Vision based ADAS for Forward Vehicle Detection using Convolutional Neural Networks and Motion Tracking

Chen-Wei Lai[1], Huei-Yung Lin[2] and Wen-Lung Tai[3]

[1]*Department of Electrical Engineering, National Chung Cheng University, 168 University Road, Min-Hsiung, Chiayi 621, Taiwan*

[2]*Department of Electrical Engineering and Advanced Institute of Manufacturing with High-Tech Innovation, National Chung Cheng University, 168 University Road, Min-Hsiung, Chiayi 621, Taiwan*

[3]*Create Electronic Optical Co., LTD, 868 Zhongzheng Road, Zhonghe, New Taipei 23557, Taiwan*

Keywords: Forward Vehicle Detection, Advanced Driving Assistance Systems, Convolutional Neural Networks, Motion Tracking.

Abstract: With the rapid development of advanced driving assistance technologies, from the very beginning of parking assistance, lane departure warning, forward collision warning, to active distance control cruise, the active safety protection of vehicles has gained the popularity in recent years. However, there are several important issues in the image based forward collision warning systems. If the characteristics of vehicles are defined manually for detection, we need to consider various conditions to set the threshold to fit a variety of the environment change. Although the state-of-art machine learning methods can provide more accurate results then ever, the required computation cost is far much higher. In order to find a balance between these two approaches, we present a detection-tracking technique for forward collision warning. The motion tracking algorithm is built on top of the convolutional neural networks for vehicle detection. For all processed image frames, the ratio between detection and tracking is well adjusted to achieve a good performance with an accuracy/computation trade-off. Th experiments with real-time results are presented with a GPU computing platform.

1 INTRODUCTION

This paper proposes an idea that effectively enables a common network to achieve real-time computation on the hardware that generally has GPUs. With the development of Convolutional Neural Networks (CNN) in recent years, the detection of objects has become more and more accurate and faster. However, using these network architectures in real time has a major problem for computationally inefficient platforms or embedded boards. In order to deal with such problems, it is required to use the network to detect every frame under the platform with poor computing power. Thus, this becomes a severe issue to be solved.

This work is based on the concept that there is little difference between the consecutive image frames. In an image sequence, we can only detect some other frames rather than each of them. The detection framework is based on the sparsely detected bounding box, in order to achieve the real-time performance of the networks. In addition to the comparison on computation speed, we also conduct the experiments using 29,681 images recorded with a dashcam to verify the reliability of the proposed method. On the platform using a GPU GTX 950M, the average operation speed is about 30–55 frames per second, and the accuracy is 76%. The results demonstrate that the technique proposed in this work can be effectively applied to most common platforms.

2 RELATED WORK

The early research on vehicle detection includes manual detection and using the characteristics of the vehicles (Sun et al., 2006b) . More recently, machine learning techniques are widely adopted and many approaches have been proposed. The former methods are usually to identify the features of the vehicles for detection purposes, and one popular approach for the latter case is the convolutional neural network based methods.

Lai, C., Lin, H. and Tai, W.
Vision based ADAS for Forward Vehicle Detection using Convolutional Neural Networks and Motion Tracking.
DOI: 10.5220/0007626902970304
In *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2019)*, pages 297-304
ISBN: 978-989-758-374-2

2.1 Conventional Approaches

The conventional approaches for vehicle detection commonly adopt the image feature extraction techniques and classification algorithms.

Vertical and Horizontal Edges. The edge has always been an important feature in computer vision applications. Srinivasa (Srinivasa, 2002) and Sun em et al. (Sun et al., 2002) used this characteristic to identify front vehicles. However, the problem with the edge methods is how to set a suitable threshold. Different thresholds must be used for different application scenarios to obtain the best results. Although the methods can be improved in an adaptive manner, when the color of the vehicle body is similar to the surrounding environment, the edges cannot be detected easily and correctly.

