Fast, Accurate Vehicle Detection and Distance Estimation

QuanMeng Ma[1], Guang Jiang[1*], DianZhi Lai[1], Hua cui[2], Huansheng Song[2]
[1]School of Telecommunication Engineering, Xidian University, Xi'an, China.
[2]School of information engineering, Chang'an University. Xi'an, China.
[e-mail: gjiang@mail.xidian.edu.cn]
*Corresponding author: Guang Jiang

Received June 5, 2019; revised August 10, 2019; accepted September 15, 2019; published February 29, 2020

Abstract

A large number of people suffered from traffic accidents each year, so people pay more attention to traffic safety. However, the traditional methods use laser sensors to calculate the vehicle distance at a very high cost. In this paper, we propose a method based on deep learning to calculate the vehicle distance with a monocular camera. Our method is inexpensive and quite convenient to deploy on the mobile platforms. This paper makes two contributions. First, based on Light-Head RCNN, we propose a new vehicle detection framework called Light-Car Detection which can be used on the mobile platforms. Second, the planar homography of projective geometry is used to calculate the distance between the camera and the vehicles ahead. The results show that our detection system achieves 13FPS detection speed and 60.0% mAP on the Adreno 530 GPU of Samsung Galaxy S7, while only requires 7.1MB of storage space. Compared with the methods existed, the proposed method achieves a better performance.

Keywords: Light-Car Detection, Deep learning, vehicle distance, object detection, mobile platform.

http://doi.org/10.3837/tiis.2020.02.008 **ISSN : 1976-7277**

1. Introduction

In recent years, Unmanned and Assisted Driving have been developing dramatically. In driverless and assisted driving, obtaining the distance to the vehicle ahead accurately is a critical prerequisite for the normal driving of the car. In this field, various methods [1-9] have been proposed to calculate the distance between the driving vehicle and the vehicle ahead. There are roughly three tools that these methods rely on: traditional computer vision theory, laser sensors, and deep learning theory. The traditional computer vision based method [1, 2, 3, 7, 8, 9] detects the presence of the preceding vehicle and calculates the distance by means of vanishing point detection, vehicle segmentation, particle filtering and road detection. However, most of these methods can only calculate the distance of single vehicle in the image of complicated scene. These methods rely heavily on hand-crafted feature, making it difficult to precisely calculate the distance between the host and the vehicle ahead in the case of complicated vehicle conditions or chaotic backgrounds. Geiger *et al.* [4] made great progress in calculating distance, who first used DPM to detect the vehicle and then calculated the distance to the vehicle ahead through the calibrated camera model. In this way, it can obtain the position and distance information of multiple vehicles at the same time. However, it is difficult to accurately obtain the vehicle position information in different scenarios with this camera model, so the accuracy is not improved. In the radar sensor-based approach [6, 7, 10], Chen *et al.* proposed that we can use the connected car network to calculate the distance between vehicles [7]. Tsai *et al.* used the Integrated Lane Departure Warning System (LDWS) [6] and Forward Collision Warning System (FCWS) to calculate the distance. However, considering that this method is complicated and expensive, a wide range of popularization and application will not be worth the candle. Chen *et al.* [5] tried to apply CNN directly to the vehicle distance information, but the results show that its performance is not significantly better than the traditional methods. In the existing method, the method of using only picture information cannot calculate the vehicle distance stably and accurately, so it is improper to make application when high security is required, such as Advanced Driver Assistance System (ADAS). The others making use of laser are too expensive to get widely used, which motivates our approach in this paper.

This paper makes two contributions. First, based on Light-Head RCNN, we propose a new vehicle detection framework called Light-Car Detection which can be used on the mobile platforms. Second, the planar homography of projective geometry is used to calculate the distance between the camera and the vehicles ahead. Thanks to the development of deep learning, the accuracy of object detection has been improved notably. In this paper, an object detector based on convolutional neural network is proposed, which combines the homography transformation method to detect the vehicle and calculate the distance. In the implementation of the algorithm, we first trained a fast detector that can be applied to a variety of scenarios to determine the vehicle in a bounding box. Then, using the homography transformation between the plane and the image, the center point of the lower edge of the bounding box is projected onto the ground plane, so that the bounding box can be used as the scale for distance detection. But the detected bounding boxes are constantly shaking, so we use weighted NMS and object tracking to improve the stability and accuracy of the detection and calculation. Compared to traditional computer vision and deep learning methods, our method is faster, more accurate and suitable for a variety of scenarios, and achieve better results. The Mean Absolute Error (absolute difference between calculating distance and the truth distance) has been reduced by

three times compared to the methods of Geiger *et al.* and Chen *et al.* [4, 5], reaching 1.248m. As shown in **Fig. 1**, we can estimate the distance of the preceding car more accurately. In the figure, TDistance represents the true distance, and PDistance is the estimated distance. The implement steps of our proposed method are shown in **Fig. 2**. It contains a detection module, a calibration module and distance calculation module. In the following sections, we will describe each module in detail.

