Research on Real-Time Vehicle Detection Algorithm Based on Deep Learning

Wei Yang, Ji Zhang, Zhongbao Zhang, and Hongyuan Wang(✉)

Changzhou University, Changzhou, Jiangsu, China
hywang@cczu.edu.cn

Abstract. At present, the demand for transportation is continuously increasing, and the consequent traffic congestion problem has become more and more prominent. How to automatically and timely detect vehicles to analyze road traffic information is an important issue for intelligent traffic monitoring systems (ITS). In some existing methods for to detect vehicles, real-time performance and precision cannot be taken into account at the same time. Hereby, a method of automatic vehicle detection, which has the high performance on real-time and precision, is proposed in this paper. This method improves the YOLOv2 framework model in following aspects: introducing a new loss function, expanding the grid size, and optimizing the number and size of anchors in the model to automatically learn vehicle characteristics. Compared with YOLOv2, YOLOv3 and Faster RCNN, both the precision and the real-time performance of this method are improved competitively.

Keywords: Vehicle detection · Real-time detection · YOLOv2
Loss function

1 Introduction

At present, the level of urbanization in China has exceeded 50% and the number of car ownership has reached 310 million by the end of 2017. With traffic congestion and numerous potential safety issues emerged, it is on rise for the demand for intelligent traffic monitoring systems (ITS). Countries all over the world have invested a large amount of manpower and material resources into the research and development of various transportation technologies. Object detection is an indispensable part of ITS. It is also one of the most important research topics in computer vision, artificial intelligence, pattern recognition, image processing and machine learning. It can provide strong information to support for many traffic links such as road traffic control, highway management and emergency management. Its performance has a direct impact on follow-up objection tracking, objection classification, and objection recognition.

Detection of moving objects from video sequences is an important research issue in computer vision and other application fields. The traditional object detection is generally divided into three stages (Fig. 1): selecting candidate regions, extracting the corresponding features and classification. In the feature extraction process, it usually uses artificial methods, such as Histogram of Oriented Gradient (HOG) [1–3], Scale-invariant Feature Transform (SIFT) [4, 5]. The effect of the model recognition by

© Springer Nature Switzerland AG 2018
J.-H. Lai et al. (Eds.): PRCV 2018, LNCS 11259, pp. 126–137, 2018.
https://doi.org/10.1007/978-3-030-03341-5_11

this method is greatly influenced by the artificial. The extracted features are inputted into the classification, such as the Support Vector Machine (SVM) [6], the AdaBoost [7], DPM [8] and RF [9]. In 2010, Felzenszwalb et al. proposed DPM, in which the features extraction doesn't needed to be separated from the classifier training, and make full use of the advantages of HOG and SVM. This has greatly improved the detection effect. However, the disadvantage of the DPM model is its high complexity and low object detection speed and precision.

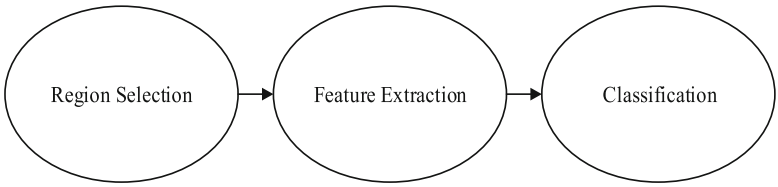

Fig. 1. Three stages of traditional object detection

However, as the traffic environment becomes more and more complex, the characteristics of the manual design are not very robust to the diversity of the objection. Therefore, the application of deep learning to object detection is a research trend. The RCNN [10] proposed by Ross et al. has made great breakthroughs in the field of object detection, followed by SPP-net [11], Fast-RCNN [12], Faster-RCNN [13], R-FCN [14], and YOLO [15], SSD [16] and other algorithms. CNN was first added to RCNN [10] for object detection. The method solves the problem that how to train high-quality models with a small amount of the labeled data and becomes the mainstream method in the field of object detection. However, RCNN [10] needs to perform a forward CNN to carry out feature extraction for each proposal extracted from Selective Search [17] (SS). Therefore, the calculation volume is too large to be updated in real time. Fast-RCNN [12] model is an improvement on RCNN [10], which avoids the redundant feature extraction operation in RCNN [10], and adopts adaptive scale pooling to optimize the entire network to improve the precision of deep network detection and recognition. However, Fast-RCNN [12] uses the Selective Search Algorithm (SS) [9] to extract candidate regions, which takes a long time. And the Faster-RCNN [13] algorithm introduces the RPN network on this foundation, which shortens the time of proposal extraction obviously, and achieves the highest object detection precision on the VOC2007 and VOC2012 datasets. Although the RCNN series have high detection precision, they still can't meet the real-time requirement on the detection speed. On the other hand, Ross et al. proposed the YOLO [11] algorithm, which can achieve 45FPS using GPU acceleration on PASCAL VOC, and YOLOv2 algorithm, which can achieve 67FPS in a specific environment. However, YOLO is not effective in detecting small and low-precision objects, its generalization ability is weak, and the error of location influences the detection effect seriously. In response to these problems of YOLO, SSD [12] combines with RPN structure to improve YOLO and achieves higher detection precision and speed. Subsequently, YOLOv2 makes a balance between precision and speed. At the same time, YOLO9000 [18] is trained in