Shadow. Extracting features directly from the objects is sometimes not the best solution for object detection. The information around the object might be used effectively. One good example is the shadow under the vehicle. Based on the observation, the darkest part of the image has a very high probability of being located at the bottom of the vehicle region. According to this characteristic, Tzomakas and von eelen (Tzomakas and von Seelen, 1998) proposed a way to effectively determine the gray value threshold to detect the vehicle. However, this method is prone to misjudge the shadow regions in the night scenes.

Tail Light. To deal with the problem of detecting vehicles at night, OMalley *et al.* (O'Malley et al., 2008) used tail lights to identify the front vehicle location. The core idea is that the tail lights are red and easy to recognize. Since some vehicles do not have red tail lights or have the lights on, this does not guarantee that all true positives can be considered.

Support Vector Machine. Through the characteristics of vehicles, such as the simple features described above, we are able to detect the vehicle location under normal conditions. For more complicated situations, it is difficult to detect the vehicles correctly. Some previous works proposed to use Haar transform or HOG to extract textures, and use support vector machines or simple neural networks for vehicle detection and verification (Ortega et al., 2013; Sun et al., 2006a). The results obtained from this approaches are generally better, but the computation is the key issue. The operation time is dramatically increased due to the bounding box extraction using a sliding window. It is thus fatal for real-time applications.

2.2 Convolutional Neural Networks

Since the year of 2012, the convolutional neural networks used for object detection and classification have gained a great success. The object detection research is mainly divided into two categories: one-stage and two-stage detectors.

Two-Stage Detector. The two-stage detector is the first development with a significant detection rate based on the recent deep learning framework. Its architecture mainly consists of two parts: region proposal and prediction of the content of the detected bounding box (Girshick et al., 2014; Girshick, 2015; Dai et al., 2016). Most region proposals are designed to be very large in pursuit of good results, and thus it slows down the computing speed of the network. In order to mitigate this issue, Li *et al.* (Li et al., 2017) refer to the idea of (Szegedy et al., 2015; Chollet, 2016) to reduce the amount of calculation while keeping the results satisfied.

One Stage Detector. Even the two-stage detectors have very good recognition rates, they are not suitable for many application scenarios in terms of the execution speed. To deal with this problem, Redmon and Farhadi divide the input image into an $n \times n$ grid and use them to generate several different sizes of bounding boxes instead of using the network with region proposal (Redmon and Farhadi, 2016). In the proposed method, each grid only predicts one object and the major cost is that it does not work well for the detection of small objects. To reduce the cost, Liu *et al.* propose a method to forecast at different scales and this is able to deal with the small size objects fairly well (Liu et al., 2016).

3 METHOD

In the proposed technique, we first use a machine learning algorithm to detect the vehicles, and then SVM is adopted as a verification mechanism to effectively remove the non-vehicle bounding boxes. Finally, a tracking algorithm is carried out to reduce the overall computation time. The real-time system can be realized by adjusting the detection and tracking rate. Figure 1 shows the flowchart of the proposed method.

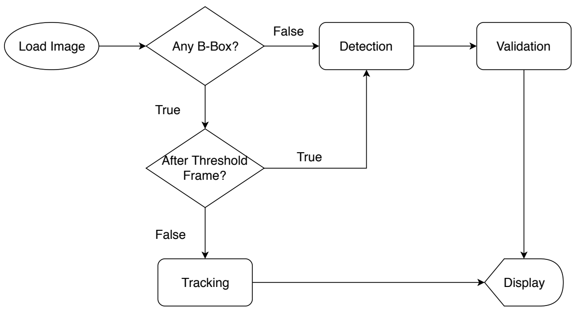


Figure 1: The flowchart of the proposed method. It combines the detection, verification and tracking to improve the overall performance.

3.1 Detection

The network architecture we adopt is YOLOv2. In order to detect the location of the vehicle more correctly, we not only extract the features from the lower layers, but also modify the category prediction part of the cost function. Furthermore, the focal loss is added to improve the overall accuracy.

