Vol 4 | Issue 1 | Pages 122-130

Journal of Robotics and Automation
ISSN: 2642-4312

Research Article

DOI: 10.36959/673/363

Optimized Convolutional Neural Network-Based Object Recognition for Humanoid Robot

Ming Xiao, Nanfeng Xiao, Mengjun Zeng and Qunyong Yuan*

School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

Abstract

In recent years, many researchers have proposed a series of algorithms based on convolutional neural networks and achieved good performances in the field of object detection and recognition. For humanoid robots, they are designed to assist or replace people in completing a series of anthropomorphic tasks, and their ability to recognize and grasp surrounding objects is the most basic requirement. Therefore, this paper proposes an algorithm based on the optimized convolutional neural network (CNN) and can be used to detect and classify the objects by humanoid robots. Particularly, Faster region-based CNN (Faster R-CNN) is used for detecting the objects and the original network is refined by feature concatenation strategy. The feature fusion layer is also proposed to make the feature map contain more information, the improved non-maximum suppression (NMS) algorithm is adopted to solve the overlapping object detection problem, and some layers are also modified in the original network to get faster speed and smaller network size. The recognition and grasping performance of the humanoid robots are tested by the proposed algorithms in a virtual simulation environment, and the experiment results show that the proposed algorithms are feasible and perform well.

Keywords

Deep learning, Object detection, Humanoid robot, CNN

Introduction

Object recognition is one of the basic abilities of the humanoid robots, it allows the humanoid robots to recognize nearby objects and perform many anthropomorphic tasks, which can help humans in the repetitive work or the dangerous operational environments [1]. It is well known that humans can easily distinguish objects and perform various tasks through their vision; however, without the vision system and the object detection and classification algorithm, it is difficult for the humanoid robots to complete the object recognition task. Therefore, the key to improving the other capabilities of the humanoid robots is to first develop high-performance algorithms for their object detection and classification system.

As the cornerstone of image understanding and computer vision, object detection is the basis for more complex and higher-level visual tasks, such as segmentation, scene understanding, target tracking, image description, event detection, and activity recognition. Object detection is necessary for artificial intelligence and information technology, including robotics, autonomous driving, human-computer interaction, intelligent video surveillance, surface defect detection [2], ship and plane detection in the optical remote sensing images [3,4], augmented reality [5], and has been studied for years. The object detection tasks usually contain two steps: Use a bounding box to locate the objects and then refine the box and classify the object [6]. With the popularity of deep learning and continuous improvement of computer performance in recent years, deep neural networks have shown better results than the traditional object detection algorithms, such as edge detection [7], gradient matching [8], etc. Recently, CNNs have achieved great success in the fields of object detection [9], recognition [10], object grasping control [11], robotic manipulator motion path planning [1], etc. The most representative algorithms are Faster R-CNN [9] and Single Shot multibox Detector (SSD) [12], they can learn the target object's features in the datasets with built-in neural network and improve their performance gradually through training. Faster R-CNN and SSD represent two typical target detection approaches: Two-stage approaches and one-stage approaches. For the two-stage approaches, they divide the object detection process into two phases: Using a feature extracting network to extract the feature information of the target objects and roughly locate objects, and then refine the locations

***Corresponding author:** Ming Xiao, School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

Accepted: February 25, 2020

Published online: February 27, 2020

Citation: Xiao M, Xiao N, Zeng M, et al. (2020) Optimized Convolutional Neural Network-Based Object Recognition for Humanoid Robot. J Robotics Autom 4(1):122-130

and classify the objects. For the one-stage approaches, they can locate and predict the target objects directly through the network [13].

Different from single object detection task, the object recognition applied to the humanoid robots has strict requirements on speed, accuracy and environmental adaptability. Although there are plenty of researches on the robotic object detection and various applied object detection methods on the single or the multi-fingered industrial robotic hands, most robotic hands cannot achieve the degrees of freedom (DOF) of a human and are much less flexible than a human. Therefore, this paper uses Faster R-CNN as the basic method and improves the speed and accuracy of object detection. The algorithm is combined with a humanoid robot to test the algorithm performance in the experiments.

