The RPN layer is to classify, regress, and sort the retrieved candidate frames, then performs the ROI Pooling on the candidate frame. The region proposal network layer can implement an End-to-End convolutional network target detection model which can quickly extract high quality candidate frames. The Region proposal network not only speeds up the target detection speed, but also improves the target detection performance.

B. SINGLE SHOT MULTIBOX DETECTOR (SSD)

The Single Shot MultiBox Detector (SSD) is an object detection method proposed in 2016 [19]. The backbone of the SSD

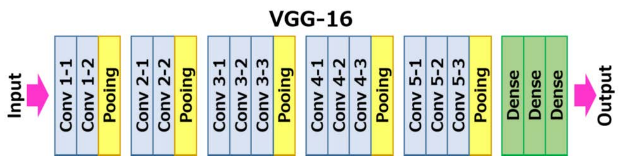


FIGURE 6. The architecture diagram of VGG-16 [20].

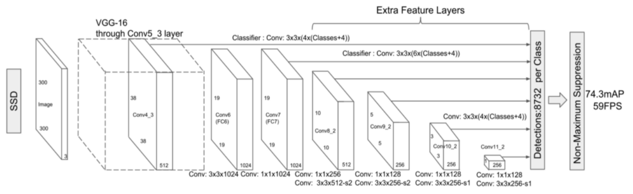


FIGURE 7. The architecture diagram of SSD [19].

is VGG-16 [20], FIGURE 6 is the architecture diagram of VGG-16. By removing the FC8 of the connected layers, The SSD converts the FC6 and FC7 into convolutional layers, and adds some convolutional layers with decreasing resolution.

The architecture diagram of SSD is shown in FIGURE 7 [19]. The SSD removes the generation of traditional candidate boxes and uses the anchor box. The steps are as follow, 1. It generates the feature maps of different sizes in the image. 2. It uses a 3 $\times$ 3 convolution to process default box for each feature map. 3. It uses the IOU coefficient to predict the default box to match the anchor box in the training process.

C. YOU ONLY LOOK ONCE (YOLO)

The YOLO [21] was developed in 2016, it is a regression method based on deep learning which can predict multiple prediction candidate boxes and categories of convolutional neural networks at one time. It also can realize the end-to-end convolutional network target detection model. The biggest advantage of YOLO [21] is its fast calculation speed, it focuses on the entire image during the training and use phases, so it has better results for background detection, and its background error detection rate is half lower than the Fast R-CNN. The YOLOV2 [22] was developed in 2017, it [22] has made many improvements form The YOLO [21] which significantly improved the mAP and the computational speed. The YOLOV2 [22] used the batch normalization, high resolution classifier, convolutional with Anchor boxes, and dimension clusters to improve the mAP. The YOLOv3 [23] was developed in 2018, it [23] also made improvements in all parts of the YOLO [21]. The YOLOv3 [23] applies the Darknet as the backbone network, which has faster output frame rate than other object detection methods. The YOLOV4 [24] was developed in 2020, it made improvements in all parts of YOLOv3 [23] to ensure the speed while greatly improves the detection accuracy of the model, and reduces the hardware requirements. The YOLOV4 [23] uses the CSP-Darknet53 [24] as the backbone, the SPP [26] and the PAN [27] as the neck, and the YOLOv3 [23] as the head. In addition, the YOLOV4 [24] modified the state-of-the-art methods

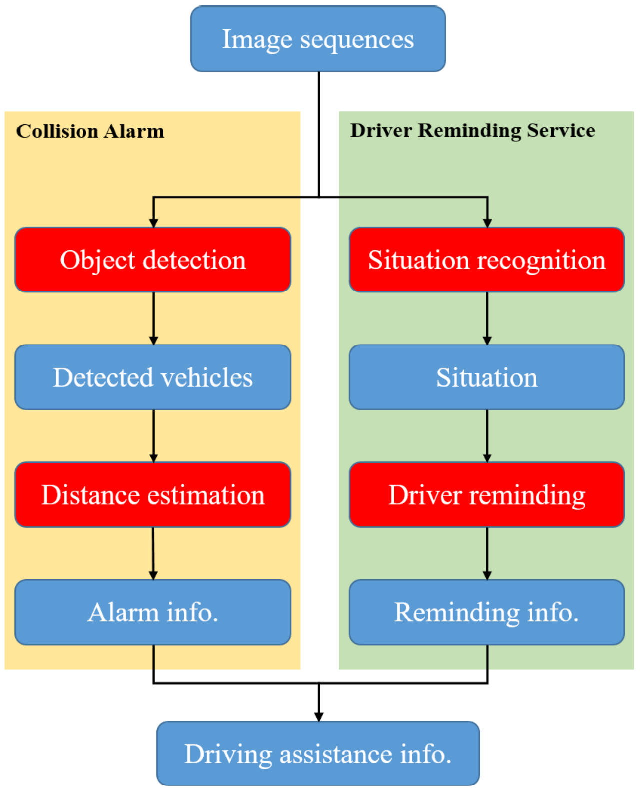


FIGURE 8. The architecture of the proposed system.

and make them more efficient and suitable for single GPU training.

D. LIGHTWEIGHT DEEP LEARNING MODEL

The requirements for accuracy and category are getting higher and higher with the development of technology. Not only the number of the network layer but also the memory and time requirements has increased, therefore, there are mainly two ways to train the lightweight deep learning model. The first way is to compress the pre-training network model and the second is training light weight network models directly. For example, the MobileNet [13] and the MobileNetV2 [28] apply the depth wise convolution in the training lightweight network model. As mentioned above, this study applied the lightweight deep learning model for vehicle detection.

III. PROPOSED SYSTEM

In this section, we are going to describe our proposed system, including system architecture, light weight object detection model, object distance estimation method, and situation recognition method.

A. SYSTEM ARCHITECTURE

Our proposed intelligent driving assistance system includes two functionalities, collision alarm and driver reminding service. The collision alarm was implemented based on the lightweight deep learning model and the distance estimation method; the driver reminding service was implemented

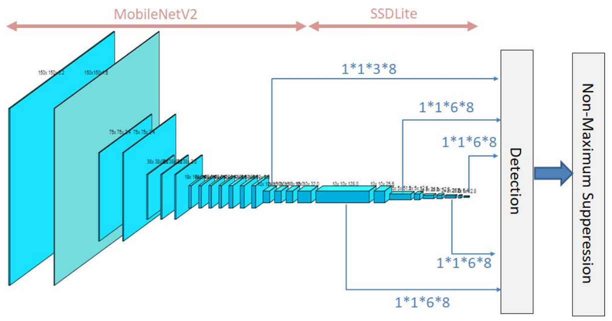


FIGURE 9. Architecture diagram of model in this study.