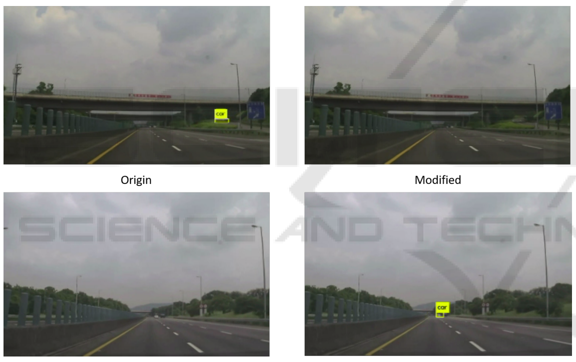


Figure 2: The result comparison between the original YOLO and ours. The left images are the original and the right images are obtained from our modification. After the improvement, the false positive is removed and the small object can be detected.

3.1.1 Extraction from Lower Layer

The detection of small objects is the most criticized part of YOLO. In the existing literature, there are three approaches to cope with this problems: 1. Enlarge the images for the input network. 2. Increase the number of the grid cell (Behrendt et al., 2017). 3. Extract features from the lower layers (Kong et al., 2016). The third approach is adopted in this work because the first two modifications are more computationally intensive for object detection. The main reason that this method is effective is because the lower layers pass fewer convolution layers. The characteristics of the small objects do not cause the features to disappear due to the convolution layers. In order to avoid the network prediction part becomes too large, we only combine the features extracted from the sixth layer to the back-end for prediction. Figure 3 shows the original architecture and the one with our modification.

3.1.2 Category Prediction Modification

Another problem with YOLO is that many times the position of the bounding box is correct but the category prediction is not. In order to deal with such problems, we increase the penalty for the category prediction. The following is the original cost function used by YOLO.

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbf{1}_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned} \tag{1}
$$

The first two terms represent the position error of the prediction and the training set respectively, and the middle two terms represent how the predicted object is compared with the training set in the grid cell. The last item represents whether the predicted object's category is the same as the training set.

In the last part, the objective is to improve the prediction by modifying the penalty function. To avoid the situation of not converging when training the network due to too many changes, we only double the result of each calculation.

From such a modification, we find that the improvement is fairly limited. Thus, we refer to the RetinaNet (Lin et al., 2017) and incorporate the focal loss as part of loss function. The RetinaNet has put forward new ideas for the low accuracy issue of one stage detectors. The work points out two major reasons for poor detection: 1. Extremely uneven data between the positive and negative samples. 2. The loss function is easily affected by negative samples. In order to deal with this situation, they modify the original cross entropy, and the results are shown in Figure 4. Such a modification really reduces the overall loss, and it does have the same effect and improvement in our experiments.

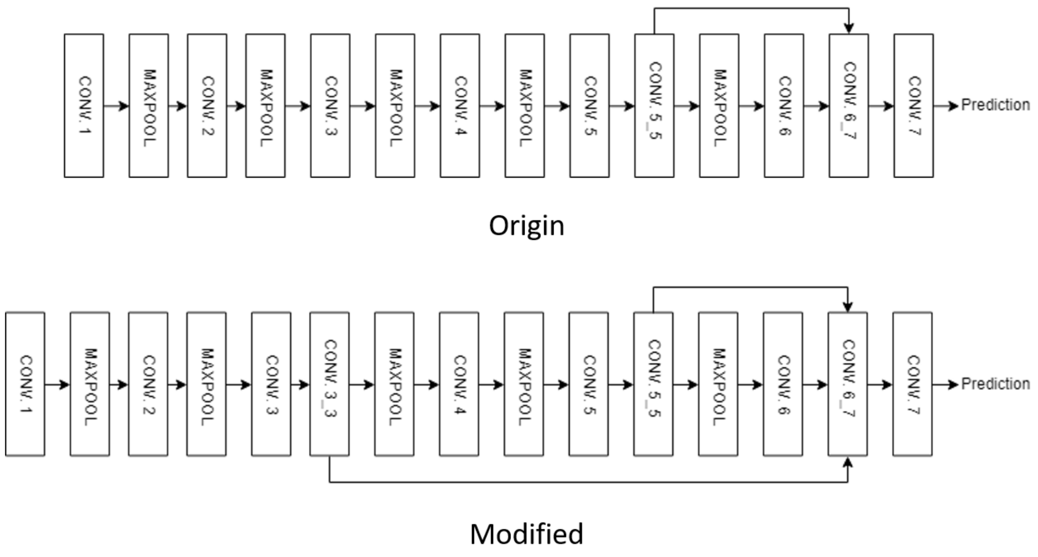


Figure 3: The original YOLO network architecture (top) and the one with our modification (bottom).

