An object detection research method based on CARLA simulation

Weihua Gao, Jiakai Tang, Taotao Wang
Wuhan University of Technology, 1178 Heping Avenue, Wuchang District, Wuhan City, Hubei Province
gwh@whut.edu.cn, jiakai_tang@whut.edu.cn, 284219@whut.edu.cn.

Abstract. At present, the research of object detection in automatic driving is mainly based on the actual automatic driving vehicle. This method is expensive, less feasible, and difficult to test. Based on CARLA, we proposed an efficient and low-cost object detection method, which can overcome the shortcomings of the actual vehicle research. In this paper, firstly, we obtained images from CARLA, and then the data set is obtained by cleaning and labeling the image. Finally, based on the data set established in this paper, we train the models of YOLOv4, CenterNet and Faster-RCNN respectively, and make a comparative study on the test set. The results show that YOLOv4 has the best detection effect without special optimization method.

1. Introduction
At present, the object detection research of autonomous driving is mainly divided into three stages: data acquisition, model training, and testing. However, automatic driving research based on actual vehicles is costly and low in feasibility. At the same time, data acquisition in limited driving scenarios leads to homogenization and simplification of data content, and low data diversity. It is inevitable to train models under such data sets. As a result, the accuracy of the object detection model is insufficient. In addition, the testing of self-driving cars is also facing big problems. According to data, self-driving cars need to be tested at least 240 million kilometers without accidents to prove that the safety of the system is not lower than that of human drivers [1-2]. Due to the particularity of the driving environment of automobiles, a more complex test environment is needed for object detection to be applied in autonomous vehicles. However, under current technical and social conditions, traditional highway tests cannot meet this demand.

The virtual simulation platform can solve the above problems very well. First of all, virtual simulation platforms are generally built based on virtual engines, which greatly reduces the cost of research. Secondly, the various parameters in the virtual simulation platform are controllable. The application of the virtual simulation platform for research can easily reproduce human daily driving behavior patterns and dangerous human driving patterns, and it can be used without safety risks, increase the operability and repeatability of the experiment [3]. Moreover, the virtual simulation platform can also supplement the deficiencies of road testing, reducing the difficulty of achieving object detection and reducing the uncertainty related to terrain by providing a safer environment and more control conditions.

In 2018, Intel Visual Computing Lab launched a driving simulator for urban automatic driving research, CARLA [4], which can well meet the requirements of object detection research in automatic driving. Firstly, the implementation of CARLA is based on the open source layer of UE4, which provides the most advanced rendering quality and realistic physical characteristics, greatly improves
the authenticity and flexibility of the virtual scene, and ensures the quality of the data set used for object detection and the effectiveness of subsequent tests. Secondly, CARLA hides the underlying driver of the sensor and the analysis of the bus protocol inside the car. Developers only need to use the external Python API to realize the flexible configuration of the sensor, which provides a very convenient condition for the research of object detection in automatic driving. In addition, CARLA provides many free digital assets, such as city maps, various types of vehicles, pedestrians, traffic signs. These digital assets have been carefully designed by CARLA developers to ensure visual fidelity. According to the demand, we can fully evaluate the performance of the object detection model by simulating various and complex traffic environments in CARLA. More importantly, CARLA simulator can repeatedly repeat dangerous situations to help driving system learn. Compared with other simulators, CARLA overcomes the defects of closed source, virtual reality, unrealistic system, unfriendly system, changing driving conditions, setting up motion models in scene, and failing to provide technical information needed for automatic driving system learning.

Therefore, we proposed an object detection method based on CARLA simulation. Firstly, the sensors in CARLA simulation are called to obtain a large number of images in the actual driving scene. Then label the image and make the data set. Finally, the data set is divided to train the object detection model based on convolutional neural network.