COCO and ImageNet datasets to achieve real-time detection of over 9000 species. In 2018, Redman et al. proposes YOLOv3 [19] based on YOLO algorithm. YOLOv3 has improved a lot in speed while maintaining its original precision. At the same time, its ability to detect small objects has increased, but its performance on medium-sized and larger objects has been relatively poor.

In summary, YOLOv2 model is a better choice for real-time vehicle detection in the real scene. However, it is still insufficient for vehicle detection: (1) the precision is lower than Faster RCNN. (2) YOLOv2 is trained and tested on VOC2007 and COCO datasets, these datasets have many categories and some categories have very different shapes. So the anchors obtained by clustering the bounding boxes are not completely suitable for vehicle detection. (3) The recognition rate of difficult samples on the datasets is not high. In view of the above, based on the YOLOv2 model an improved algorithm for to detect vehicle is presented in this paper. The method introduces a new loss function, increases the number of cell units in the detection window and improves the number and sizes of anchors in the model. Finally, this model automatically learns the vehicle characteristics and realizes real-time and high-precision vehicle automatic detection and vehicle category recognition.

2 Real-Time Object Detection YOLOv2 Algorithm

Reference to the YOLO and SSD network structure, YOLOv2 designs a new classification network Darkne-19 as the basis of the network model, which is improved on the basis of YOLOv1 (YOLOv1 structure is shown in Fig. 2). Most object detection frameworks use VGG-16 as the feature extraction network before YOLOv2, but VGG-16 is more complex and requires more computation. The YOLO framework uses a network structure similar to GoogleNet, and the amount of calculation is less than VGG-16, but the precision is slightly lower than VGG-16.