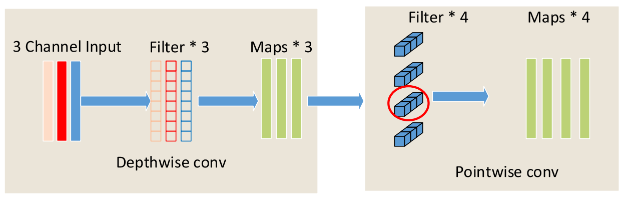


FIGURE 10. The architecture diagram of depth wise separable convolution.

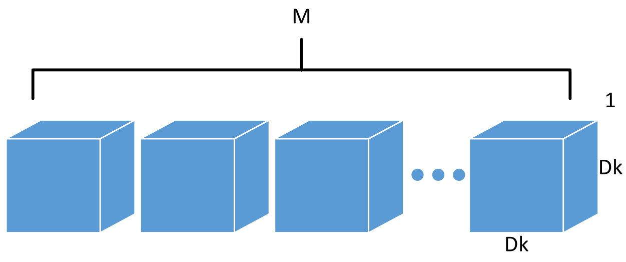


FIGURE 11. The diagram of depth wise convolutional.

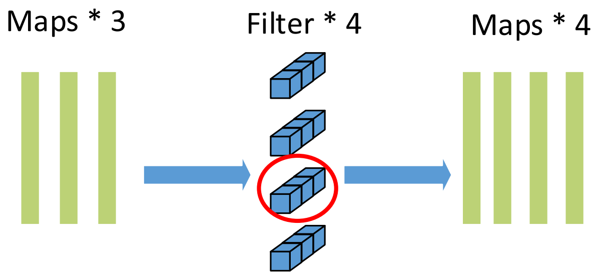


FIGURE 12. The diagram of pointwise convolution.

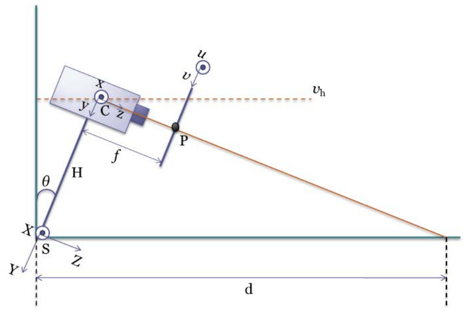


FIGURE 13. The relation between the HDR camera and objects.

through the situation recognition method. The architecture of our proposed system is shown as FIGURE 8. As the image sequences captured from cameras continuously, the system will process immediately and feedback to the driver. The processing details of the system functionality will be illustrated in the following sub-sections.

B. LIGHTWEIGHT OBJECT DETECTION MODEL

Since the driver assistance systems are usually required to be implemented on the in-car embedded platforms with limited computing resources, this study adopted lightweight deep learning based object detection method based on the MobileNet framework.

The MobileNetV2 [28] is implemented based on the following techniques: inverted residual, the linear bottlenecks, and the depth wise separable convolutions, which balances accuracy, model parameters and computational time. As shown in FIGURE 9, the adopted lightweight object detection model integrate the concepts of MobileNetV2 [28] with the SSD [19] to propose a lightweight convolutional neural network. The descriptions of the MobileNetV2[28] techniques are as follows:

Inverted residuals: When the image was input into the model, it makes the feature map larger by applying 1×1 convolution, then it uses the depth wise convolution for the convolution calculation and makes the feature map smaller by applying 1×1 convolution.

Linear bottlenecks: Because the linear bottlenecks has better result than nonlinear bottlenecks after using the depth wise convolution, it applied the ReLU6.

Depth wise separable convolution: FIGURE 10 is the architecture diagram of the depth wise separable convolution. There are two parts in the depth wise separable convolution which are the depth wise convolution and the pointwise convolution. The computational cost of convolutional neural networks can be reduced through the depth wise separable convolution.

The diagram of the depth wise convolutional is shown as FIGURE 11. First, if there are M channels, it creates a Dk $\times$ Dk $\times$ 1 filter for each channel of the input layer, then the convolution is performed on all data in the layer.

The diagram of pointwise convolutional is shown as FIGURE 12. The pointwise convolutional is very similar to the normal conventional convolution, but the difference is that the size of the convolution kernel is $1 \times 1 \times$ M which is the depth of the pervious layer. Therefore, the convolution performs a weighted combination of the depth direction filter and previous map to generate a new feature map. The amount of the Feature maps should be the same as the Filters.

C. OBJECT DISTANCE ESTIMATION METHOD

Based on the inference result of the lightweight object detection model, we can obtain the vehicles position in the image 2D coordinate system, represented as (u, v). In this subsection, we will talk about how we project the positions from the image 2D coordinate system to real world distance.

FIGURE 13 shows the relation between the HDR camera and objects. There are three plane coordinates which are (X, Y, Z), (x, y, z) and (u, v) in the position of S, C and P. (u, v) is the coordinate in the video. Point P is the center point of the object image that represents by (u_0, v_0) in FIGURE 13. Point S is the position of the vehicle and point C is the position of the camera.

v_h represents the position of the horizon in the driving image, which can be calculated by formula (1).

$$v_h = v_0 - \alpha \tan\theta \tag{1}$$

For formula (2), t_{pu} is the actual horizontal length of each pixel in the video; t_{pv} is the actual vertical length of each pixel in the video.

$$\alpha = \frac{f}{t_{pv}} \approx \frac{f}{t_{pu}} \tag{2}$$

The plane coordinate system (u, v) of point P is converted into the three-dimensional space coordinate system (x, y, z) at the position of point C by formula (3). The $\alpha\frac{x}{z}$ and $\alpha\frac{y}{z}$ are the offset.

$$\begin{cases} u=u_0+\alpha\frac{x}{z} \\ v=v_0+\alpha\frac{y}{z} \end{cases} \tag{3}$$

Substituting formula (1) into formula (3) becomes the following formula (4)

$$\frac{v - v_h}{\alpha} = \frac{y}{z} + \tan\theta \tag{4}$$

Then convert formula (4) from (x, y, z) coordinate system to (X, Y, Z) coordinate system, where H is the distance from point C to point S in FIGURE 13. The following formula (5) is obtained.

$$\frac{v - v_h}{\alpha} = \frac{Y + H}{Z} + \tan\theta \tag{5}$$

Now let $(X, Y, Z) = (X, -\mathrm{d}\sin\theta, \mathrm{d}\cos\theta)$, substituting it into formula (5) and simplifying to get formula (6).