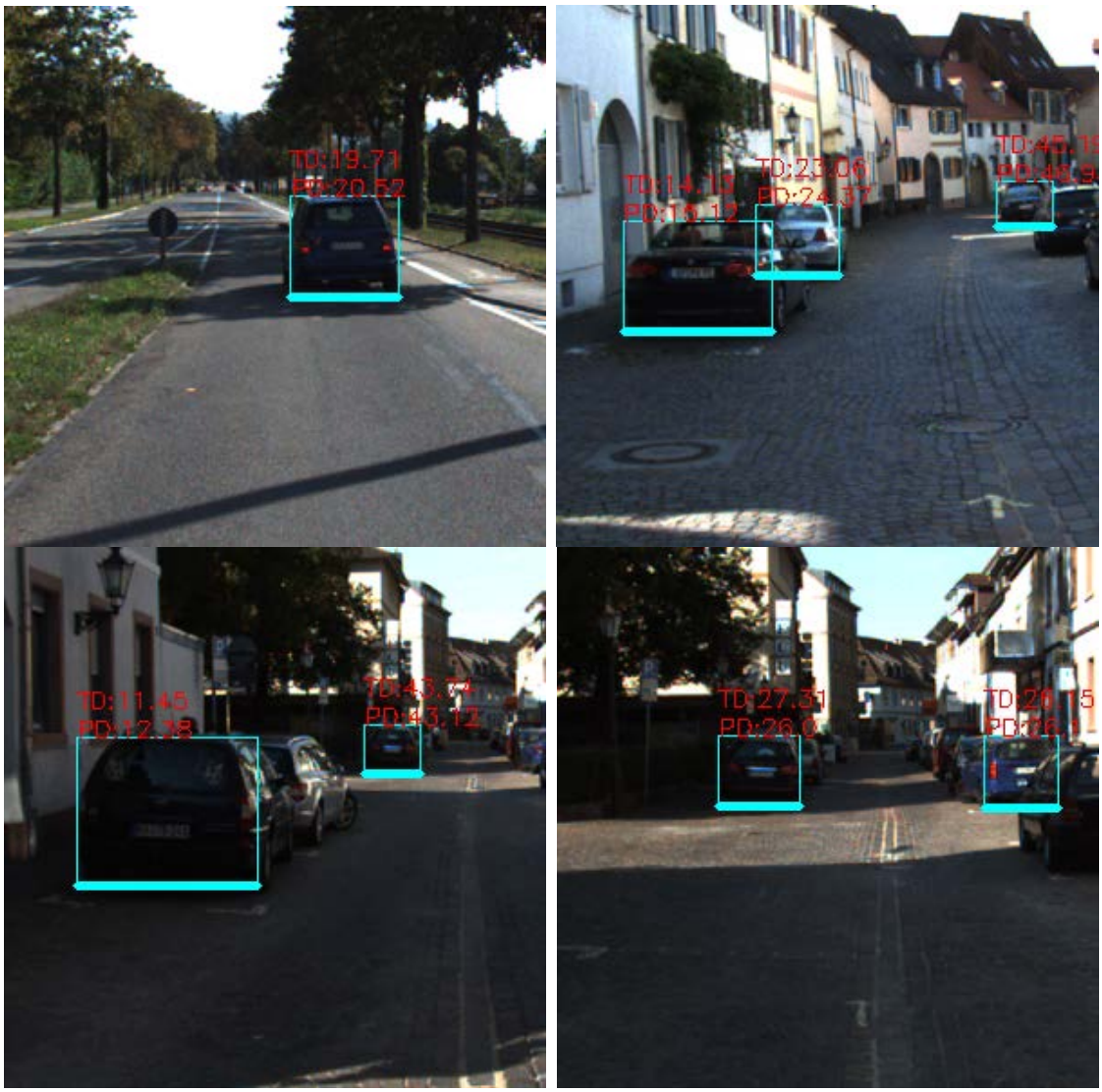

Fig. 1. Some results predicted by our method, TD represents the true distance, and PD is the estimated distance. The distances are in meters.

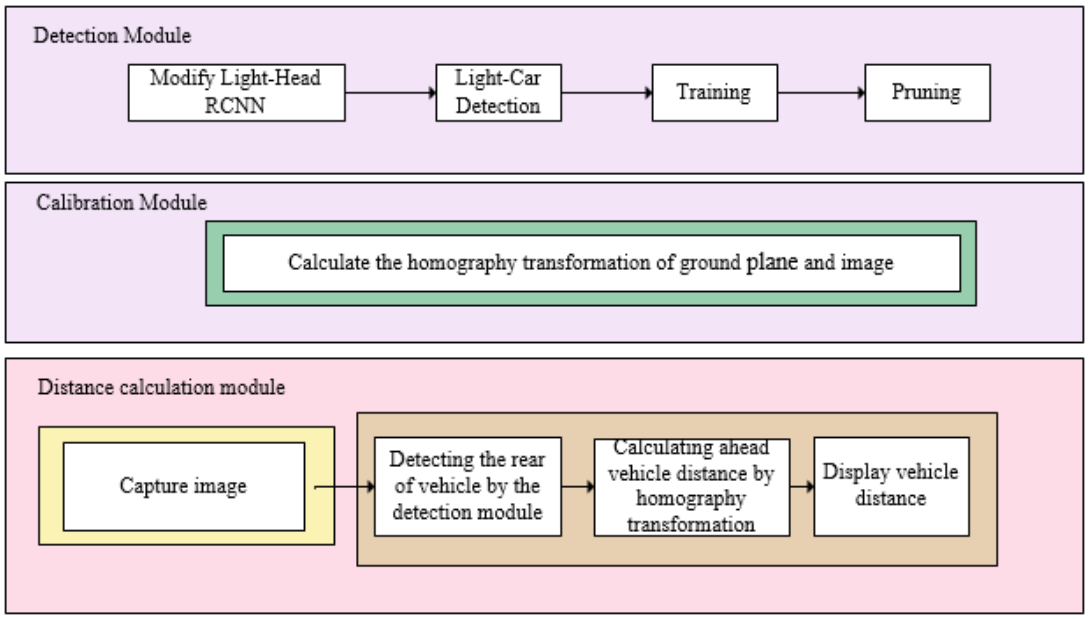

Fig. 2. The implementation steps of the proposed framework.

2. Vehicle Detection Method Selection

In order to estimate the distance and implement the application, we hope that the algorithm can consume a small amount of memory while ensuring accuracy in the shortest possible running time. The rise of deep convolutional neural networks has led to significant development of object detection. The methods of real-time object detection consists of direct regression detection and region-based detection. SSD [20], YOLOV2 [21], RetinaNet [22] and YOLOV3[23] are direct regression detection methods. SSD presents a better performance than YOLOV2 and RetinaNet, while it has almost the same performance with YOLOV3. R-FCN [24], Light-Head R-CNN [25] and Faster-RCNN [28] are region-based detection method, and Light-Head R-CNN has better results in terms of speed and accuracy than Faster-RCNN and R-FCN. Unfortunately, SSD, YOLOV3 and Light-Head R-CNN only detect runtime on a computer GPU, but the platform on which each experiment relies is different, so it is impossible to compare speed and accuracy objectively and fairly. To clarify which algorithm performs better, we designed the experiments discussed in Section 2.1.

2.1 Experimental configuration

2.1.1 Feature extractors

In order to enable the vehicle detection can obtain position information in real time, we use MobileNetV1 as a feature extractor for SSD, YOLOV3 and Light-Head R-CNN. MobileNetV1 is dedicated to effective inference in mobile vision applications. For the sake of discussion, we present an overview of MobilenetV1 in **Table 1**. Its building block is a deeply detachable convolution, which decomposes standard volume integration into deep convolution and 1×1 convolution, effectively reducing computational cost and number of parameters.

In Light-Head R-CNN, we need to select a layer for predicting region proposals. In our experiments, each convolution layer of the feature extractor can be used to predict region proposals. However, taking it into consideration that the depth of the MobileNetV1 network is not enough, we use the conv5_5 layer as shown in **Table 1** to predict region proposals.

Table 1. The network structure of MobileNetV1.

Name/Stride	Filter Shape	Inpute Size
Conv1 /s2	3×3×3×32	224×224×3
Conv2_1 /dw[1] /s1[2]	3×3×32	112×112×32
Conv2_1 /s1	1×1×32×64	112×112×32
Conv2_2 /dw /s2[3]	3×3×64	112×112×64
Conv2_2 /s1	1×1×64×128	56×56×64
Conv3_1 /dw /s1	3×3×128	56×56×128
Conv3_1 /s1	1×1×128×128	56×56×128
Conv3_2 /dw /s2	3×3×128	56×56×128
Conv3_2 /s1	1×1×128×256	28×28×128
Conv4_1 /dw /s1	3×3×256	28×28×256
Conv4_1 /s1	1×1×256×256	28×28×256
Conv4_2 /dw /s2	3×3×256	28×28×256
Conv4_2 /s1	1×1×256×512	14×14×256
Conv5_1 /dw /s1	3×3×512	14×14×512
Conv5_1 /s1	1×1×512×512	14×14×512
Conv5_2 /dw /s1	3×3×512	14×14×512
Conv5_2 /s1	1×1×512×512	14×14×512
Conv5_3 /dw /s1	3×3×512	14×14×512
Conv5_3 /s1	1×1×512×512	14×14×512
Conv5_4 /dw /s1	3×3×512	14×14×512
Conv5_4 /s1	1×1×512×512	14×14×512
Conv5_5 /dw /s1	3×3×512	14×14×512
Conv5_5 /s1	1×1×512×512	14×14×512
Conv5_6 /dw /s2	3×3×512	14×14×512
Conv5_6 /s1	1×1×512×1024	7×7×512
Conv6 /dw /s1	3×3×1024	7×7×1024
Conv6 /s1	1×1×1024×1024	7×7×1024
Avg Pool /s1	7×7×1024	7×7×1024
FC /s1	1024×1000	1×1×1024
Softmax /s1	classifier	1×1×1000

[1] /dw represents convolution type is depthwise convolution.