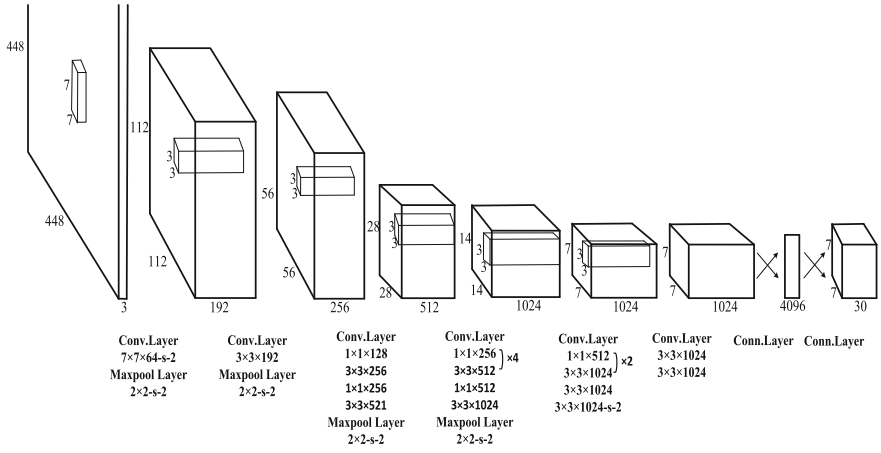

Fig. 2. YOLOv1 network structure

YOLOv2 detection network extracts features based on Darknet-19 and then modifies its network structure accordingly. The last convolution layer of the Darknet-19 is changed to three convolution layers, which the size of is 3 * 3 and the channel is 1024. In order to compress features and increase the depth of the network, the number of channels will be doubled after each pooling operation and places the 1 * 1 convolution kernel between the convolution kernels of 3 * 3. Compared with YOLOv2, the paper draws on the idea of anchor in Faster RCNN: using k-means [20] algorithm to cluster the bounding boxes in the dataset and determine the size and quantity of the anchor. In order to introduce anchor boxes to predict bounding boxes, the author removes the full connection layer from the network. The model contains only the convolutional layer and the pooling layer, so the input size can be changed at any time. During training, the model input size is changed every few rounds to make the model robust to different size images. Class prediction in YOLOv2 is no longer bound to each cell, but the anchor is used to predict categories and coordinates at the same time. Instead of direct prediction of the coordinates, the prediction of the relative offset is used to simplify the problem and facilitate the learning of network. Each time after 10 training, the model randomly chooses a new input image size to continue training. This training rule forces the model to adapt to different resolutions. The model is faster for small size, so YOLOv2 can adjust speed and precision according to the demand.

3 Improved Method Based on YOLOv2 Model

Although YOLOv2 has achieved good real-time detection results, the precision is still lower than that of the Faster RCNN and the algorithm does not fully apply to vehicle detection. For the specific problems in the application, this paper makes following improvements based on YOLOv2:

1. Constructing a new loss function to reduce the weight of easy-to-classify samples, so that the model was more focused on hard-to-classify samples during training.
2. K-means is used to cluster the bounding boxes of KITTI dataset to determine the number and size of the anchor. The anchor of YOLOv2 is determined by the VOC2007 and VOC2012 datasets clustering. These datasets are rich in categories and have different shapes. The anchor's parameters are universal, but not suitable for data in KITTI dataset. The clustering operation needed to be redone in KITTI dataset.
3. Improving the size of the network's feature map behind the multiple convolutional layers and pooling operations, which enabled more bounding boxes to be detected to reduce missed detection rates.

3.1 Construct New Loss Function

The design goal of the loss function is to achieve a good balance between the coordinates, the confidence of the bounding boxes and the category. The total loss function of YOLOv2 is shown in Formula 1:

$$
\begin{aligned}
&\lambda_{noobj}\sum_{i=0}^{l.h*l.w}\sum_{j=0}^{l.n} l_{ij}^{noobj}\left(c_i-\widehat{c}_i\right)^2+\lambda_{obj}\sum_{i=0}^{l.h*l.w}\sum_{j=0}^{l.n} l_{ij}^{obj}\left(c_i-\widehat{c}_i\right)^2\\
&+\lambda_{class}\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{l.n} l_{ij}^{obj}\sum_{c\in classes}\left(p_i(c)-\widehat{p}_i(c)\right)^2\\
&+\lambda_{coord}\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{j.n} l_{ij}^{obj}(2-w_i*h_i)\left[\left(x_i-\widehat{x}_i\right)^2+\left(y_i-\widehat{y}_i\right)^2+\left(w_i-\widehat{w}_i\right)^2+\left(h_i-\widehat{h}_i\right)^2\right]\\
&+0.01*\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{j.n} l_{ij}^{noobj}\left[\left(p_{jx}-\widehat{x}_i\right)^2+\left(p_{jy}-\widehat{y}_i\right)^2+\left(p_{jw}-\widehat{w}_i\right)^2+\left(p_{jh}-\widehat{h}_i\right)^2\right]
\end{aligned}
\tag{1}
$$

In the loss function, c_i is the real category, $\widehat{c}$ is the prediction category; (x_i, y_i, w_i, h_i) is the boundary box information of the real object, $(\widehat{x}_i, \widehat{y}_i, \widehat{w}_i, \widehat{h}_i)$ is the boundary box information of the prediction object, and $\lambda_{noobj}, \lambda_{class}, \lambda_{coord}$ are the weight parameters. The first two items calculate the IOU loss of the anchors containing the object and the non-object, that is the prediction of the confidence of the bounding boxes with or without the object; the third item represents the category prediction; the forth item represents the coordinate prediction, that is, for each objection, calculate the nearest coordinate gradient of the anchor; The last item calculates the loss of those anchors that failed to provide a valid prediction of truth in the anchors. It is somewhat similar to the difference in loss for cells that contain object and no objects. The anchors that do not provide valid predictions use the weight of scale = 0.01 to calculate the loss. The main purpose is to be more stable in the early stages of model training.