$$\frac{v - v_h}{\alpha} = \frac{H}{d\cos\theta} \tag{6}$$

d is the real distance between P and the camera. The actual distance d can be estimated by formula (7).

$$d = \begin{cases} \dfrac{\lambda}{(\mathrm{v} - \mathrm{v}_h)} & if\ v < v_h \\ \infty & if\ v < v_h, \end{cases} \quad \text{where } \lambda = \frac{\mathrm{H}\alpha}{\cos\theta} \tag{7}$$

Then substitute the two point v_1, v_2 in the $(\mathrm{u}_0, \mathrm{v}_0)$ coordinate system into formula (5) and subtract them to correction formula (8) to find λ.

$$\lambda = \frac{d_1 - d_2}{\left(\frac{1}{v_1 - v_h} - \frac{1}{v_2 - v_h}\right)} \tag{8}$$

Finally, formula (9) can be used to find the distance.

$$\mathrm{d}_1 - \mathrm{d}_2 = \lambda\left(\frac{1}{v_1 - v_h} - \frac{1}{v_2 - v_h}\right) \tag{9}$$

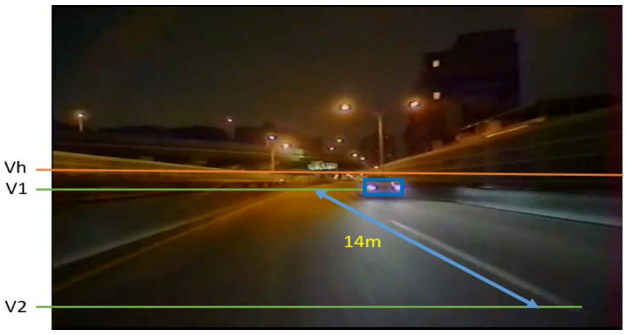


FIGURE 14. Schematic diagram of the object distance estimation.

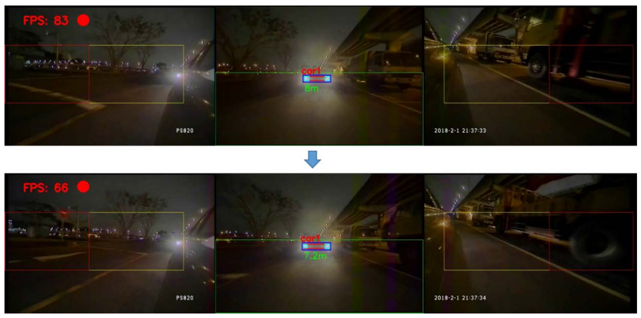


FIGURE 15. Example of warning an approaching vehicle.

FIGURE 14 is the schematic diagram of the image distance estimation. The correction formula (8) and the actual distance between two points in the driving video is adopted to find λ. In Taiwan, the lane line segment is four meters long and six meters apart. Points v_1 and v_2 are the length of the two lanes plus a distance. Once we obtain the actual distance between the two points in real world (14 meters), λ and v_h are able to be obtained, they can be applied in formula (7) to estimate the distance between the vehicle and the objects.

After estimating the distance between vehicles from formula (9), if the distance between the front and the back vehicles is less than 10 meters, formula (10) will then be applied to obtain the distance variation between t and $t - 1$. Our method will warn the driver that a car is approaching when $\delta \mathrm{d}^{t'}$ is negative. FIGURE 15 shows an example of warning an approaching vehicle. From the image above, we can see that the distance was 8m, and it became 7.2m in the next time unit, which made the color of alarm sign turn into red.

$$\delta \mathrm{d}^{t'} = d^t - d^{t-1} \tag{10}$$

D. SITUATION RECOGNITION METHOD

In this study, the proposed system not only raises the alarm when a vehicle approaches, but also reminds the drivers to turn on the headlights or wipers as the situation needed. Such function was implemented through the sky and weather situation recognition method, the implementation details will be described as following. FIGURE 16 is the flowchart of the sky and weather situation recognition method.

Improved from our previous research [29], we come up with a more robust method. In this method, five situations are defined which are raining in daytime, cloudy, sunny, night,

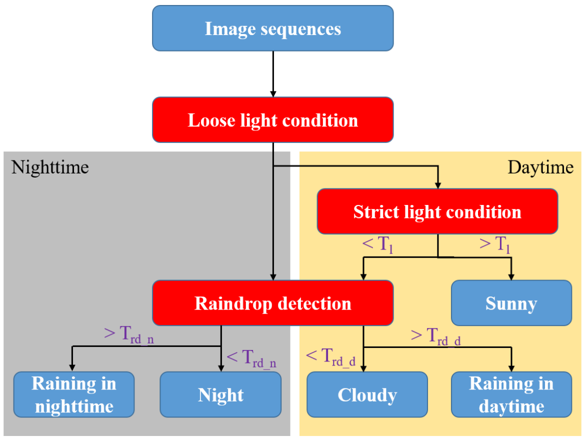


FIGURE 16. Flowchart of the sky and weather situation recognition method.

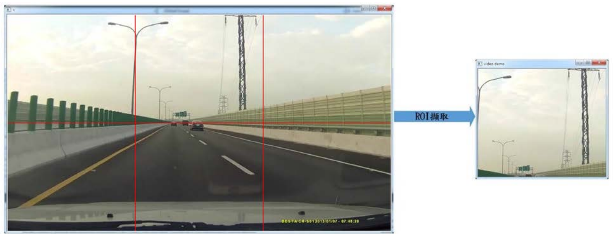


FIGURE 17. The diagram of the ROI.

and raining in nighttime. Raining in daytime, cloudy, and sunny are group into "daytime" while night and raining in nighttime are group into "nighttime". The current situation will first be roughly classified into "daytime" or "nighttime" through loose light condition, next, the specific situations will be determined through strict light condition and raindrop detection results. As the situation is determined, the drivers will perceive the reminding from the system asking to do actions according to current situations such as turn on the head lights, wipers, or fog lamps.

A front-view image from image sequences will be captured first, the sample image is shown as the left side of FIGURE 17, an RoI will be located and converted from RGB color space into HSV color space to get the fixed pixel of brightness form. Suppose we cut the image into 6 parts evenly, the RoI in our proposed method was defined as the top center part of the image, the captured image is shown as the right side of FIGURE 17.

The RoI part of the image will first be determined as "daytime" or "nighttime" through the loose light threshold, the threshold was defined as 90 in our research.