2. Image dataset obtained from CARLA

2.1. Image acquisition
CARLA provides many easy-to-use sensors, such as RGB cameras, depth cameras, and the sensor’s Python API is open source. Developers do not need to consider the underlying driver of the sensor and the analysis of the car’s internal bus protocol. By writing the corresponding scripts of Python can get the picture in the specified format. CARLA also allows developers to add any number of vehicles and pedestrians to the Python script to simulate different road conditions. In addition, CARLA has built-in a variety of weather conditions including sunny days, rainy days, and strong winds. Developers only need to make corresponding configurations in the Python script to simulate different weather conditions and various road conditions.

CARLA allows to set the motion state of the car. We install three RGB cameras to a self-driving car. The camera position parameters are shown in Table 1:

| Camera ID | x   | y   | z   |
|-----------|-----|-----|-----|
| 1         | 0.5 | 0   | 2.4 |
| 2         | 0.5 | -5  | 2.4 |
| 3         | 0.5 | 5   | 2.4 |

In this table, x represents the front and rear position of the camera relative to the car, y represents the left and right position of the camera relative to the car, and z represents the vertical position of the camera relative to the car. These three positions correspond to the front of the car, the front left, and the front right. They are used to obtain images from different angles, and the size of the collected images is set to 1080*720. Examples of the collected image is shown in Figure 1:

![Image Examples](image1.png)
We got 3000 pictures in total, but there were great similarities between these pictures. If these images are directly input into the convolutional neural network without processing, it will lead to the over-fitting phenomenon of the network, and ultimately affect the detection effect. So we need to clean the image.

2.2 Image cleaning
We use SSIM to measure the similarity of two pictures, and the picture with higher SSIM is deleted to achieve the purpose of data cleaning. SSIM is the structural similarity, which evaluates the similarity of two input images through the brightness, contrast and structure of the image. The specific calculation method is shown in formula (1):

$$SSIM(x, y) = \left[ I(x, y) - c(x, y) \right] \cdot s(x, y)$$

$I(x,y)$ is used to calculate the luminance, $c(x,y)$ is used to calculate the contrast, $s(x,y)$ is used to calculate the structure, the calculation methods are respectively as formula (2), (3), (4) as shown:

$$I(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

$\mu_x$ is the average of $x$, $\sigma_x^2$ is the variance of $x$, $\mu_y$ is the average of $y$, $\sigma_y^2$ is the variance of $y$, $\sigma_{xy}$ is the covariance of $x$ and $y$, $C_1$, $C_2$, and $C_3$ are three constants. To avoid system errors caused by the denominator being 0. This article specifies $\alpha=\beta=\gamma=1$, $C_3=C_2/2$, and the calculated formula is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

We use formula 5 to calculate the SSIM of any two pictures in the data set. If the SSIM is greater than 0.7, one of them will be deleted. In the end we got 1512 pictures. which can be used to train the object detection model. Some examples are shown in Figure 2:

![Image examples of data set](image.png)

2.3 Image annotation
We use LabelImg to label the images. LabelImg is a visual semi-automatic labeling tool, which can easily draw rectangle on the object, and has a label memory function. After an image is annotated, an xml annotation file will be automatically generated.

We sets up a total of four classes, which correspond to cars, pedestrians, traffic lights, and motorcycles that appear frequently in actual driving scenes.

3. Object detection
We use the object detection algorithm based on convolutional neural network to train the high-quality model. The selected algorithms are: YOLOv4[5], CenterNet[6] and Faster-RCNN[7].
3.1. **YOLOv4**

YOLOv4 was proposed by Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao and others in April 2020. It is another major improvement of the YOLO series network. Compared with YOLOv3, YOLOv4 has higher detection accuracy and faster detection speed.

The author of YOLOv4 divides the object detection network into Input, Backbone, Neck and Head, which correspond to the input layer, feature extraction layer, middle layer and output layer of the network respectively. YOLOv4 uses Mosaic data augmentation, CmBN, SAT training and other tricks in the input layer to reduce training costs. It uses CSPDarknet53 as the backbone structure of the network, and adds the SPP module after the backbone network to increase the network’s receptive field and separate context features. Compare to the FPN used in YOLOv3, YOLOv4 uses PANet for multi-scale feature fusion. The output layer still retains the structure of YOLOv3. The YOLOv4 network structure is shown in Figure 3.