However, the YOLO algorithm is unipolar and has low precision. The reason is that the background and foreground is unbalanced. This imbalance leads to a large number of easy sample classes (including easy positive and easy negative, but mainly easy negative) during training. Although the loss of each sample class is small, because of the large number, it dominates the final loss, and results in a degenerate model that is eventually trained. Therefore, this paper prepares to change term of the third item and adds a modulation factor $(1-p_i(c))^{\gamma}$ to increase the precision. The specific loss function is as follows:

$$
\begin{aligned}
&\lambda_{noobj}\sum_{i=0}^{l.h*l.w}\sum_{j=0}^{l.n} l_{ij}^{noobj}\left(c_i-\widehat{c}_i\right)^2+\lambda_{obj}\sum_{i=0}^{l.h*l.w}\sum_{j=0}^{l.n} l_{ij}^{obj}\left(c_i-\widehat{c}_i\right)^2\\
&+\lambda_{class}(1-p_i(c))^{\gamma}\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{l.n} l_{ij}^{obj}\sum_{c\in classes}\left(p_i(c)-\widehat{p}_i(c)\right)^2\\
&+\lambda_{coord}\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{j.n} l_{ij}^{obj}(2-w_i*h_i)\left[\left(x_i-\widehat{x}_i\right)^2+\left(y_i-\widehat{y}_i\right)^2+\left(w_i-\widehat{w}_i\right)^2+\left(h_i-\widehat{h}_i\right)^2\right]\\
&+0.01*\sum_{i=0}^{l.w*l.h}\sum_{j=0}^{j.n} l_{ij}^{noobj}\left[\left(p_{jx}-\widehat{x}_i\right)^2+\left(p_{jy}-\widehat{y}_i\right)^2+\left(p_{jw}-\widehat{w}_i\right)^2+\left(p_{jh}-\widehat{h}_i\right)^2\right]
\end{aligned}
\tag{2}
$$

3.2 K-Means Dimension Clustering

Drawing on the idea of Faster RCNN, YOLOv2 introduces k-means to cluster the bounding boxes in dataset by dimension, which can determine the sizes and number of anchors. Anchors are a set of initial candidates of fixed size and aspect ratio. The quality of the anchors affects the speed of the object detection and the precision of the location of bounding box. However, because the number and width dimensions of the anchor in the Faster-RCNN are manually set, Redmon et al. proposes a method of dimensional clustering to cluster bounding boxes manually labeled in the dataset by k-means. The clustering result of VOC and COCO datasets is 5, so the number of anchors in YOLOv2 network is 5. However, this clustering result is not suitable for the KITTI datasets. In this paper, the method of dimensional clustering is used to find the proper K by adjusting the objective function $d(box, centroid) = 1 - IOU(box, centroid)$ to the minimum, variable *box* represents the information of the bounding box and *centroid* represents the information of the cluster center. Then, the k-means algorithm is used to cluster the bounding boxes corresponding to the object area in the KITTI datasets, and the optimal number of anchors and the width-height dimension suitable for the detection data set are obtained.

3.3 Grid Size Expansion

In YOLOv2, all of the images must be divided into $S * S$ grids. In different environments, the number of grids affects the precision of detection. When multiple objects are included in the images, especially small objects, the expanded network size can increase the number of objects extracted and improve recognition precision of the system. However, for sparse objects, increasing the number of grids will not greatly improve the detection effect, and will increase the complexity and calculation of the model. Therefore, selecting the appropriate parameter S is also critical for the precision and speed of the object detection. Since the object density in the image varied with the traffic flow, several sets of contrast experiments are conducted when determining the size of S. In the experiment, $S = 7$, 9 and 14 are taken, and recall rate and precision have improved when $S = 14$, so takes $S = 14$ in this paper.

4 Result and Analysis

4.1 Experimental Data

Based on YOLOv2, the train phase of vehicle detection model needs a large amount of labeled vehicle data. The larger number of training data brings the improvement of the recognition rate and the generalization performance of the model. Therefore, the establishment of vehicle detection dataset is very important for to improve YOLOv2 algorithm.

This paper uses the KITTI dataset a public computer vision algorithm evaluation dataset under the world's largest autopilot scenario. KITTI dataset contains real image data collected from urban, rural, and highway scenes with a maximum of 15 cars and 30 pedestrians per image. There are various degrees of occlusion and truncation. In the

experiment, 4000 images are selected as the training set A and 3481 images are used as the test set A* from the KITTI dataset. Considering the actual situation, the dataset was expanded by left and right flipping. In addition, in order to verify the validity of the model and the generalization ability, 7000 images are selected from the public dataset BIT-Vehicle Dataset as test set B, 7000 images under different conditions in the actual scene were collected as test set C. Table 1 shows the dataset information.