If the RoI part passes the threshold, it will be determined as "daytime", then a strict light condition will be applied to decide if it is sunny or not, the strict light condition T_l was defined as 200 in our research. If the condition is not determined as sunny, the raindrop detection method will be applied to obtain the raindrop amount, if the number of raindrops is larger than the defined threshold, the condition will be determined as raining in daytime, otherwise it will be

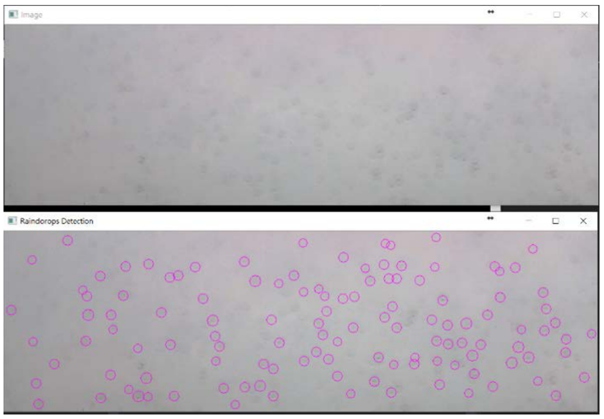


FIGURE 18. The diagram of the drop detection.

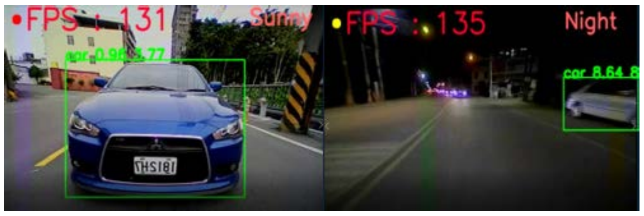


FIGURE 19. The diagram of the day and night.

determined as cloudy, the raindrop amount threshold T_{rd_d} in our research was defined as 100.

If the RoI part does not pass the threshold, it will be determined as "nighttime", the raindrop detection method will then be directly applied to obtain the raindrop amount. Although in the "daytime", we defined 100 as the condition for deciding it was raining or not, in the "nighttime", we defined 20 as the raining condition threshold T_{rd_n}. One of the reasons why we decreased the raining condition threshold is that the raindrops at nighttime is more difficult to be detected, as a result, it might be actually more raindrops than we observed, which indicates that it rains heavier than we know. Another reason is that more serious accidents happens in the nighttime rather than the daytime, especially in rainy days, therefore, the reminder of the wipers is more important during nighttime. The thresholds defined in this method were observed from the collected 19,798 images, including 13,488 daytime images and 6,310 nighttime images, which can be trusted that the conditions can make reliable decisions.

The raindrop detection, which adopted in our research for rainy determination, was implemented through AdaBoost classifier [30]. We collected raindrop sample images on the front window of the vehicles, extracted the Haar-like features of it, and trained a classifier through features from the sample images. The detection result of our proposed method is shown as FIGURE 18, each purple circle represents a detected raindrop, from the result, it is obvious that the raindrops were detected successfully.

FIGURE 19 is a diagram of day and night. The left side of the image is the diagram of the day and the diagram of the night is at the right side. Day or night scene is determined and shown on the upper right corner.

TABLE 1. Specifications of the desktop training platform.

Type	Object	Introduction
Hardware	CPU	Intel® Core™ i7-7700K CPU @ 4.20GHz × 8
	GPU	GeForce® GTX 1080 Ti
	RAM	DDR4 24GB
Software	OS	Ubuntu 16.04.4 LTS (Xenial Xerus) 64bit
	Programming Language	Python2.7
	Framework	Caffe

IV. EXPERIMENTS

In this section, we are going to illustrate the details of the experiments of our proposed system. First, the way how we collect and label the data will be explained, the data processing and augmentation for generating more reliable training samples will also be mentioned. Next, the evaluation indicators and their calculation will be noted. Finally, the comparison and the evaluation of model size, object detection performance, distance estimation performance, and situation recognition performance will be shown.

A. EXPERIMENTAL ENVIRONMENT

The experimental environments of this study applied NVIDIA GeForce GTX1080Ti graphics card as the training model. The experiments were conducted on PC and NVIDIA Jetson TX2 embedded platform. The programming language of the system proposed in the experiment is Python2 and the framework of deep learning is the Caffe framework using C/C++ language. The specifications of the desktop training platform for the lightweight CNN models are shown in TABLE 1, and the trained models are implemented on both desktop PC and NVIDIA JETSON TX2.

TABLE 2 shows the specifications of NVIDIA Jetson TX2. NVIDIA Jetson TX2 is a compact and energy saving embedded computing platform. It is suitable for applications that require high computing performance in a low energy environment.

NVIDIA Jetson TX2 is an embedded device with limited GPU computational resources, thus we applied this platform in the performance evaluation during testing process.

The training samples of this study were self-labeled from the sequences recorded from the cameras mounted on four different directions, for recording the vehicles and pedestrians in the front, behind, left, and right of the car in Taiwan driving environments under different situations with the car speed between 40 to 50 kilometers per hour.

We came up with a sample labeling rule, if the objects are occluded but their major parts and the appearance are still recognizable, those objects should be labeled; otherwise, if the objects are nearly blocked or covered by other objects, those objects should not be labeled. If the object is too small that it becomes blurry when we resize it to the model's input image size, those samples will be determined as bad data and should be filtered before model training. FIGURE 20 is an example of the data with bad quality.

TABLE 2. NVIDIA jetson TX2 specifications.

Type	Object	Introduction
Hardware	CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2
	GPU	NVIDIA Pascal™, 256 CUDA cores
	RAM	DDR4 8 GB 128 bit
Software	OS	Ubuntu 16.04.4 LTS (Xenial Xerus) 64bit
	Programming Language	Python2.7
	Framework	Caffe

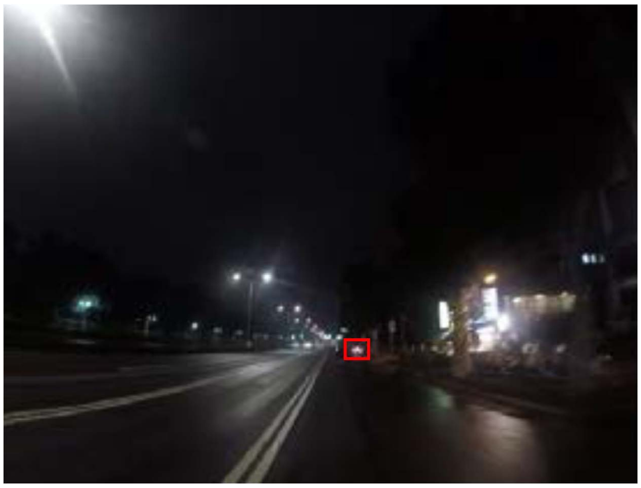


FIGURE 20. The example of the bad quality data.