[2] /s1 represents the filter stride size is 1.

[3] /s2 represents the filter stride size is 2

2.1.2 Dataset configure

The Nexar2 dataset is a large collection of 50,000 images with bounding box annotations of the rear of vehicles from around the world in different lighting and weather conditions. The original image size is 1280 pixels wide and 720 pixels high. In the experiment, we only pay attention to the vehicles in the narrow field of view in front of the current vehicle, we crop a image of 300×300 pixels from the center of each original image. In this way, we can train the model with a smaller picture to get a shorter inference time.

2.1.3 Hyperparameter tuning

In the Nexar2 dataset, most of the images have very few object pixels. In order to improve the accuracy of small object detection, we add smaller anchors in the Light-Head R-CNN. Three aspect ratios {1:2,1:1,2:1} and five scales 16^2,32^2,64^2,128^2,256^2 are set to cover objects of different shapes. Since there are many proposals heavily overlapping with each other, non-maximum suppression (NMS) is used to reduce the number of proposals before feeding them into RoI prediction subnetwork for processing. We set the intersection-over-union (IoU) threshold to 0.7 for NMS. Then we assign anchors training labels based on their IoU ratios with ground-truth bounding boxes. If the anchor has IoU over 0.7 with any ground-truth box, it will be assigned a positive label, whereas if any anchor has IoU less than 0.3 with all ground-truth box, the label will be negative. We train with 2000 RoIs in each iteration as is described in detail in reference [28]. Simultaneously, we adopt online hard example mining (OHEM) [29] during training, and select 256 RoIs that have the highest loss to perform Backpropagation [30]. For SSD and YOLOV3, we use the configuration as described in Liu *et al.* [20] and Redmon *et al.* [23] respectively.

2.2 Results

In this section, we mainly evaluate the accuracy of the Light-Head R-CNN, SSD and YOLOV3 when applied to Nexar2 dataset. In the experiment we use Ubuntu16.04 operating system, Inter i5-7500 CPU, 8G memory and GTX1060 GPU training the network. For Light-Head R-CNN, the optimization method is SGD, and batch-size is 2. Learning rate is set to 0.01 for first 0.16M iterations (processing 1 image will be regarded as 1 iteration), and 0.001 for later 0.08M iterations. For SSD, the optimization method is RMSProp optimization with an initial learning rate of 0.0001. For YOLOV3, the optimization method is Adam optimization with an initial learning rate of 0.0001. We use mean Average Precision (mAP) to measure the network accuracy. For Light-Head R-CNN, we use 300 RoIs to measure the network accuracy. In order to accelerate the inference, we merge the convolutional and Batch-normalization [15] layers. For a fair and objective comparison, we use the mobile optimization library MACE to perform code migration and test the speed of SSD, YOLOV3, Light-Head R-CNN. Finally, the accuracy and inference time of the three networks are shown in **Table 2**, from which it can be found that the accuracy of Light-Head R-CNN significantly outperform SSD and YOLOV3 in nexar2 dataset. In addition, it is easier to increase the speed than to improve the accuracy, so we use Light-Head R-CNN in our task.

We analyzed the reason for the results. In general, low-resolution feature maps have larger receptive field and are easier to detect large objects, whereas high-resolution feature maps are applicative for detecting small objects. The SSD does not choose to use low-level features, but builds the pyramid from the high layers in the MobileNetV1 network and adds several new CNN layers instead. Thus it misses the opportunity to reuse the higher-resolution maps of the feature hierarchy, leading to instability. For YOLOV3, it dose not solve the problem of

foreground-background class imbalance very well. For Light-Head R-CNN, it uses RPN to rapidly reduce the number of candidate boxes. In the second classification stage, OHEM is used to keep balance between foreground and background, so the network is fully trained and its ability to detect small objects is stronger. Therefore, on the Nexar2 dataset which conforms to real driving situation, the SSD detection accuracy is weaker than that of Light-Head R-CNN.

Table 2. Performance comparison of Light-Head R-CNN and SSD on NEXAR2 validation set.

Model	Backbone	Test Size short/max edge	Speed(ms)	mAP
Light-Head RCNN	MobileNet	300/300	501[1]	62.1
SSD	MobileNet	300/300	293[1]	59.9
YOLOV3	MobileNet	300/300	200[1]	53.6
Light-Head RCNN	MobileNet	300/300	36.2[2]	62.1
SSD	MobileNet	300/300	18.7[2]	59.9
YOLOV3	MobileNet	300/300	15[2]	53.6

[1]CPU milliseconds on Intel Core i5-7500, 3.40 GHz

[1]GPU milliseconds on GTX1060

3. Light-Car Detection

The Light-Head R-CNN can meet real-time requirements on the GPU of computer, but its inference time on the CPU or GPU of mobile device is long so it is impossible to run in real-time. Consequently, we have to improve the Light-Head R-CNN network to make it run faster. To run the neural network on mobile devices, we need to reduce the storage and computation costs of neural network. Generally, there are two directions for decreasing computation costs and the number of parameters: the first is to use a network module with less parameters; the second is to prune the trained network and then retraining. In this paper, we use both two methods to reduce the number of network parameters. In section 3, we use bottleneck module to build the detection network. In section 4, we prune and retrain the trained detection network.

3.1 Thin feature maps

For a standard convolution, the structure is shown in **Fig. 3(A)**. Its cost is

$$w \cdot h \cdot C_{out} \cdot k \cdot k \cdot C_{in} \tag{1}$$

Its parameter size is

$$k \cdot k \cdot C_{out} \cdot C_{in} \tag{2}$$

Where w, h and C_{out} are the width, height, the number of output channels of output feature map respectively, k is the size of convolution kernel, C_{in} is the number of channels of input feature map.

In the Light-Head R-CNN, the thin feature maps structure is shown in **Fig. 3(B)**. It is called large separable convolution. In this structure, the size of input tensor is $w \times h \times C_{in}$, the size of output tensor is $w \times h \times C_{out}$, so the structure have the computational cost of

$$2 \cdot w \cdot h \cdot C_{mid} \cdot k \cdot C_{in} + 2 \cdot w \cdot h \cdot C_{out} \cdot k \cdot C_{mid} \tag{3}$$

Its parameter size is

$$2 \cdot k \cdot C_{mid} \cdot C_{out} + 2 \cdot k \cdot C_{in} \cdot C_{mid} \tag{4}$$

Where C_{mid} is the output channels of middle layer in large separable convolution module.