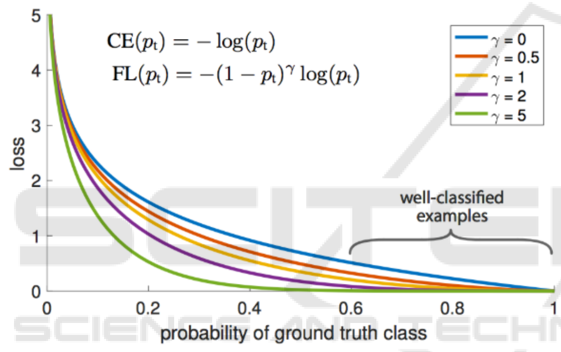


Figure 4: The comparison on the focal loss and the cross entropy (Lin et al., 2017).

3.2 Validation

Even many changes have been made in Section 3.1 to reduce the false positives, the mis-detection problem still exists. To reduce the incorrect classification more effectively, we use HOG+SVM to remove the wrong bounding boxes. Figure 5 shows an example that the negative bounding box is removed after we add this validation stage.

3.3 Tracking

In the existing literature, the tracking methods are usually used to reinforce the results they have detected, rather than improve the overall computation time. The front vehicles in a common traffic scene recorded with 30 frames per second do not change too much in the images. Thus, it is reasonable that performing object detection for each image frame is not necessary. In order to make the algorithm achieve the

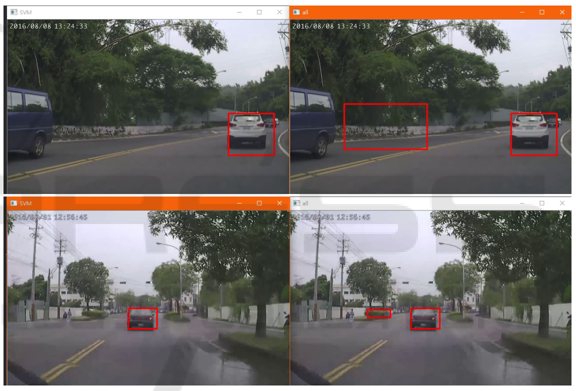


Figure 5: The results from HOG+SVM. The right and left images show the results before and after the verification mechanism is adopted.

overall real-time operation, we use only a few frames for detection. The tracking algorithm is applied on the remaining frames instead of frame-by-frame detection. Eq. (2) indicates how the threshold, f_{th}, is defined:

$$f_{th} = \frac{30 - f_d}{f_t - f_d} \tag{2}$$

where f_d is the detection speed and f_t is the tracking speed.

After the detection and verification are completed, we track the detected bounding box, switch to detection verification after the f_{th} frame, and clear the tracking bounding box. In this work, our tracking algorithm is modified from the compressive tracking (Zhang et al., 2012) with the multi-object tracker.

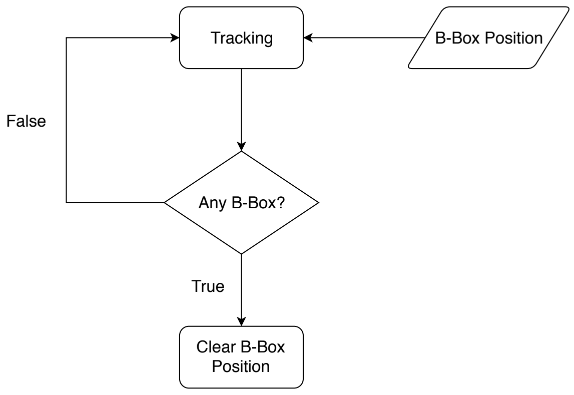


Figure 6: The flowchart of our proposed method with the tracking step.