FIGURE 21 shows an example of better quality data which is a clearer image with bigger vehicle samples. The training data were collected through LabelImg [31], the feature boxes and classes are able to be label respectively, shown as FIGURE 22.

After collecting the training data, we applied the data augmentation methods to increase the amount of the data for training a more robust model, the methods are shown as following.

1. Rotation: The images are rotated randomly.
2. Flip: The images are flipped horizontally or vertically.
3. Zoom: The images are zoomed in or zoomed out in equal proportions.
4. Shift: The images are changed by applying a random method to specify the range and step length of the shift, and move in the horizontal or vertical direction.
5. Scale: The images are enlarged or reduced according to the specified scale.
6. Contrast: In the HSV color space of the image, it changes the saturation and the brightness, maintains the hue, and performs exponential calculations on the components of each pixel to increase the brightness change.
7. Noise: The salt and pepper noise and Gaussian noise are added to the image.

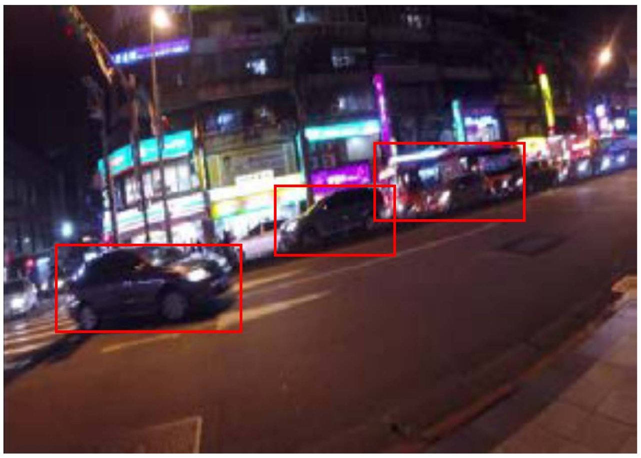


FIGURE 21. The example of data with better quality.

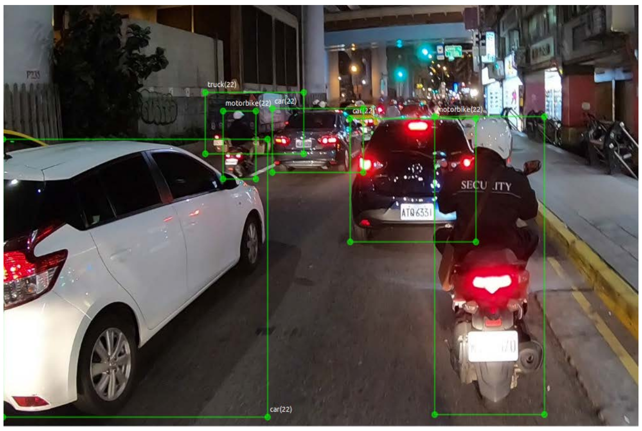


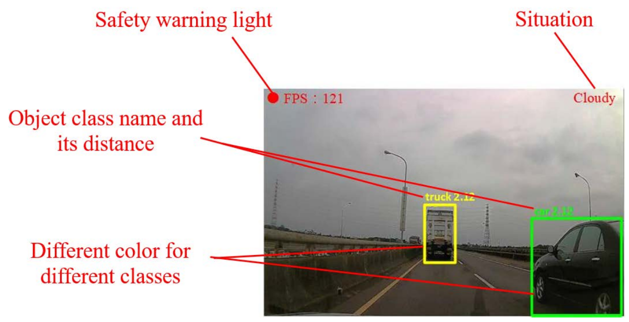
FIGURE 22. The labeled training set image.

B. EVALUTION INDICATORS

The confusion matrix and mean average precision (mAP) were adopted for the detection and recognition evaluation indicator of this study. The mAP is a popular evaluation metric for object detection. The localization determines the location of an instance and classification shows the target class.

There are four parameters in the confusion matrix: true positive (TP), which indicates that both actual situation and detection results are positive samples; true negative (TN), which indicates that both actual situation and detection results are negative samples; false positive (FP), which indicates that the actual situation is a negative sample but the detection result is a positive sample; and false negative (FN), which indicates that the actual situation is a positive sample but is detected as a negative sample. In formula (11), accuracy is the ratio of the correct detection of all samples. In formula (12), precision is the ratio of positive samples of events in all detected positive samples. In formula (13), recall is the ratio of the detected positive samples in the positive samples of all events.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$


Safety warning light
Situation
Object class name and its distance
Different color for different classes
FPS : 121
Cloudy
truck 2.12
FIGURE 23. The information provided in the visualization result.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

C. EXPERIMENTAL ANALYSIS

In this subsection, we have done the comparison of our adopted deep learning model and the other object detection models; furthermore, the performance evaluation of each function module in our system were completed. In addition, the visualization results were provided, shown as FIGURE 23 with the information included, the distance of the vehicles were measured in meters and the situation result only appears in front-view images.

1) OBJECT DETECTION MODEL SIZE COMPARISON

TABLE 3 shows the model size comparative results between different object detection models. We can see that the sizes of the models based on MobileNet framework are all smaller than the others, which are more suitable for the implementations of hardware with limited computing resources, such as embedded platforms and mobile devices.

2) OBJECT DETECTION PERFORMANCE COMPARISON ON PC AND EMBEDDED PLATFORM

TABLE 4 shows the comparative results of different CNN models' mAPs, and the corresponding computational efficiency implemented on the desktop PC with and without GPU computational resources, respectively. All the models have been trained until converged, we can found that it is a trade of between accuracy and computational efficiency. As the object detection accuracy will affect the performance of the overall system significantly, MobileNetV2-SSD[28], which obtain the highest mAP (60.72) among the lightweight deep learning model had chosen as the object detection model for the proposed system. TABLE 5 shows the parameters of the related comparison method, including batch sizes, optimizers, weight decay and base learning rates.

3) OBJECT DETECTION RESULTS ON DIFFERENT VIEWPOINTS

Refers to the conclusion in the above subsection, we chose MobileNetV2-SSD [28] as our object detection model in

TABLE 3. Model size comparison.

Model	Size
VGG16-Faster-RCNN[18]	528MB
VGG16-SSD[19]	98.2MB
Yolo[21]	365.4MB
YoloV3[23]	983.7MB
YoloV3-Tiny[32]	33.97M
MobileNet-SSD[13]	23.3MB
MobileNetV2-SSD[28]	12.8MB
MobileNetV3-SSD[33]	11.4MB

TABLE 4. Comparative results of models map and computational times on PC with/without GPU.