![Figure 3 YOLOv4 network structure](image)

3.2. **CenterNet**

CenterNet was proposed by Xingyi Zhou, Dequan Wang, and Philipp Krahenbuhl in April 2019. It is an improved algorithm for CornerNet. Different from the classic object detection algorithm, CenterNet is based on the Anchor-Free method for object detection. The algorithm mainly obtains the object by predicting the position of the center point of the object. It is more accurate and faster than the two-stage object detection algorithm. The network structure of CenterNet is shown in Figure 4.

![Figure 4 CenterNet network structure](image)

3.3. **Faster-RCNN**

Faster-RCNN is proposed by Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun in June 2016. This algorithm is an improvement and optimization of the Fast-RCNN network, and excellent methods such as RPN and Anchor mechanism are proposed, which have been significantly improved in speed and accuracy.
In order to solve the time-consuming problem of the Selective Search algorithm in Fast-RCNN, the Faster-RCNN algorithm proposed an RPN network to generate Region Proposal. At the same time, RPN shares convolution with Fast RCNN, which greatly reduces the amount of calculation and improves network training and testing speed. The network structure of Faster-RCNN is shown in Figure 5.

**4. Experiment and result analysis**

We use the object detection algorithm based on convolutional neural network to train the high-quality model. The selected algorithms are YOLOv4, CenterNet and Faster-RCNN.

**4.1. Evaluation index**

We divided 1512 pictures into training set, validation set and test set according to 8:1:1, of which 1209 are training set, 151 are validation set, and 152 are test set. For the detection results, this paper uses the classic mAP (mean Average Precision) and FPS (Frames Per Second) in the field of object detection for comparative evaluation.

**4.2. Experimental equipment**

The experimental equipment used in this article is a single GeForce GTX 1080Ti graphics card with 11G video memory. The computer is equipped with CUDA10.0 package; CPU is Intel i5-9400F, 16G running memory. The code running environment is Python 3.6.

Among the three algorithms, YOLOv4 uses the keras framework and tensorflow as the backend, and the frameworks used by CenterNet and Faster-RCNN are both PyTorch.

**4.3. Parameter settings**

**4.3.1. YOLOv4**

Firstly, we performs K-Means clustering on the data set to obtain anchors. The anchors are 7, 21, 11, 27, 15, 38, 21, 23, 23, 54, 36, 32, 53, 51, 82, 90, 158, 191.

We used transfer learning to train model. Firstly we frozen the Backbone, fine-tuning the upper layer weights, and then trained the entire network. The parameters in the frozen training phase were set to: learning_rate_base = 1e-3, batch_size = 8, Freeze_epoch = 50, warmup_learning_rate=1e-4, min_learn_rate=1e-6.

The parameters used in the global training phase were: learning_rate_base = 1e-4, batch_size = 4, Epoch = 500, warmup_learning_rate=1e-5, min_learn_rate=1e-6.

The optimizer used in both stages is Adam.

**4.3.2. CenterNet**

CenterNet is a object detection network without anchor, that is, there is no need to specify anchor during training. We used the classic "two-step method" for training, that is, first obtain the pre-trained model of the Hourglass backbone network on ImageNet, and fine-tune the entire network on our data set. The parameters used for training were learning_rate=1.25e-4, batch_size=16, master_size=15, Epoch=600. The optimizer used is Adam.
4.3.3. Faster-RCNN
We set the anchors of Faster-RCNN to be 8, 16, 32, and the aspect ratio is 0.5, 1, and 2 respectively. At the same time, the VGG16 network trained on ImageNet was used as Backbone. The parameters used in the training phase were learning_rate=0.01, lr_decay_step=50, lr_decay_gamma=0.1, epochs=400. We used stochastic gradient descent, so batch_size=1, and the optimizer used is SGD.