Table 1: The comparison of our method and the original YOLO algorithm on the KITTI dataset. The results are reported with vehicle detection only.

	Precision	Recall
YOLOv2	0.55	0.49
Our method	0.64	0.47

4 EXPERIMENTS

Our training set for YOLO is the COCO (Lin et al., 2014) dataset and the training sets for SVM are KITTI (Geiger et al., 2012) and GTI (Arróspide et al., 2012). The testing set is captured by a driving recorder with the resolution is 1920×1080. The computation and evaluation are carried out on a laptop with Nvidia GTX950M GPU and Intel i7-6700HQ CPU. The processing speed for the original YOLO algorithm, YOLO+SVM and YOLO+SVM+Tracking are 12 – 13 fps, 12 fps and 30 – 40 fps, respectively. It shows that our method is capable of performing real-time applications

We first test the algorithms on the public datasets. Table 1 shows the results (precision and recall) of our method and the original YOLOv2 algorithm. We can see that, after our modification, the precision is greatly improved even the recall rate is slightly reduced. The algorithms are then tested on the COCO datasets. Table 2 presents the comparison on the mean average precision (mAP) between the original YOLOv2 and our technique. It shows that the results in terms of mAP from our method are lower than the original one. This is due to the modification is mainly for precision improvement and the recall rate is somehow sacrificed (as illustrated in Table 1). We believe that the precision is more important if the technique aims to be used for forward vehicle collision detection. Table 3 shows the precision of the categories of interests in the COCO dataset.

Finally, we test the algorithms using our own dataset. Table 4 shows the comparison between our method and others on far distance vehicles. Through our modifications, the results of detecting vehicles at longer distances are more precise. For the recall rate, the result is much lower than YOLOv3. It shows that our modification is an effective method for the detection of long distance vehicles.

Table 5 tabulates the overall comparison between other methods and ours. It can be seen that, with our improvement, the precision is higher than the original YOLOv2 and YOLOV3. The recall rate is the lowest but very close to YOLOv2, which is considered acceptable. For the method of "Modification + SVM", the recall is the lowest but it has the best precision. There are two reason of the low recall:

Position/size of the Bounding Box. If the bounding box contains part of the vehicles, in most case SVM does not identify them correctly.

Training Set. In our training set, there are almost no images of vehicles being obscured. Thus, the bounding box with obscured vehicles will be removed.

5 CONCLUSIONS

This paper proposes a solution that effectively implements the real-time operation of existing neural networks on low-computing GPU platforms. Compared to the commonly used techniques to track the robust detection results, we use it to speed up the overall computation time. On the other hand, we modify the network architecture of YOLOv2, add the verification mechanisms to reduce false positives, and port the Darknet network architecture to Texas Instruments TDA2 platform running on a CPU. In the future, the TIDL development kit working on GPU will be used to improve the detection speed.

ACKNOWLEDGMENTS

This work was financially/partially supported by the Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, the Ministry of Science and Technology of Taiwan under Grant MOST 106-2221-E-194-004 and Create Electronic Optical Co., LTD, Taiwan.

Table 2: The comparison on the mAP between YOLOv2 and our method.

	YOLOv2	Our method	
Low layer extraction		√	√
Low layer extraction with focal loss			√
mAP(%)	44.78	33.21	33.71

Table 3: The precision comparison on several interested categories.

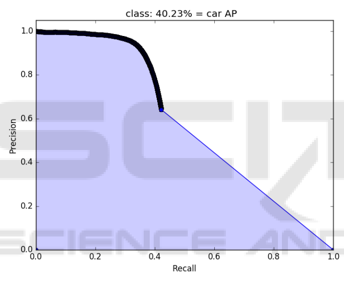
	Original YOLOv2	Low layer extraction	Low layer extraction and focal loss
Car	0.61	0.6	0.68
Bus	0.84	0.82	0.86
Truck	0.71	0.67	0.68

Table 4: The comparison of far distance vehicle detection on our dataset.