In this paper, we use bottleneck instead of large separable convolution, whose structure is shown in **Fig. 3(C)**. It has the computational cost of

$$w \cdot h \cdot C_{mid} \cdot C_{in} + w \cdot h \cdot C_{mid} \cdot k \cdot k + w \cdot h \cdot C_{out} \cdot C_{mid} \tag{5}$$

Its parameter size is

$$C_{in} \cdot C_{mid} + k \cdot k \cdot C_{mid} \cdot C_{mid} + C_{mid} \cdot C_{out} \tag{6}$$

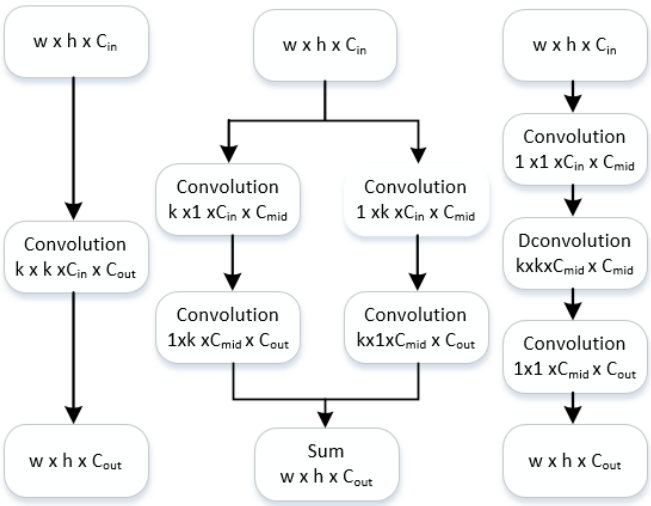

Fig. 3. Overview of three different structures. Dconvolution represents depthwise convolution.

We let k be 3, C_{out}, C_{in} and C_{mid} be 64, 512 and 64 respectively. In order to reduce the loss of information, we removed the relu layer behind the depthwise convolution layer. Compared to the large separable convolution and the standard convolution, the bottleneck dramatically reduces both parameters and computational costs as shown in **Table 3**. We also compare mAP between the large separable convolution and the bottleneck, and the results are shown in **Table 4**. The bottleneck is a more efficient model, but its accuracy is slightly low.

Table 3. Comparison of the parameters and the computational costs of three different structures with w be 8 and h be 30 on NEXAR2 validation set.

Method	Params	Mult-Adds
Standard convolution	0.29M[1]	70.8M
Large separable convolution	0.22M	53.1M
Bottleneck	0.07M	9.0M

[1] M represents million

Table 4. Performance comparison of large separable convolution and bottleneck on NEXAR2 validation set.

Method	mAP
Large separable convolution	62.2
Bottleneck	62.1

3.2 RPN

Since there are fewer classes in the Nexar2 dataset and the features are easier to extract, we change the RPN convolution channel from 256 to 128 channels, whose results are shown in **Table 5**. The accuracy has increased slightly, and we speculate that this is because we double the initial learning rate for training.

Table 5. The modified RPN performance has improved. The performance is evaluated on NEXAR2 validation set.

RPN	mAP
256	62.1
128	63.2

We named the modified network Light-Car Detection. In **Table 6**, we compare Light-Car Detection with the original network. The results show the performance is better and the calculation speed is faster.

Table 6. Performance comparison of Light-Head R-CNN and Light-Car Detection on NEXAR2 validation set.

Model	Backbone	Test Size short/max edge	Speed(ms)[1]	mAP
Light-Head RCNN	MobileNet	300/300	501	62.1
Light-Car Detection	MobileNet	300/300	276	63.2
Light-Head RCNN	MobileNet	300/300	36.2(GPU)	62.1
Light-Car Detection	MobileNet	300/300	17(GPU)	63.2

[1] Speed is the time required to process an image.

4. Pruning filters

Due to the limited computational ability and insufficient power resources of mobile devices, we need to prune filter of Light-Car Detection. There are many methods for pruning filters, one is based on data [36], and another is based on the amplitude of filters, such as [37, 38]. In this paper, we use L1-norm to select unimportant filters and physically prune them. Light-Car Detection has three different convolutions, standard convolution, depthwise convolution, and pointwise convolution. In comparison, depthwise convolution is an efficient one, so we only prune filters of standard convolution and pointwise convolution. We divide the procedure of pruning into four steps: First, we calculate the L1-norm of filters on each layer. In each layer, we compute the ratios of the L1-norm of each filter and the maximum of the L1-norm in this layer. After that, we count the proportion of the filter whose L1-norm ratio is less than a given value, and the result is shown in **Fig. 4**. Second, we prune several filters with small L1-norm and their corresponding feature maps. The kernels on the next convolutional layer and parameters in the next batch-normalization layer corresponding to the pruned feature maps are removed. From **Fig. 4**, we can find that the L1-norm of 10% convolution kernels is very low on the conv1 and conv2_1 /dw layers, so we remove these kernels. Third, we fine-tune Light-Car Detection using 10k mini-batches with 0.0001 learning rate on nexar2 dataset. In the process of pruning filters, we find conv1 and conv2* stages are more sensitive than conv3*, conv4* conv5* stages, where conv2* represents conv2_1, conv2_2, so it is with conv3*, conv4* and conv5*. After multiple pruning, the accuracy and running times of Light-Car Detection are shown in **Table 7**. It is worth noting that after pruning, Light-Car Detection is more efficient with only slightly accuracy loss.

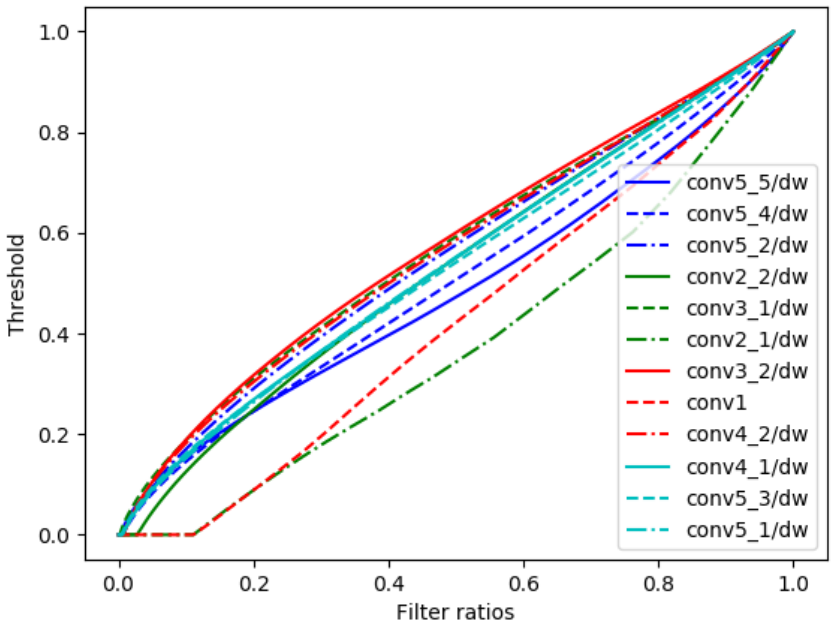

Fig. 4. Sorting filters by ratios for each layer of Light-Car Detection on nexar2. The y-axis is the given value. The x-axis is the proportion of the filter whose L1-norm ratio is less than a given value.

Table 7. The number of parameters, accuracy and running time after pruning filters. The performance is evaluated on NEXAR2 validation set.