Related Work

There have been many studies on the grasping objects of the humanoid robots. However, in real-world environments, if there are other distracting objects in the humanoid robots' view field, it is a challenge to accurately detect and locate the target objects. In this section, the existing methods of object detection and localization for the humanoid robots are discussed and summarized as follows.

Two-stage object detection methods

The goal of the generic object detection is to determine the locations and classifications of all the target instances in a natural image based on a large number of predefined categories, which is the most fundamental and challenging issues in machine vision. The deep learning technology emerged in recent years is a powerful method for learning feature representation directly from data, and has brought significant breakthroughs for object detection.

In 2012, Krizhevsky, et al. [14] proposed the deep convolutional neural network Alexnet, and achieved record-accurate object recognition accuracy in the Large-Scale Visual Recognition Challenge (ILSVRC) competition. Since then, many target detection algorithms based on deep learning have emerged. In 2014, Ross Girshick, et al. proposed a region-based CNN(R-CNN), which uses the selective search algorithm to generate about 2000 region proposals from the bottom to the top of the input image, and then warp these region proposals to 227 × 227 and input them into the CNN, and the output of the fc7 layer of the CNN is taken as features of these proposals, and they are trained by the support vector machine (SVM) for classification. R-CNN is slow to train, consumes a lot of computing resources, and has limitations on the size of the input image. Kaiming He, et al. proposed SPP-net [15] which introduced a spatial pyramid pooling (SPP) layer to remove the fixed-size constraint of the network, thus making the speed of SPP-net several times higher than R-CNN. In 2015, Ross Girshick proposed fast region-based CNN (Fast R-CNN) [16], which aims to complete the classification and positioning task with two parallel fully connected layers. Fast R-CNN combines the essence of R-CNN and SPP-NET, and introduces multi-task loss function, which makes the training and testing of the entire network very convenient. Nevertheless, the fast R-CNN still uses the selective search method to generate region proposals and this step is the most time-consuming part, and the whole network is not an end-to-end network. S Ren, et al. proposed Faster R-CNN [9] algorithm, which proposed a region proposal network (RPN) to replace the selective search method. The RPN generates about 300 region proposals for each input image and shares the parameters of the convolutional layer, thus greatly improving the speed of the object detection. Based on the idea of Faster R-CNN, many two-stage object detection methods have been proposed these years, such as Mask RCNN, R-FCN, and so on.

One-stage object detection methods

Although the two-stage object detection methods like faster R-CNN have great advantages in object recognition accuracy, they are not satisfactory in some real-time recognition situations. A new algorithm named You Only Look Once (YOLO) was developed by Joseph Redmon, et al. [17] in 2016. YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Bounding boxes predictions and classifications are done directly through a single neural network, thus making it much faster compared to the industry standards, reaching 45 frames per second in the real-time object detection tasks. Many other one-stage detection methods are proposed after YOLO, such as YOLO v2, YOLO 9000 [18], YOLO v3 [19], SSD [12], etc. Compared with the two-stage methods, the one-stage methods such as YOLO do have a big advantage in the real-time processing capabilities, but the accuracy of the object recognition is relatively low. Therefore, there is a necessity to strike a trade-off between the two-stage methods and the one-stage methods.

Researches on robot recognition and object detection

In recent years, many CNN-based studies have focused on such problems as small object detection [6], face detection [20], crowd counting [21,22], traffic sign detection [23], and car detection [24]. In the meantime, there are increasing research on the deep learning application for the humanoid robot's object recognition, grasp detection, etc. [25]. Lenz, et al. [26] considered the detecting robotic grasps in an RGB-D view of a scene containing objects and applied an approach to avoid time-consuming feature design by the manual work. Levine, et al. [11] described a learning-based approach to hand-eye coordination for the robotic grasp from the monocular images. Pinto and Gupta [27] used large-scale datasets and multi-stage training for the grasping task and get state-of-the-art performance on generalization to the unseen objects. Noda, et al. [28] proposed a computational framework enabling the integration of the sensory-motor on the time-series data and the self-organization of multi-modal fused representation based on a deep learning approach. Yu, et al. [29] presented a vision-based robotic grasping system that could recognize different objects as well as estimate their poses by using a deep learning model, finally grasped them and moved to a predefined destination. Zhang, et al. [30] introduced a machine learning-based system for controlling a robotic ma-