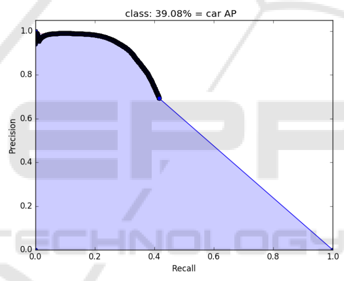
	Original YOLOv2	Our method	YOLOv3
Recall	0.09	0.14	0.41
Precision	0.18	0.38	0.31

Table 5: The overall comparison of different approaches on our dataset.

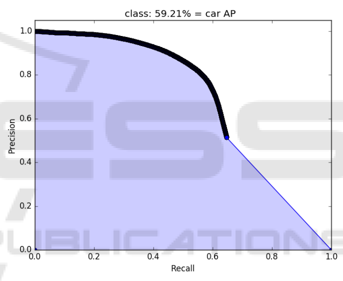
	Original YOLOv2	Our method	Modification+SVM	YOLOv3
Recall	0.42	0.41	0.28	0.65
Precision	0.64	0.69	0.76	0.51



(a) Original YOLOv2



(b) Our method



(c) YOLOv3

Figure 7: The comparison of average precision on different methods.

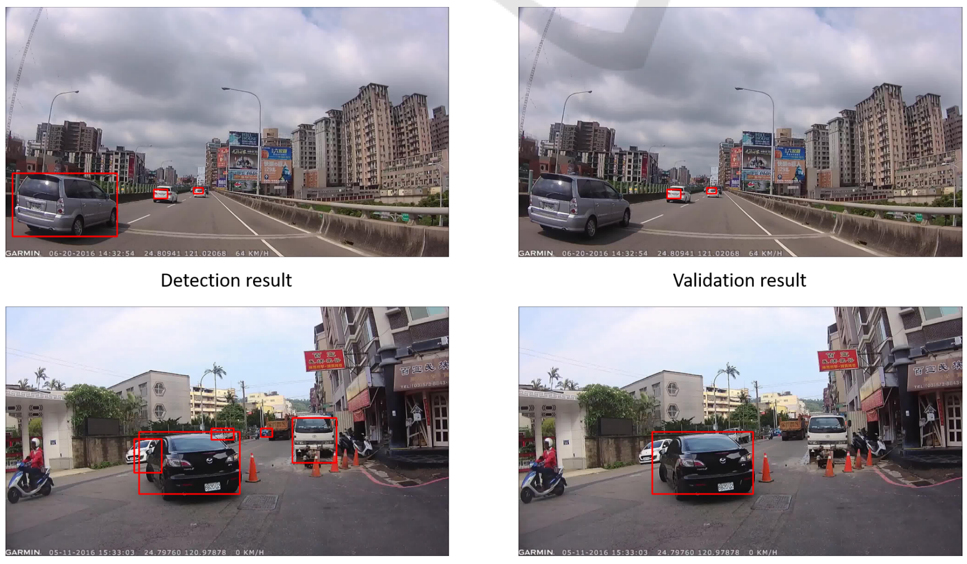


Figure 8: Incorrect detection from SVM.