Model	Backbone	Test Size short/max edge	Speed(ms)	mAP	Params(M)[1]
Light-Head RCNN	MobileNet	300/300	501	62.1	13.3
Light-Car Detection F[2]	MobileNet	300/300	200	63.2	13.0
Light-Car Detection L[3]	MobileNet	300/300	183	60.0	7.1
Light-Car Detection L[3]	MobileNet	300/300	10.9(GPU)	60.0	7.1

[1] Number of million bytes occupied by the network.
[2] F represents the first time pruning network.
[3] L represents the last time pruning network.

5. Stability of detection

The current evaluation criteria for object detection performance depends mainly on the IOU, but stability is also an essential factor to consider in our mission. If the center and scale of the detection bounding box of the detected target object are erratic, the distance between the current vehicle and the vehicle ahead we calculated will be unstable and not accurate enough. Therefore, ensuring a certain stability is necessary for reliable results. In order to reduce the jitter of the detection bounding boxes and ensure the real-time processing, we use the following two methods to improve the stability and evaluate the performance using method proposed by Zhang *et al.* [32].

5.1 Using weighted NMS to improve the detection stability

In the object detection methods, after scoring each proposal box according to the degree of coincidence, the NMS is generally used to suppress the redundant detection bounding box. The NMS selects the highest scored bounding box and suppresses the bounding box where the IOU value is higher than the set threshold. Admittedly, these suppressed borders also contain some useful information, so suppressing them can result in the loss of valid information. In this paper, we use weighted NMS [33] instead of NMS. In weighted NMS, the target bounding box coordinates are refined by all candidate bounding boxes. By using this method, the mAP and stability of object detection can be improved.

5.2 Using tracking to improve the detection stability

In video detection, bounding box jitter is unavoidable. We use the Median Flow (MF) [34], a short-term and efficient tracking method to track the bounding boxes. In the specific implementation process, we only track the bounding box with a score higher than 0.8. When the MF tracker reports reliable tracking bounding box, we obtain a target bounding box with a high IOU value, and then average the two bounding boxes to optimize the result.

6. Calculating vehicle distance

A portion of the road in front of the vehicle can be considered to be on the ground plane such that there is a homography transformation between the image and the ground [35]. The formula is

$$P = H \cdot p \tag{7}$$

Where p represents the point in the image, and P the point on the ground. The homography transformation H can be calculated by using 4 markers on the ground as shown in **Fig. 5**. We setup the ground coordinates as the x-axis coincide with the front of host vehicle, and the y-axis the moving direction. If it is assumed that the lower edge of the detected object's bounding box coincides with the ground, then any point p on the lower edge can be used to calculate the distance between the camera and the vehicle ahead. We use the center point of the lower edge of bounding box.

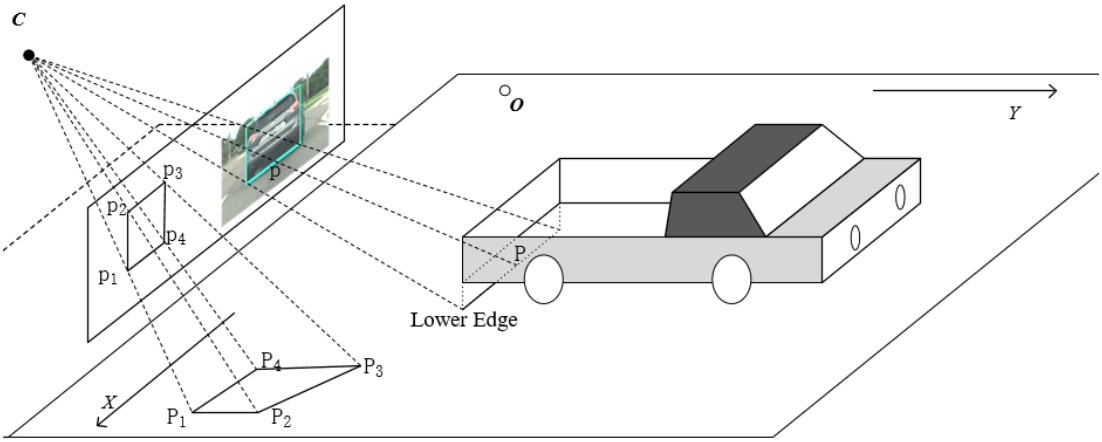

Fig. 5. The calculation method of homography transformation between the flat ground plane and the image, p is the center point of the lower edge of bounding box, so the distance is the y-coordinate of P .

7. Experiments

In this section, we first evaluate the accuracy and inference time of Light-Car Detection using the nexar2 dataset on PC. Aside from that, we evaluate the effect of weighted NMS and object tracking stability. After that, we use the mobile optimization library MACE to deploy Light-Car Detection to the smart phone and measure its inference time. Finally, we count the errors of our method in calculating the vehicle distance, and compare with methods using monocular camera.

7.1 Testing on PC

We test the runtimes on CPU and GPU under the condition of Ubuntu16.04 operating system, Inter i5-7500 CPU, 8G memory and GTX1060 GPU training network. The accuracy and inference time are shown in **Table 7**. Some detection results are given in **Fig. 6** to quantitative analysis the performance of proposed network.