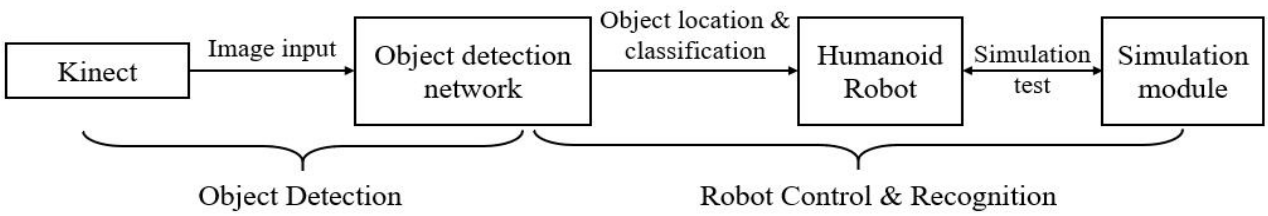

Figure 1: The proposed system architecture for the object detection of humanoid robots.

Table 1: PC specifications.

Name	Detail
CPU	Intel(R) Core(TM) i7-7820X CPU @ 3.60 GHz
GPU	Nvidia GeForce GTX 1080 Ti
Memory	16GB

nipulator with visual perception. Gu, et al. [31] presented an asynchronous deep reinforcement learning approach, which could be used to learn the complex robotic manipulation skills from scratch on the real physical robot manipulator.

System Overview

This paper proposes a system architecture for the object detection of the humanoid robots, as shown in Figure 1, which contains an object detection module and a humanoid control module as well as a recognition module. The object detection module is designed to quickly and accurately detect the target object in the input image, while the humanoid robot control module and the recognition module use the former to perform object recognition task and other operations such as grasping, sorting, etc.

For the object detection module, Kinect sensor captures the work area image of the humanoid robot, that is, the image of the object to be recognized, and the image is fed into the pre-trained object-detection network, and then the bounding box and the category information of the related object can be obtained. These results are input to the humanoid robot control module.

According to the object detection result, the object and the humanoid robotic hand are simulated in the simulation module and the humanoid robotic hand can move to the top of the detected object. Finally, the other tasks such as grasping can be completed. The simulation module combines the virtual humanoid robotic hand and the object detection program, the images captured by Kinect are fed into the program and the results are inputted into the virtual hand operating system for motivation.

Hardware specifications

The object detection module contains a Kinect v2 and it is connected to a PC. The specifications of the PC are given in Table 1. The humanoid robot control module and the recognition module is mainly composed of a pair of the humanoid robotic hands, each of them has 21 DOFs just like that of the human hand structure, and each robotic hand has a total of

Table 2: Rotation angle (degrees) range of each finger.

Knuckle	Thumb	Index	Middle	Ring	Little
Base	[0, 80]	[0, 40]	[0, 20]	[0, 20]	[0, 20]
First	[0, 80]	[0, 90]	[0, 90]	[0, 90]	[0, 90]
Second	[0, 90]	[0, 108]	[0, 108]	[0, 108]	[0, 108]
Third	[0, 80]	[0, 90]	[0, 90]	[0, 90]	[0, 90]

22 steering gears. The size of the robotic hand is designed according to the normal person's proportion. The five fingers are also driven independently by the respective motors. Each finger has four knuckles, the base knuckle and the first to third knuckles. The rotation angle range of each finger is shown in Table 2.