Model	mAP	FPS (with GPU)	FPS (without GPU)
Yolo[21]	21.24	109	6.4
YoloV3[23]	36.23	42	2.5
YoloV3-Tiny[32]	36.03	336	43.35
MobileNet-SSD[13]	49.22	47	2.85
MobileNetV2-SSD[28]	60.72	46	2.8
MobilNetV3-SSD[33]	13.9	2	0.12

TABLE 5. Parameters of the related comparison method.

Model	batch size	optimizer	weight decay	base learning rate
Yolo[21]	64	momentum	0.0005	0.0005
YoloV3[23]	64	momentum	0.0005	0.001
YoloV3-Tiny[32]	64	momentum	0.0005	0.001
MobileNet-SSD[13]	24	adam	0.00005	0.0005
MobileNetV2-SSD[28]	24	adam	0.00005	0.0005
MobilNetV3-SSD[33]	32	sgd	0.0005	0.001

our proposed intelligent driving assistance system. TABLE 6 shows the object detection performance of our proposed system on the desktop PC platform with GPU. The objects appear in front of or behind the host car are more than those in the left and right, as a result, the detection time for the images captured from the front and rear sides will be slightly longer than those captured from the left and right sides. As the shape appearances of the objects appear on left and right sides might be more complicated, the detection accuracy rate is lower than those from the front and rear sides.

FIGURE 24 demonstrates the detection results and the corresponding FPS of our proposed system on the desktop PC with GPU computing resources in daytime. The snapshots in FIGURE 24 show the vehicle object detection results of the host car from different viewpoints, FIGURE 24(A) shows the result from the front, the weather is cloudy, FIGURE 24(B) shows the result from the back, FIGURE 24(C) shows the result from the left, and FIGURE 24 (D) shows the result from the right. We can see that the proposed system can achieve the computational efficiency with an average of 47 frames per second (FPS) on the desktop PC with GPU computing resources.

FIGURE 25 demonstrates the detection results and the corresponding FPS of our proposed system implemented on the desktop PC with GPU computing resources in night time. FIGURE 25(A) shows the result from the front, FIGURE 25(B) shows the result from the back, FIGURE 25(C) shows the result from the left, and FIGURE 25 (D) shows the result from the right.

TABLE 6. Object detection performance on PC with GPU.

Viewpoints	Total	Hit	Recall	Precision	Accuracy	FPS
Front side	20107	19912	99.0%	99.0%	99.0%	47
Rear side	20135	19900	98.8%	98.8%	98.8%	48
Left side	4213	3925	93.2%	93.2%	93.0%	46
Right side	4288	4129	96.3%	96.2%	96.2%	46

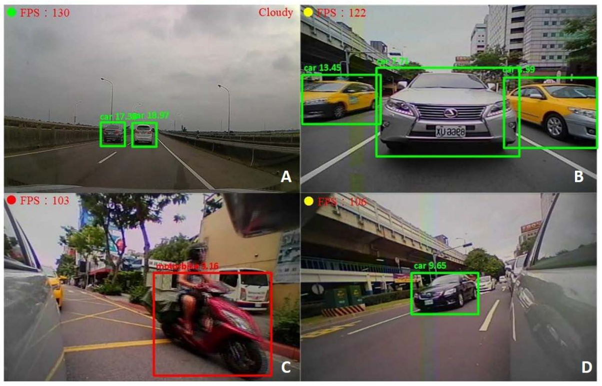


FIGURE 24. The results of object detection and FPS on the PC in daytime.

TABLE 7. Object detection performance on NVIDIA jetson TX2.

Viewpoints	Total	Hit	Recall	Precision	Accuracy	FPS
Front side	20107	19710	98.0%	98.0%	98.0%	3
Rear side	20135	19702	97.8%	97.8%	97.8%	3.2
Left side	4213	3917	93.0%	93.0%	92.9%	3.1
Right side	4288	4129	96.3%	96.2%	96.2%	3

TABLE 7 shows the object detection results of our proposed system on the NVIDIA Jetson TX2 embedded platform with limited CPU computing resources. As the complex computations causing the high loading for CPU, the computational efficiency decreased, although similar detection performance comparing to those of PC with GPU have obtained, proved that it can detect accurately with limited CPU computing resource.

FIGURE 26 shows the detection results of our proposed system implemented on the JETSON TX2 embedded platform with limited GPU computing resources in daytime. FIGURE 26(A) shows the result from the front, the weather is cloudy, FIGURE 26 shows the result from the back, FIGURE 26(C) shows the result from the left, and FIGURE 26 (D) shows the result from the right. FIGURE 27 shows the detection results of the proposed system implemented on the JETSON TX2 embedded platform with limited GPU computing resources at night.

The traffic objects detection accuracies of four surrounding viewpoints of the host car is higher than 90% in the desktop PC with GPU resources and NVIDIA Jetson TX2 embedded

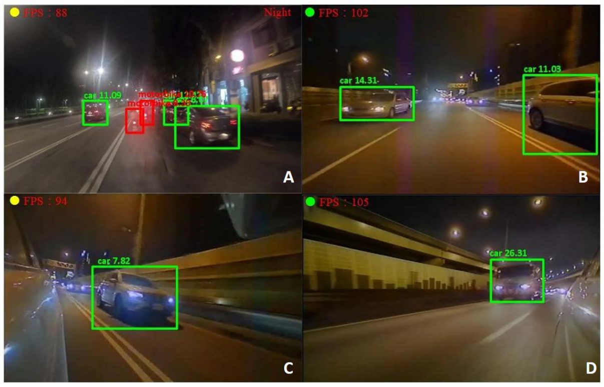


FIGURE 25. The results of object detection and FPS on the PC at night.

TABLE 8. Distance estimation evaluation results.

Viewpoints	Total	Hit	Precision	Accuracy
Front side	163	140	85.8%	85.9%
Rear side	142	125	88.0%	88.0%
Left side	113	90	79.6%	79.6%
Right side	106	85	80.2%	80.2%

TABLE 9. Situation recognition evaluation results.

Situations	Total samples	Hit	Precision	Accuracy
Cloudy	562	446	79.36%	79.36%
Sunny	493	396	80.32%	80.32%
Rainy Day	289	255	88.24%	88.24%
Nighttime	506	456	90.12%	90.12%
Rainy Nighttime	247	200	80.97%	80.97%
Total	2097	1753	83.60%	83.60%

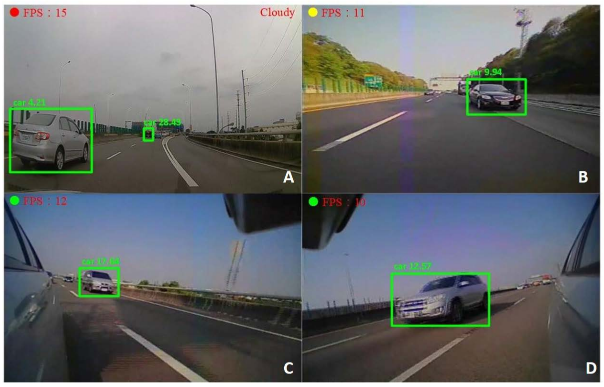


FIGURE 26. The results of the detection on Jetson TX2 in daytime.

platform. The network model trained by the experiment in this study is only 12.8MB. The adopted lightweight CNN models can be implemented on the embedded and mobile devices with limited GPU resources in the driving safety assistance system and the accuracy rate can still achieve satisfactory results and computational efficiency. Therefore, the adopted lightweight deep learning model can be applied on the embedded systems for practical and consumer car safety systems.