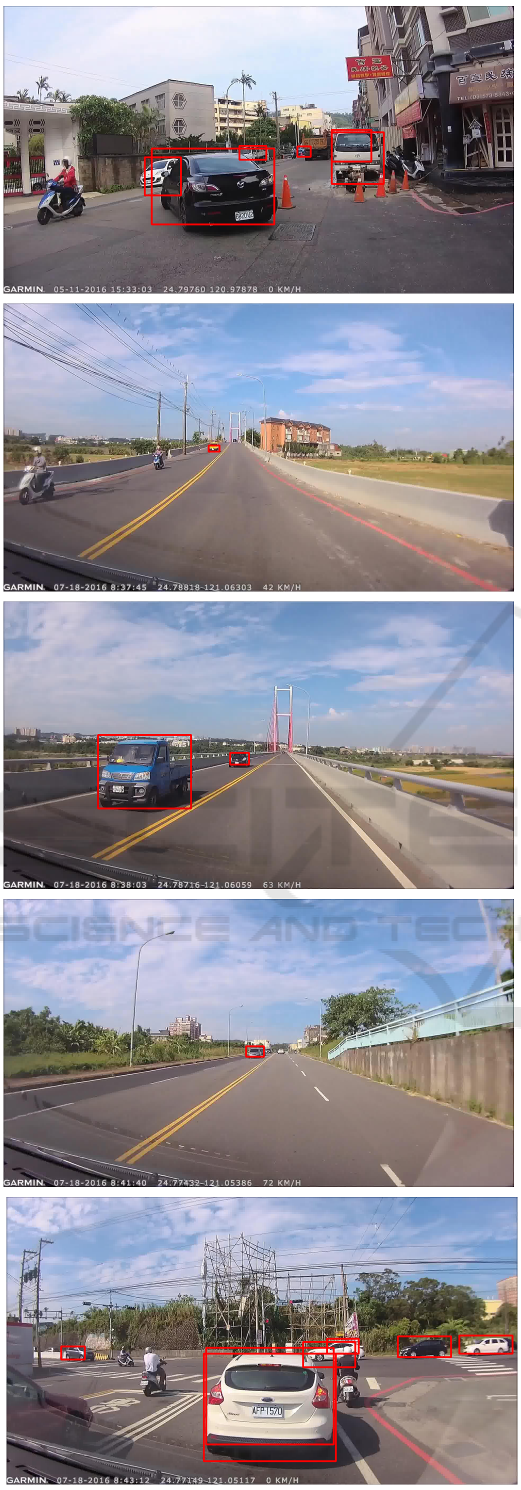


Figure 9: The detection results on our dataset.

REFERENCES

Arróspide, J., Salgado, L., and Nieto, M. (2012). Video analysis-based vehicle detection and tracking using an mcmc sampling framework. *EURASIP Journal on Advances in Signal Processing*, 2012(1):2.

Behrendt, K., Novak, L., and Botros, R. (2017). A deep learning approach to traffic lights: Detection, tracking, and classification. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1370–1377. IEEE.

Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R. (2015). Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*.

Kong, T., Yao, A., Chen, Y., and Sun, F. (2016). Hypernet: Towards accurate region proposal generation and joint object detection. *CoRR*, abs/1604.00600.

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., and Sun, J. (2017). Light-head R-CNN: in defense of two-stage object detector. *CoRR*, abs/1711.07264.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37.

Ortega, J. D., Nieto, M., Cortes, A., and Florez, J. (2013). Perspective multiscale detection of vehicles for real-time forward collision avoidance systems. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 645–656. Springer.

O'Malley, R., Glavin, M., and Jones, E. (2008). Vehicle detection at night based on tail-light detection. In *1st international symposium on vehicular computing systems, Trinity College Dublin*.

Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

Srinivasa, N. (2002). Vision-based vehicle detection and tracking method for forward collision warning in automobiles. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 626–631. IEEE.

Sun, Z., Bebis, G., and Miller, R. (2006a). Monocular pre-crash vehicle detection: features and classifiers. *IEEE transactions on image processing*, 15(7):2019–2034.

Sun, Z., Bebis, G., and Miller, R. (2006b). On-road vehicle detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 28(5):694–711.

Sun, Z., Miller, R., Bebis, G., and DiMeo, D. (2002). A real-time precrash vehicle detection system. In *null*, page 171. IEEE.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

Tzomakas, C. and von Seelen, W. (1998). Vehicle detection in traffic scenes using shadows. In *Ir-Ini, Institut fur Nueroinformatik, Ruhr-Universitat*. Citeseer.

Zhang, K., Zhang, L., and Yang, M.-H. (2012). Real-time compressive tracking. In *European conference on computer vision*, pages 864–877. Springer.