4.4. Result analysis
Three networks of YOLOv4, CenterNet and Faster-RCNN were trained in the given environment. After training, we used these three models to test on the test set. Examples of test results are shown in Figure 6, Figure 7, and Figure 8.

Calculating the mAP and FPS of the three models respectively, we got Table 2.

| Algorithms      | AP-person | AP-car | AP-trafficLight | AP-motorcycle | mAP   | FPS  |
|-----------------|-----------|--------|-----------------|---------------|-------|------|
| YOLOv4          | 75.37%    | 88.41% | 66.56%          | 33.48%        | 65.96%| 16.31|
| CenterNet       | 49.56%    | 83.14% | 50.96%          | 35.68%        | 54.83%| 12.25|
| Faster-RCNN     | 53.12%    | 69.90% | 58.28%          | 29.37%        | 52.67%| 7.43 |

From Table 2, we can find that among the three models, YOLOv4 has the highest accuracy and fastest speed, mAP reaches 65.96%, FPS is 16.31. Followed by CenterNet, mAP is 54.83%, FPS is 12.25. Faster-RCNN is the worst, its mAP is only 52.67%, FPS is 7.43.

YOLOv4, as a newest object detection algorithm, has the best detection effect in this article. YOLOv4 not only can accurately identify large objects, but also overcomes the shortcomings of the previous single-stage object detection algorithms for small object detection, which is sufficient to prove that many tricks adopted by YOLOv4 are effective. However, limited by the size of the data set in this article, we has not trained a very good model, and compared with large objects, YOLOv4 still has insufficient detection of small objects. The detection accuracy still have room for improvement. We will continue to study
5. Conclusion
Based on CARLA, we conducted simulation research on object detection, overcome the shortcomings of the previous automatic driving research based on actual vehicles, and proposed an efficient and low-cost object detection research method. We obtained the images required for object detection training from the CARLA, and got the data set after cleaning and labeling. The models of YOLOv4, CenterNet, and Faster-RCNN were respectively tested on the test set. It was found that YOLOv4 has the best detection effect, with the highest detection accuracy and fastest detection speed among the three models. However, due to some defects in the data set, YOLOv4 did not achieve the desired effect. We will continue to expand the data set based on CARLA, and conduct improved research on YOLOv4 to train a better object detection model. At the same time, we will continue to follow up the most advanced object detection algorithms abroad, and apply it to CARLA for research.

Acknowledgments
This paper is supported by national innovation and entrepreneurship training program for college students S202010497102.

At the point of finishing this paper, I’d like to express our sincere thanks to Dr. Yufen Sun, who has given us so much useful advice on my writing, and has tried her best to improve this paper.

References
[1] CHRISTENSEN A, CUNNINGHAN A, ENGELMAN J, et al. Key Considerations in the Development of Driving Automation Systems[C]/NHTSA.24th International Technical Conference on the Enhanced Safety of Vehicles (ESV). New York:NHTSA, 2015:1-14.
[2] BENMIMOUN M. Effective Evaluation of Automated Driving System[J]. SAE Paper, 2017-01-0031.
[3] Johan Olstam, Stéphane Espié, Selina Måardh, Jonas Jansson, Jan Lundgren. An a-lgorithm for combining autonomous vehicles and controlled events in driving simulator experiments[J]. Transportation Research Part C, 2011, 19(6): 1185-1201
[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun. CARLA: An Open Urban Driving Simulator [C]. Proceedings of the 1st Annual Conference on Robot Learning. PMLR 78:1-16, 2017.
[5] YOLOv4: Optimal Speed and Accuracy of Object Detection. Bochkovskiy A, Wang C Y, Liao HY M., 2020
[6] Zhou, Xingyi, Wang, Dequan, Krähenbühl, Philipp. Objects as Points[J].
[7] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // International Conference on Neural Information Processing Systems, 2015:91-99.