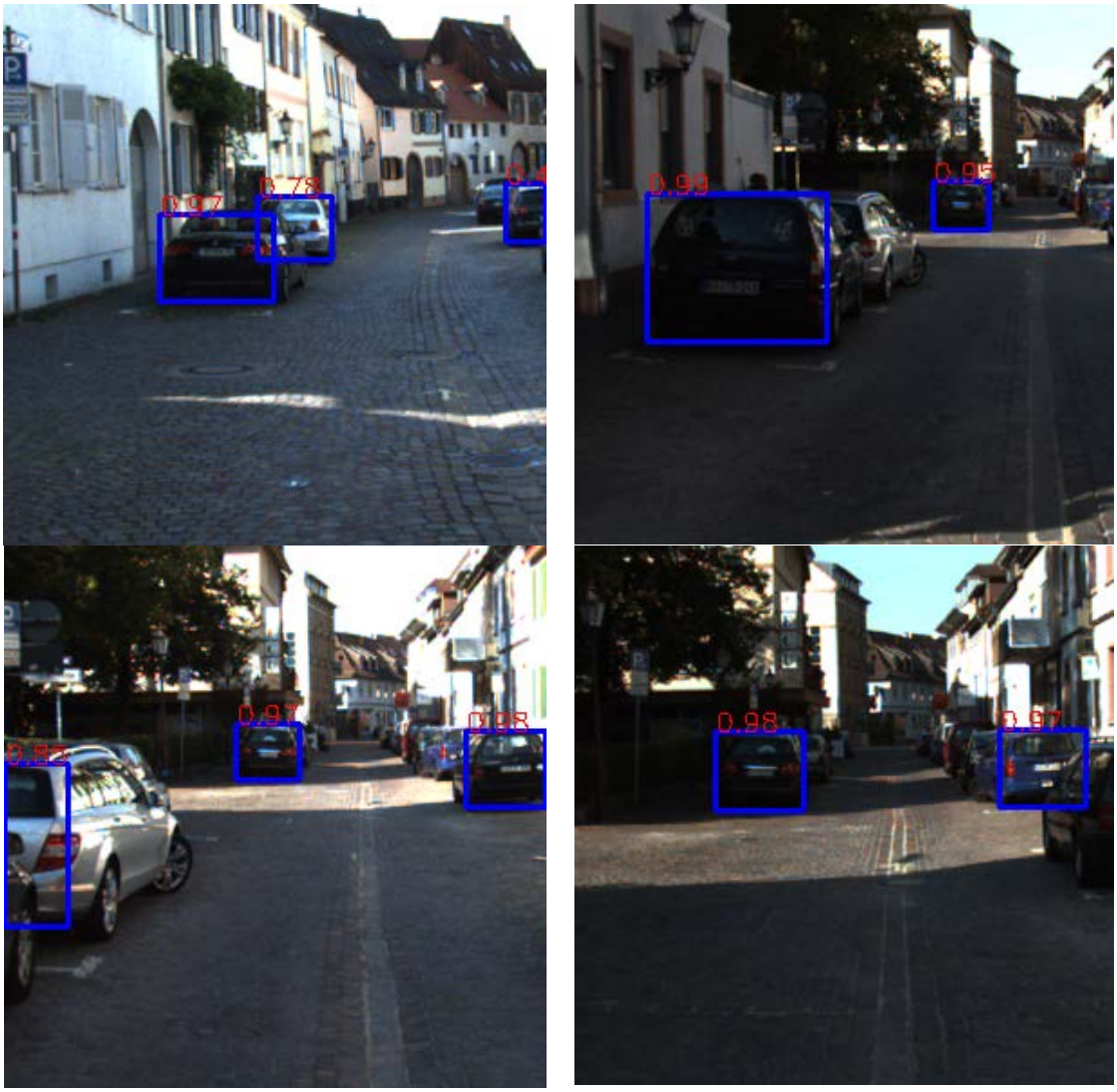

Fig. 6. Some examples of Light-Car Detection on the KITTI dataset.

7.2 Stability analysis

We measure object stability on KITTI object tracking dataset. In this dataset, the ground-truth 3D box surrounds the entire object in 3D space, but what we favored is the 2D bounding box of rear of the vehicle. We use the calibrated camera to project the 3D coordinates of the vehicle's 4 tail points to the image, and then we create the minimum bounding rectangle with these points as the real ground box. As shown in **Fig. 7**, the blue box is the 3D box of the car. The stability of the detection is measured by the scale and ratio of the bounding box, and the difference of positions of the center point [32]. The smaller the resulting value, the higher the stability of the test. We can find it in **Fig. 8a** and **Fig. 8b** that the weighted NMS and tracking can ensure the stability of the bounding box to a certain extent with keeping the stability of the center positions.

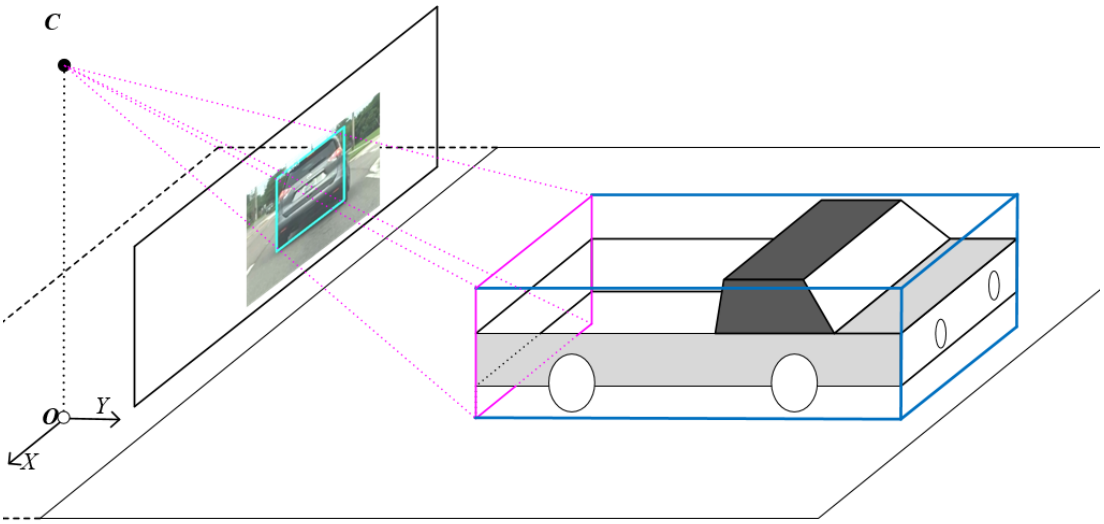

Fig. 7. The 3D car is surrounded by a 3D blue box, 4 tail points on the rear of the 3D box are projected to the image to create ground truth box. Here, we use the coordinate system defined in KITTI dataset.

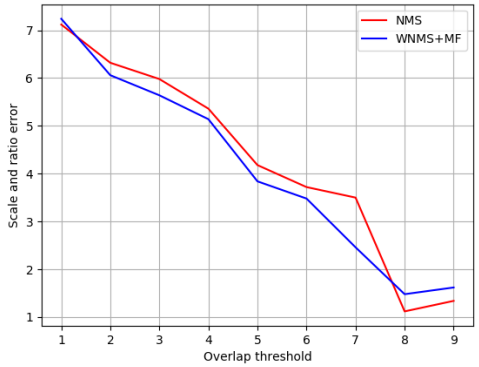

Fig. 8(a). Scale and ratio Error with respect to the overlap threshold[1]

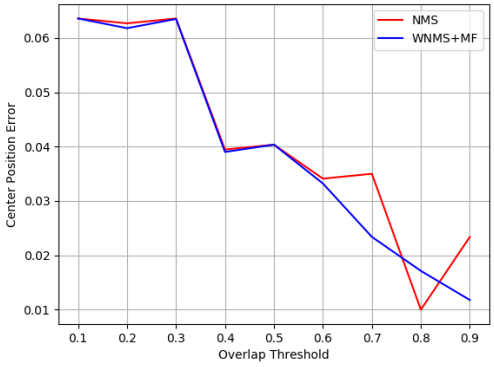

Fig. 8(b). Center Position Error with respect to the overlap threshold[1]

[1]The overlap is the IOU between predicted bounding box and ground truth box.

7.3 Deploying on mobile platforms

Mobile applications are benefited from the development of the deep learning, but mobile hardware (e.g. GPU and memory size) is limited compared to the desktop. After comparing several open source mobile deep learning frameworks, we chose MACE because it requires less memory and less time when inferring the trained model. To evaluate mobile-based inference, we use a Samsung Galaxy S7 mobile phone under Android 6.0 to run our trained model. The inference time is shown in **Table 8**, which satisfies real-time requirements on Adreno 530 GPU.

Table 8. Inference time on Samsung Galaxy S7.

SmartPhone	Device	Model	Image Size	Time(ms)
SamSung Galaxy S7	CPU	Snapdragon 820	300×300	215
SamSung Galaxy S7	GPU	Adreno 530	300×300	78.4

7.4 Car distance evaluation on the KITTI dataset

Recently, Geiger *et al.* [4] presented DPM based on conventional computer vision and Chen *et al.* [5] presented DeepDriving based on the convolutional neural network end-to-end distance prediction. Both of them achieve superior performance than previous methods. In this paper, we make a comparison with the two methods on the KITTI dataset.