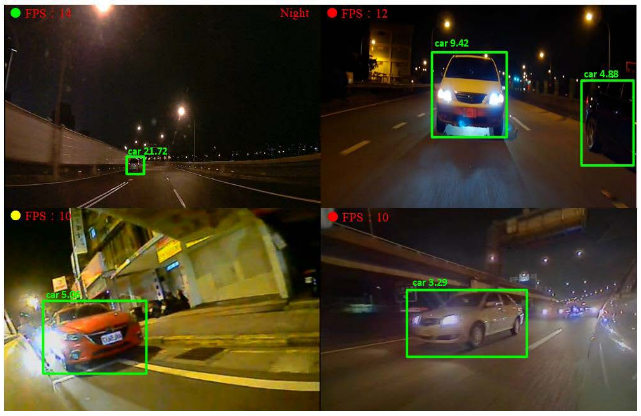


FIGURE 27. The results of the detection at night on Jetson TX2.

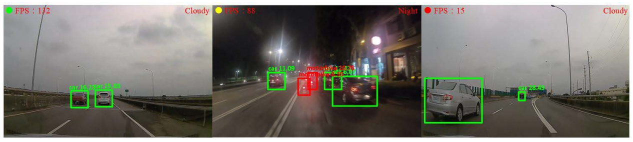


FIGURE 28. The results of the approaching vehicle warnings.

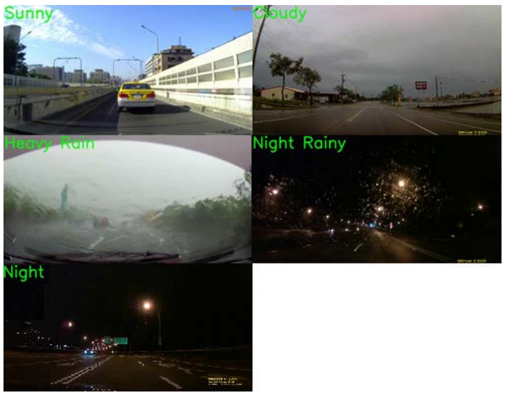


FIGURE 29. Sample results of the proposed situation recognition method.

4) DISTANCE ESTIMATION PERFORMANCE EVALUATION

TABLE 8 shows the experimental results of the proposed distance estimation method in our proposed system. We can see that the proposed method can provide detection accuracy on different viewpoints, including the front side, rear side, left side, and right side.

FIGURE 28 shows the results of the approaching vehicle warnings. If the vehicle behind is more than 10 meters away from the host vehicle, a green dot appears at the upper left corner which means it is safe for the host vehicle. If the vehicle behind is between 10 meters and 5 meters away from the host vehicle, a yellow dot appears at the upper left corner which is a warning for the host vehicle. If the vehicle behind is less than 5 meters away from the host vehicle, a red dot appears at the upper left corner which means it is dangerous for the host vehicle.

5) SITUATION RECOGNITION PERFORMANCE EVALUTION

TABLE 9 shows the experimental results of the proposed situation recognition method in our proposed system. We can see that the proposed method can provide satisfactory recognition accuracy in different situations for assisting the drivers to perceive the situations of driving environments.

As mentioned above, the total average recognition accuracy rate for the situations is about 83.60%. Among all weather situations, nighttime achieves the best accuracy rate, in the contrary, cloudy has the lowest accuracy rate. Although cloudy has a relatively low accuracy, it still reaches 80%. FIGURE 29 shows the results of the detection in different situations, including the sample results in sunny day, cloudy day, rainy day, rainy night, and nighttime, respectively.

V. CONCLUSION

The functionalities of our proposed system include collision alarm and driver reminding service. The collision alarm is implemented through lightweight CNN model and distance estimation method; the driver reminding service is implemented through situation recognition method. Refer to the experiment results, the proposed methods and the adopted model obtain sufficient computational efficiency and performance, especially the adopted CNN model size was smaller enough, which obtain higher probability to be inference through limited computing resource embedded system. In conclude, we proposed a computational efficient solution of driver assistance systems that can be implemented on the consumer embedded platforms for ADAS.

ACKNOWLEDGMENT

The authors would like to thank Chun-Ting Chen for her English editing.

REFERENCES

[1] W. W. Wierwille, R. J. Hanowski, J. M. Hankey, C. A. Kieliszewski, S. E. Lee, A. Medina, A. S. Keisler, and T. A. Dingus, ‘‘Identification and evaluaion of driver errors: Overview and recommendations,’’ U.S. Dept. Transp., Washington, DC, USA, Tech. Rep. FHWA-RD-02-003, 2002.

[2] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, ‘‘Pedestrian recognition using high-definition LiDAR,’’ in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 405–410.

[3] E. Lozano-Monasor, M. T. López, A. Fernández-Caballero, and F. Vigo-Bustos, ‘‘Facial expression recognition from webcam based on active shape models and support vector machines,’’ in *Proc. Int. Workshop Ambient Assist. Living*. Cham, Switzerland: Springer, 2014, pp. 147–154.

[4] P. Viola and M. Jones, ‘‘Rapid object detection using a boosted cascade of simple features,’’ in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2001, pp. I-511–I-518.

[5] N. Dalal and B. Triggs, ‘‘Histograms of oriented gradients for human detection,’’ in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893.

[6] Q. Sun, H. Zhang, Z. Li, C. Wang, and K. Du, ‘‘ADAS acceptability improvement based on self-learning of individual driving characteristics: A case study of lane change warning system,’’ *IEEE Access*, vol. 7, pp. 81370–81381, 2019.