Table 1. Sample number information

Dataset	Quantity	Left and right flip	Resolution/pixel
training set A	4000	8000	1242 * 375
test set A*	3481	6962	1242 * 375
test set B	9850	–	1600 * 1200
test set C	8041	–	1920 * 1080

4.2 Experimental Configuration and Training

Our experiments are performed on a computer equipped with an Ubuntu 16.04 system. The model of the graphics card is a TITAN XP discrete graphics card produced by NVIDIA, and a GPU development package CUDA8.0 and a deep learning acceleration library cudnn5.1 are installed. Development environment is Python2.7, and the framework is Darknet-19.

The network parameters are as follows: learning-rate is 0.0001; policy is steps; batch is 128; steps respectively take 100, 20000, 35000; max-batch is 40000; scales are 10, 0.1, 0.1; momentum is 0.9 and decay is 0.0005. As shown in Fig. 3, the horizontal ordinate represents the number of iterations, ranging from 0 to 40,000 times. After the number of network iterations exceeds 15,000 times, the parameters have stabilized. During the training, the changes of average-loss, class, region- average IOU and average recall are important parameters to measure the quality of the model. In the training phase, average-loss needs to decline, and finally maintains a stable range. In addition, the value of region average IOU and average recall tend to be as good as 1, as can be seen from the figure, which basically meets the requirements.

4.3 Comparison of Different Threshold on the Dataset

The merger of sub-regions will cause redundancy in the detection window. The sub-region detection window itself is redundant, which can lead to the main objection may be repeatedly selected and affect the detection results. In order to be able to select a better object area, the non-maximum suppression algorithm (NMS) is a common method to solve this problem. The optimal threshold can be determined based on the detection effect obtained by different thresholds. As shown in Table 2, Rps/Img represents the number of bounding boxes in each sample. Seen from Fig. 4, the precision and recall rate are inversely proportional to the change of threshold value. By observing

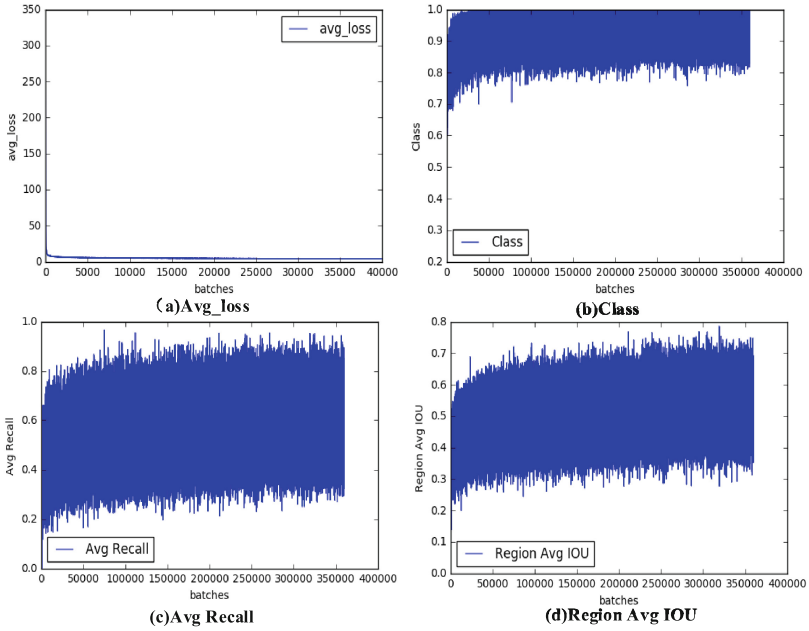

Fig. 3. Important parameters curves

the data changes of them, when the threshold is about 0.45, the recall rate is 35.08% and the precision is 93.47% and the detection effect of both recall rate and precision can be obtained. Therefore, after completing the network training by using the improved method, the threshold is set to 0.50, and the training effect is verified by using the sample of the validation set.