The KITTI 3D object detection dataset contains a training set of 7480 images and a test set of 7480 images. In training set, all images emerge with official 3D labels for the positions of nearby cars, so we can easily extract the distance to other cars in the image. For each image, we define a 2D coordinate system on the zero height plane. The origin of the coordinate system is the center of the camera, the positive direction of the y-axis is directly in front of the direction of travel of the vehicle, and the direction of the x-axis is the right side perpendicular to the direction of travel. In the picture of the dataset, each picture contains multiple vehicles but only the most recent ones (one is on the left of the host car, one on its right, one on its ahead) in the DeepDriving method are selected for error estimation. We are consistent with this process. We divide the area in front of the current car into three parts according to the value range of the x coordinate: when $x \in [-1.6, 1.6]$, it is regarded as the area directly in front; when $x \in [-12, -1.6]$, it is regarded as the left area; When $x \in [1.6, 12]$, it is regarded as the right area; For our detection method, we may detect partial cars which only partially appear on the left lower corner or right lower corner. These cars are unlikely to locate at the ahead of host car and calculating the distance of these cars is inaccurate. Simultaneously, in DeepDriving and DPM, they do not count errors of these partial cars, so we also only count errors when the closest cars fully appear in the image. In DeepDriving and DPM, they count errors when cars show up within 50 meters ahead, in here, we are consistent with DeepDriving and DPM. We use the metrics proposed by Chen *et al.* [5] to evaluate errors, calculating The Mean Absolute Error (MAE) for the y and x coordinates and the Euclidean distance d between the estimation and the ground truth of the car position. The results are shown in **Table 9**. We give two results of our approaches, one is without WNMS and tracking the other has WNMS and tracking. Compared with DeepDriving and DPM, we obtain superior performance.

Table 9. Mean Absolute Error (in meters) on KITTI 3D object detection dataset.

Parameter	x	y	d
DeepDriving	1.097	4.332	4.669
DPM+Proj	1.214	5	5.331
Our approach (w/o WNMS+Tracking)	0.087	1.555	1.562
Our approach	0.094	1.248	1.258

When the vehicles ahead are far away from host car, namely as the distances increase, the absolute value of the difference between the obtained distance and the true value becomes larger as well. The result is shown in **Fig. 9**. It is noticed that even with a distance of 50 meters from the front car, our error is only about 4 meters. In practical applications, the error is acceptable.

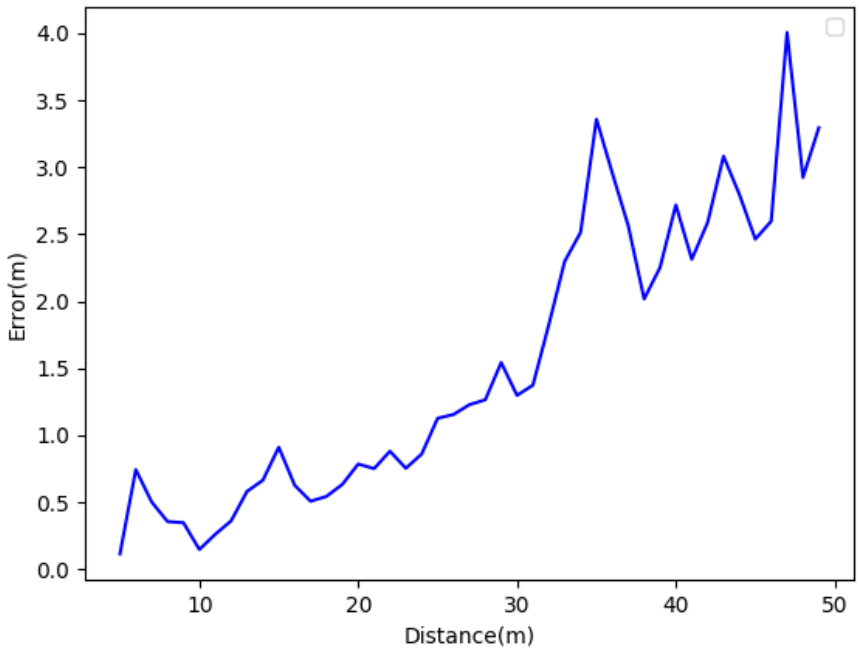

Fig. 9. The difference between the obtained distance and the true value at different car distances.

7.5 Car distance evaluation on the smart phone

To evaluate the performance of our approach in real-world, we also evaluate our approach using smartphone. Firstly, we collected 200 images, which contain 20 scenes, each scene contains 10 images. For each scene, the homography matrix was pre-computed. For each image, we used a meter ruler to measure the distance between the vehicle and the calibration board. Some examples are shown in **Fig. 10**. For quantitative evaluation, we use Mean Absolute Error to reflect the performance of our method. The results are shown in **Table 10**. The result shows that our method also can achieve good performance in the real-word scene.

Fig. 10. Some examples of our method on smart phone.

Table 10. The value of Mean Absolute Error (in meters) in different distance.

Distance	MAE
0<d<10	0.268
10<=d<20	1.457
20<=d<50	1.842

7.6 Compared with monocular depth estimation

Estimating the pixel-wise depth of scenes from RGB images has triggered wide research recently in the computer vision community. There have been many methods in the field of monocular depth estimation, which all use convolutional neural networks (CNNs) to end-to-end estimate the depth of scenes. Naturally, the methods of monocular depth estimation also can estimate the distance of ahead vehicle. In this paper, we compare our method with the state-of-art methods [41-44] of monocular depth estimation. In order to make the results fair and convincing, we all use the central point of lower edge of predicted bounding box to calculate the quantitative metrics in monocular depth estimation. Aside from that, we use 6 metrics [40] which are widely used in the field of monocular depth estimation to evaluate the performance. These metrics are shown as follow:

Threshold accuracy (δ_i): % of d_p s.t $\max(\frac{d_p}{d_p^*}, \frac{d_p^*}{d_p}) = \delta < thr$, $thr = 1.25,\ 1.25^2,\ 1.25^3$.

Average relative error (rel): $\frac{1}{n}\sum_p^n \frac{|d_p - d_p^*|}{d_p}$.

Average squared relative error (sq. rel): $\frac{1}{n}\sum_p^n \frac{|d_p - d_p^*|^2}{d_p}$.

Root mean squared error (rms): $\sqrt{\frac{1}{n}\sum_p^n (d_p - d_p^*)^2}$.

Average error ($\log_{10}$): $\frac{1}{n}\sum_p^n \left|\log_{10}(d_p) - \log_{10}(d_p^*)\right|$.

Where d_p is a pixel in the ground truth depth image, d_p^* is a pixel in the predicted depth image, and n is the number of valid pixels.

The results are provided in **Table 11**. We are able to achieve more accurate distance of ahead vehicle than monocular depth estimation.

Table 11. Comparison between our method and the method using monocular depth estimation on the KITTI dataset. The best results are bolded.