[7] *Supervised Learning? Enhanced Learning? If You Don't Understand, You Must Read This Introductory Machine Learning Term Explanation!* Accessed: Jul. 25, 2022. [Online]. Available: https://www.inside.com.tw/article/9945-machine-learning

[8] *Artificial Intelligence, Machine Learning, Deep Learning to Solve Puzzles*. Accessed: Jul. 25, 2022. [Online]. Available: https://tuna.to/artificial-intelligence-4dbb43229124

[9] *The Foundation of Machine Learning and Deep Learning*. Accessed: Jul. 25, 2022. [Online]. Available: https://ithelp.ithome.com.tw/articles/10192890

[10] *3 Minutes to Understand What Deep Learning is in*. Accessed: Jul. 25, 2022. [Online]. Available: https://panx.asia/archives/53209

[11] F. Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, and L. Yang, ‘‘Where does AlphaGo go: From church-Turing thesis to AlphaGo thesis and beyond,’’ *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 2, pp. 113–120, Apr. 2016.

[12] *Principle and Application of Deep Learning*. Accessed: Jul. 25, 2022. [Online]. Available: http://www.cc.ntu.edu.tw/chinese/epaper/0038/20160920_3805.html

[13] 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,’’ 2017, *arXiv:1704.04861*.

[14] 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.*, Jun. 2014, pp. 580–587.

[15] J. R. R. Uijlings, E. A. Van De Sande Koen, G. Theo, and W. M. S. Arnold, ‘‘Selective search for object recognition,’’ *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Sep. 2013.

[16] T. Joachims, ‘‘Text categorization with support vector machines: Learning with many relevant features,’’ in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1998.

[17] R. Girshick, ‘‘Fast R-CNN,’’ in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[18] 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. 91–99.

[19] 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.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[20] D. Frossard. (2017). *VGG in TensorFlow*. [Online]. Available: https://www.cs.toronto.edu/frossard/post/vgg16

[21] J. 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. 779–788.

[22] J. Redmon and A. Farhadi, ‘‘YOLO9000: Better, faster, stronger,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[23] J. Redmon and A. Farhadi, ‘‘YOLOv3: An incremental improvement,’’ 2018, *arXiv:1804.02767*.

[24] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, ‘‘YOLOv4: Optimal speed and accuracy of object detection,’’ 2020, *arXiv:2004.10934*.

[25] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, ‘‘CSPNet: A new backbone that can enhance learning capability of CNN,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 390–391.

[26] K. He, X. Zhang, J. Sun, and S. Ren, ‘‘Spatial pyramid pooling in deep convolutional networks for visual recognition,’’ *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Jan. 2015.

[27] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, ‘‘Path aggregation network for instance segmentation,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘‘MobileNetV2: Inverted residuals and linear bottlenecks,’’ 2018, *arXiv:1801.04381*.

[29] K.-F. Lee, H.-Y. Liang, Y.-L. Chen, C.-W. Yu, and Y.-C. Chen, ‘‘On-road weather detection and analysis based on visual models,’’ in *Proc. 7th Int. Symp. Next Gener. Electron. (ISNE)*, May 2018, pp. 1–3, doi: 10.1109/ISNE.2018.8394716.

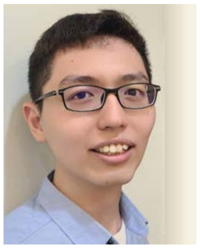
[30] K. F. Lee, Y. L. Chen, K. Y. Chen, C. W. Yu, and C. Y. Hsiao, ‘‘Surrounding view advanced driver assistance systems based on lightweight technology,’’ in *Proc. Int. Conf. Syst. Sci. Eng.*, 2020, p. 1090.

[31] *LabelImg is a Graphical Image Annotation Tool*. Accessed: Jul. 25, 2022. [Online]. Available: https://github.com/tzutalin/labelImg

[32] P. Adarsh, P. Rathi, and M. Kumar, ‘‘YOLO v3-Tiny: Object detection and recognition using one stage improved model,’’ in *Proc. 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2020, pp. 687–694.

[33] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, ‘‘Searching for MobileNetV3,’’ 2019, *arXiv:1905.02244*.

KO-FENG LEE was born in Kaohsiung, Taiwan, in May 1987. He received the B.S. and M.S. degrees in information engineering and computer science from Taichung Feng Chia University, Taichung, Taiwan, in July 2014, and the Ph.D. degree in computer science and information engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2021. His research interests include computer-aided learning, multimedia applications, mobile technology, ubiquitous learning, artificial intelligence, driver assistance, the IoT, and web technology.



XIU-ZHI CHEN was born in Kaohsiung, Taiwan, in 1995. He received the M.S. and B.S. degrees in computer science and information engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2017, where he is currently pursuing the Ph.D. degree in computer science and information engineering. He holds two Taiwan patents and one U.S. patent. His research interests include artificial intelligence, intelligent image analytics, intelligent biometrics, and intelligent transportation systems.

CHAO-WEI YU was born in Pingtung, Taiwan, in October 1985. He received the Ph.D. degree in computer science and information engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2018, where he is currently pursuing the Ph.D. (Researcher) degree with the Department of Computer Science and Information Engineering. His research interests include intelligent image, driver assistance, and intelligent IoT cloud-edge computing.

KAI-YI CHIN (Member, IEEE) received the Ph.D. degree from the Department of Information Engineering and Computer Science, Feng Chia University, in 2011. She is currently an Associate Professor with the School of Big Data Management, Soochow University, Taipei City, Taiwan. She has worked as an Associate Professor with the Department of Digital Humanities, Aletheia University, from August 2015 to July 2020, where she also an Assistant Professor, from August 2011 to July 2015. She has published over 50 papers in international journals and conferences and participated in many international academic activities. Her research interests include computer-aided learning, multimedia applications, mobile technology, ubiquitous learning, augmented reality, and wearable technology.

YIH-CHEN WANG was born in New Taipei City, Taiwan, in December 1997. She received the B.S. degree in computer science and information engineering from Yuan Ze University, Taoyuan, Taiwan, in 2020. She is currently pursuing the master's degree in computer science and information engineering with the National Taipei University of Technology, Taipei, Taiwan. Her research interests include artificial intelligence, data science, and intelligent image.

CHIA-YU HSIAO was born in Taoyuan, Taiwan, in July 1987. She received the M.S. degree in computer science and information engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2019. Her research interests include intelligent image, linux device driver assistance, and artificial intelligent.

YEN-LIN CHEN (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical and control engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2006, respectively. From February 2007 to July 2009, he was an Assistant Professor at the Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan. From August 2009 to January 2012, he was an Assistant Professor at the Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan, where he was an Associate Professor from February 2012 to July 2015, and since August 2015, he has been a Full Professor with the National Taipei University of Technology. His research interests include artificial intelligence, intelligent image analytics, embedded systems, pattern recognition, intelligent vehicles, and intelligent transportation systems. His research results have been published on over 100 journals and conference papers. He is a fellow of the IET and a member of ACM, IAPR, and IEICE.