Table 2. The Comparison of verification results for different threshold

Threshold	Rps/Img	Recall/%	Precision/%	Recall + Precision/%
0.001	79.94	63.15	3.54	66.69
0.050	4.65	56.17	54.12	110.29
0.100	3.69	53.01	64.28	117.29
0.150	3.21	50.62	70.58	121.20
0.200	2.88	48.29	75.18	123.47
0.250	2.63	46.25	78.73	124.98
0.300	2.39	43.96	82.39	126.35
0.350	2.16	41.49	86.17	127.66
0.400	1.93	38.62	89.77	128.39
0.450	**1.68**	**35.08**	**93.47**	**128.55**
0.500	1.41	30.71	95.31	126.02
0.550	1.17	26.08	96.20	122.28
0.600	1.08	23.14	97.58	120.72

4.4 Determination of Loss Function Modulation Factor γ

Due to the unbalanced categories of the samples, the number of positive and negative samples is too large, unclassifiable and easily classifiable samples are not balanced. Therefore, this paper introduces a new loss function and adds modulation factors $(1 - p_i(c))^{\gamma}$ to the class loss to improve precision. In this case, the proposed new loss function needs to be tested to obtain a good learning rate and a modulation factor, where γ is an integer. This experiment is based on a threshold of 0.45. From Table 3, it can be seen that the different γ has different effects. When γ is 0, it means that the original loss is not changed. At this time, the recall rate is 35.08%, and the precision is 93.47%. When γ is 3, although the precision slightly decreases, the recall rate is increased by 5.23%. In this paper, γ is taken as 3, that is, the modulation factor $(1 - p_i(c))^3$ is increased.

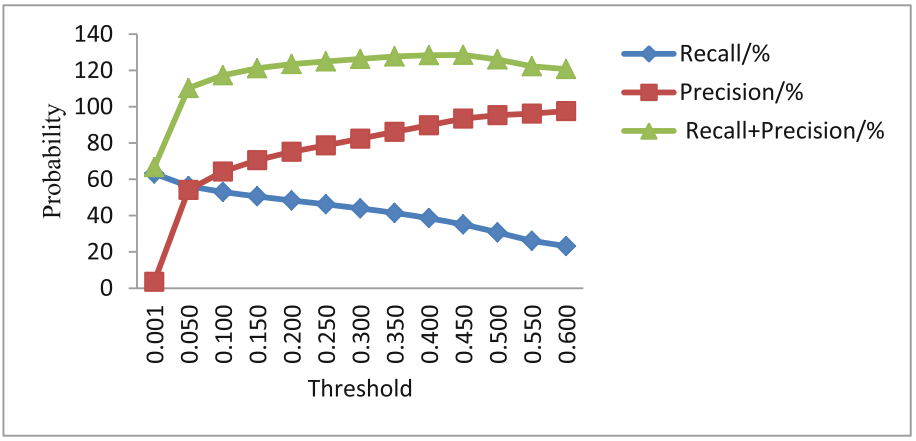

Fig. 4. The variation of precision and recall rate

4.5 The Results of the Dimension Clustering of Bounding Boxes

This paper adopts the method of dimensional clustering. The criteria evaluation is related to the IOU score. The objective function formula is $d(box, centroid) = 1 - IOU(box, centroid)$. The purpose of k-means is to adjust the objective function to the minimum, that is, the IOU is the largest. As can be seen from Fig. 5, when k is 5, the objective function reaches the minimum, so the number of anchors is 5, and k is 5 for exclusive clustering to obtain anchors. The final anchors are (0.39, 1.18), (0.69, 5.29), (0.94, 1.77), (1.78, 5.28), (3.10, 6.28). After getting the values of the anchors, the configuration file is changed to train the model, eventually increasing the precision to 95.04% and the recall rate to 41.37%.

Table 3. The model performance of different γ

γ	Recall/%	Precision/%	Recall + Precision/%
0	35.08	93.47	128.55
1	33.77	91.46	125.23
2	41.48	91.68	133.16
3	**40.31**	**93.27**	**133.58**
4	38.82	92.47	131.29

4.6 Comparison of Test Results of Different Test Sets

In order to evaluate the quality of this algorithm, we carry out a test model on three different test sets. The test results are shown in Table 4. A* is the test set in the KITTI data set, B is a test set selected in the BIT-Vehicle Dataset, and C is selected from the home-made dataset. As can be seen from the table, the algorithm has good generalization ability and the overall robustness is very good. As can be seen from Table 5, the precision and recall rate of Test Set B are particularly high. It is because BIT-Vehicle Dataset only has one vehicle per picture, and the vehicle is large and easy to identify. While Test Set C is homemade, vehicles in the dataset are relatively intensive and there are many small objects, so the recall rate is low.