Method	δ_1	δ_2	δ_3	Rel	sq. rel	rms	$\log_{10}$
DORN [41]	0.959	0.994	0.999[1]	0.060	0.302	3.105	0.091
Kuznietsov et al. [42]	0.924	0.982	0.996	0.085	0.600	4.226	0.127
Monodepth2 [43]	0.901	0.963	0.980	0.107	1.049	5.050	0.191
MonoResMatch [44]	0.909	0.959	0.981	0.096	1.233	6.191	0.171
Ours	**0.995**	**0.998**	**1**[2]	**0.047**	**0.172**	**2.798**	**0.020**
Higher is better				Lower is better			

[1]The more accurate numeric is 0.99883
[2]The more accurate numeric is 0.99992

8. Conclusion

In this paper, we present a Light-Car Detection method to detect distance of the vehicles ahead. We combine the deep learning with traditional computer vision, using WNMS and MF tracking to improve the detection stability and accuracy. We refined thin feature maps and RPN subnetwork. The pruned network achieves 13FPS on Samsung Galaxy S7 mobile phone. We compare with two methods which is used in predicting the vehicle distance and four monocular depth estimation methods which predict the depth of scenes end-to-end. Experimental results show that our approach obtains superior performance.

References

[1] H. Kong, J. Y. Audibert, and J. Pone, "Vanishing point detection for road detection," in *Proc. of 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 96-103, 2009. Article (CrossRef Link).

[2] C. H. Chen, T. Y. Chen, D. Y. Huang, and K. W. Feng, "Front vehicle detection and distance estimation using single-lens video camera," in *Proc. of 2015 Third International Conference on Robot, Vision and Signal Processing (RVSP)*, pp. 14-17, 2015. Article (CrossRef Link).

[3] D. Y. Huang, C. H. Chen, T. Y. Chen, W. C. Hu, and K. W. Feng, "Vehicle detection and inter-vehicle distance estimation using single-lens video camera on urban/suburb roads," *Journal*

Visual Communication and Image Representation, vol. 46, pp. 250-259, 2017. Article (CrossRef Link).
[4] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasum, "3d traffic scene understanding from movable platforms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 5, pp. 1012-1025, 2014. Article (CrossRef Link).
[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 2722-2730, 2015. Article (CrossRef Link).
[6] C. C. Tsai, Y. T. Lai, Y. F. Li, and J. G. Guo, "A vision radar system for car safety driving applications," in *Proc. of 2017 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1-4, 2017. Article (CrossRef Link).
[7] M. Chen, D. Zhao, J. Sun, and H. Peng, "Improving Localization Accuracy in Connected Vehicle Networks Using Rao-Blackwellized Particle Filters: Theory, Simulations, and Experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol.20, no.6, pp.2255-2266, 2019. Article (CrossRef Link).
[8] N. Sasaki, S. Tomaru, and S. Nakamura, "Development of inter-vehicle distance measurement system using camera-equipped portable device," in *Proc. of 2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pp. 994-997, 2017. Article (CrossRef Link).
[9] F. de Ponte Müller, "Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles," *Sensors*, 17(2), 271, 2017. Article (CrossRef Link).
[10] K. Y. Park and S. Y. Hwang, "Robust Range Estimation with a Monocular Camera for Vision-Based Forward Collision Warning System," *the Scientific World Journal*, vol.2014, p.9, 2014. Article (CrossRef Link).
[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097-1105, 2012. Article (CrossRef Link).
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 1-9, 2015. Article (CrossRef Link).
[14] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016. Article (CrossRef Link).
[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.
[16] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated residual transformations for deep neural network," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 1492-1500, 2017. Article (CrossRef Link).
[17] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 7132-7141, 2018. Article (CrossRef Link).
[18] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.
[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. C. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510-4520, 2018. Article (CrossRef Link).
[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. of European conference on computer vision*, pp. 21-37, 2016. Article (CrossRef Link).
[21] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271, 2017. Article (CrossRef Link).

[22] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection," in *Proc. of the IEEE international conference on computer vision*, pp. 2980-2988, 2017. Article (CrossRef Link).
[23] Redmon J, Farhadi A. "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[24] J. Dai, Y. Li, K. He and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, pp. 379-387, 2016. Article (CrossRef Link).
[25] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng and J. Sun, "Light-Head R-CNN: In Defense of Two-Stage Object Detector," *arXiv preprint arXiv:1711.07264*, 2017.
[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[27] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. of European conference on computer vision*, pp. 740-755, 2014. Article (CrossRef Link).
[28] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, pp. 91-99, 2015. Article (CrossRef Link).
[29] A. Shrivastava, A. Gupta and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761-769, 2016. Article (CrossRef Link).
[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541-551, 1989. Article (CrossRef Link).
[31] N. Bodla, B. Singh, R. Chellappa and L. S. Davis, "Soft-NMS--Improving Object Detection With One Line of Code," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 5561-5569, 2017. Article (CrossRef Link).
[32] H. Zhang and N. Wang, "On The Stability of Video Detection and Tracking," *arXiv preprint arXiv:1611.06467*, 2016.
[33] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 1134-1142, 2015. Article (CrossRef Link).
[34] Z. Kalal, K. Mikolajczyk and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Proc. of 2010 20th International Conference on Pattern Recognition*, pp. 2756-2759, 2010. Article (CrossRef Link).
[35] D.A. Forsyth and J. Ponce, "Computer Vision: A Modern Approach," *Pearson Education Inc.* 2003.
[36] Yihui He, X. Zhang, J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. of The IEEE International Conference on Computer Vision (ICCV)*, pp. 1389-1397, 2017. Article (CrossRef Link).
[37] H. Li, A. Kadav, I. Durdanovic, H. Samet and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
[38] S. Han, H. Mao, W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
[39] Y. Cuiping, "Research on Safe Distance between Vehicles for Freeway Anti-collision System," *Process Automation Instrumentation*, vol. 9, p. 005, 2008. Article (CrossRef Link).
[40] D. Eigen, C. Puhrsch, and R. Fergus. "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, pp. 2366-2374, 2014. Article (CrossRef Link).
[41] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. "Deep ordinal regression network for monocular depth estimation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2002-2011, 2018. Article (CrossRef Link).

[42] Y. Kuznietsov, J. Stuckler, and B. Leibe. "Semi-supervised deep learning for monocular depth map prediction," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6647-6655, 2017. Article (CrossRef Link).
[43] C. Godard. O. M. Aodha, and G. Brostow. "Digging into self-supervised monocular depth estimation," *arXiv preprint arXiv:1806.01260*. 2018.
[44] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia. "Learning monocular depth estimation infusing traditional stereo knowledge," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9799-9809, 2019. Article (CrossRef Link).

Guang Jiang is an associate professor in the School of Telecommunications Engineering at Xidian University, Xian, China. Currently, his main research areas include image processing and computer vision. His main research goals are to develop approaches to analyze, understand and code the images or videos of real world objects and scenes based on vision geometry.

QuanMeng Ma is a master student in the School of Telecommunications Engineering at Xidian University, Xi'an, China. His research interests involve Image Processing and Computer Vision.

DianZhi Lai is a master student in the School of Telecommunications Engineering at Xidian University, Xi'an, China. Her research interests involve Image Processing and Computer Vision.

Hua Cui received her B.S. degree in Mathematics from Henan Normal University, Xinxiang, China, in 1999. She received her M.S. and Ph.D. degrees in Applied Mathematics from Xidian University, Xi'an, China, in 2005 and in 2008, respectively. She has been on the faculty of the School of Information Engineering, Chang'an University, Xi'an, China, since 2008. Her research interests include neural network applications, and modeling and predistortion for nonlinear power amplifiers.

Huansheng Song professor with the School of Information Engineering, Chang'an University, Xi'an, China. His current research interests include digital image processing and wireless communication systems.