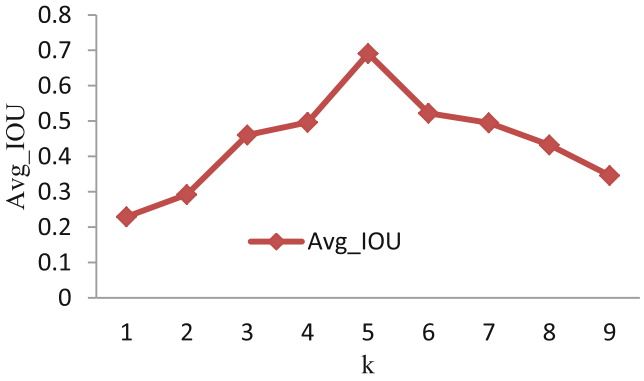

Fig. 5. The change curve of AVG-IOU

Table 4. Performance of models under different test sets

Test set	Quantity	Left and right flip	Precision/%	Recall/%
A*	6962	1242 * 375	94.45	36.33
B	9850	1600 * 1200	95.86	68.92
C	8041	1920 * 1080	92.21	62.25

Table 5. Performance of different object detection algorithms

Object detection algorithm	mAP/%	Detection speed(s/piece)
Ours	**78.35**	**0.023**
YOLOv2	75.18	0.025
YOLOv3	67.13	0.024
Faster RCNN	75.80	0.039

4.7 Comparison with Other Object Detection Algorithms

In this paper, the improved algorithm based on YOLOv2 is compared with YOLOv2, YOLOv3 and Faster RCNN. The comparison results are shown in Table 5. It can be seen from the table that the algorithm in this paper is improved in real time and precision. It can be seen from the table that detection effect in the actual application scenario is worse than that in the verification. This is because in the actual application scenario, the distant objects (such as, the small object) in the video has a poor detection effect, which needs to be improved.

5 Conclusion

In this paper, based on the YOLOv2 algorithm, an improved algorithm is proposed and its performance is validated with in many set of experiments. However, the KITTI dataset used in this paper has a maximum of 15 vehicles per image, and the number of small object samples is relatively less. As a result, the recall rate of vehicle detection is not high in the crowded roads and in distant positions. We will expand the scope of research in the next step and propose the more effective methods to increase the recall rate of small objects.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grants NO. 61502058 and 61572085, and part by Jiangsu Joint Research Project of Industry, Education and Research under Grant No.BY2016029-15.

References

1. Wang, S., Yan, J., Wang, Z.: Improved moving object detection algorithm based on local united feature. Chin. J. Sci. Instrum. **36**(10), 2241–2248 (2015)
2. Viola, P., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: IEEE CVPR, pp. 511–518 (2001)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893 (2005)
4. Lowe, D.G.: Object recognition from local scale-invariant features. In: International Conference on Computer Vision, Corfu, Greece, pp. 1150–1157 (1999)
5. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)

6. Wang, L. (ed.): Support Vector Machines: Theory and Applications. Springer, Heidelberg (2005). https://doi.org/10.1007/b95439
7. Ferreira, A.J., Figueiredo, M.A.T.: Boosting algorithms: a review of methods, theory, and applications. In: Zhang, C., Ma, Y. (eds.) Ensemble Machine Learning, pp. 35–85. Springer, Boston (2012). https://doi.org/10.1007/978-1-4419-9326-7_2
8. Felzenszwalb, P., Ross, G., McAllester, D.: Object detection with discriminatively trained part based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
9. Breiman, L.: Machine Learning. Mach. Learn. **45**(1), 5–32 (2001)
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580–587. IEEE (2014)
11. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-10578-9_23
12. Girshick, R.: Fast R-CNN. In: International Conference on Computer Vision, pp. 1440–1448 (2015)
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
14. Dai, J., Li, Y., He, K.: R-FCN: object detection via region-based fully convolutional networks (2016)
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Computer Vision and Pattern Recognition, pp. 779–788 (2016)
16. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
17. Uijlings, J.R., Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. Int. J. Comput. Vis. **104**, 154–171 (2013)
18. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger, pp. 6517–6525 (2016)
19. Redmon J, Farhadi A.: YOLOv3: An Incremental Improvement (2018). arXiv: 1804.02767
20. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)