Humanoid robotic hand kinematics

The humanoid robotic hand is composed of rotational gears connected to each other, and there are two kinds of kinematics for a humanoid robotic hand: Forward kinematics and inverse kinematics. The forward kinematics determines the end position of the manipulator based on the structure of the given humanoid robotic hand and the rotation angle of each joint. The D-H representation (Denavit-Hartenberg Convention) is a matrix representation of the connection between the various joints of the humanoid robotic hand. According to the D-H notation, assign a reference coordinate system to each joint, then determine the transformation matrix between any two adjacent coordinate systems, and finally write the total transformation matrix of the robot base coordinate system to the end effector (that is, humanoid robotic hand). The transformation matrix is expressed as:

$$A_i = \mathrm{Rot}_Z(\theta_i)\,Trans_Z(d_i)\,Trans_X(a_i)\,Rot_X(\alpha_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where θ_i represents the angle from x_{i-1} to x_i around Z_{i-1}, and d_i represents the distance from x_{i-1} to x_i along Z_{i-1}. A_i represents the distance from Z_{i-1} to Z_i along x_{i-1}, and α_i represents the angle of rotation Z_{i-1} to Z_i about the x_i axis. The main goal of the inverse kinematics problem is to calculate the angle of each joint of the humanoid robotic hand given the structure of the humanoid robotic hand and the end pose. The inverse

kinematics is more important than the positive kinematics because it allows the end of the robotic hand to move to a specified pose to perform a grab job.

Object Detection and Simulation

In this part, the object detection method based on Faster R-CNN is introduced and the simulation environment is setup. Faster RCNN is one of the most representative methods in the object detection field [32]. Considering the limited computational resources of the humanoid robots in the experiments and the disadvantages of the object recognition in the original network, such as the low recognition rate of the small object detection [33,34], the original network is optimized to balance the computational cost and the detection accuracy.

For the traditional Faster R-CNN method, the RoI (region of interest) pooling layer uses those feature maps extracted from the last layer of the convolutional network to generate the region features. Because the deeper layers get wider receptive fields in the convolutional networks, this may cause some important features to be ignored, resulting in the grosser granularity. Therefore, considering the balance of computing resource consumption and object detection accuracy, inspired by [35,36], the feature maps of the con4_3 layer and the conv5_3 layer are combined in the VGG16 network, and concatenate the results in the input to the ROI layer. Specifically, the result of the conv4_3 layer is normalized and then concatenated with the result of conv5_3 and rescaled to the size of the original feature map, and then a 1 × 1 convolution is used to match the original network channels. For example, if the input image size is 224 × 224, the map of the con4_3 layer's feature would be 14 × 14, therefore the feature map of the conv5_3 layer would be up-sampled to match the size of 14 × 14. The architecture of the VGG16 network is shown in Figure 2, and the specific structure proposed above is shown in Figure 3.

Faster R-CNN is mainly composed of four parts: Convolution layer, which is used to extract the features of input images and obtain feature maps; RPN network layer, which is a substitute for selective search in fast R-CNN and is used to

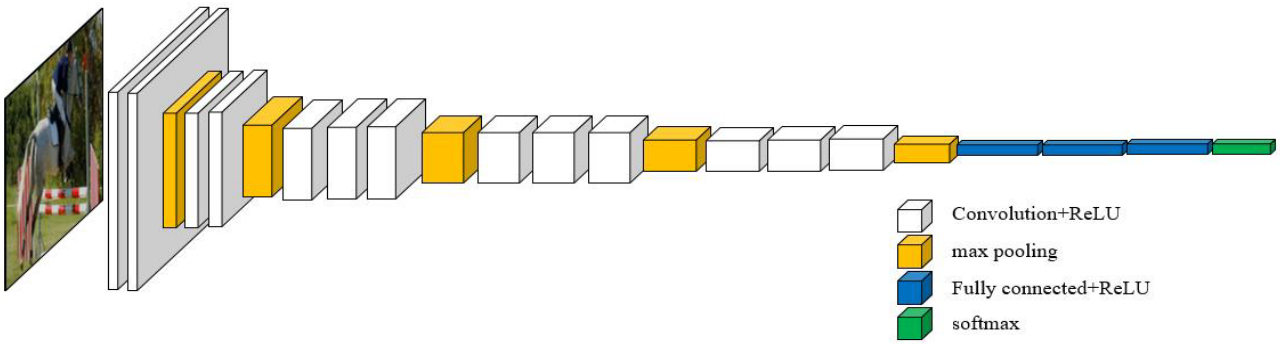

Figure 2: The architecture of VGG16 network.

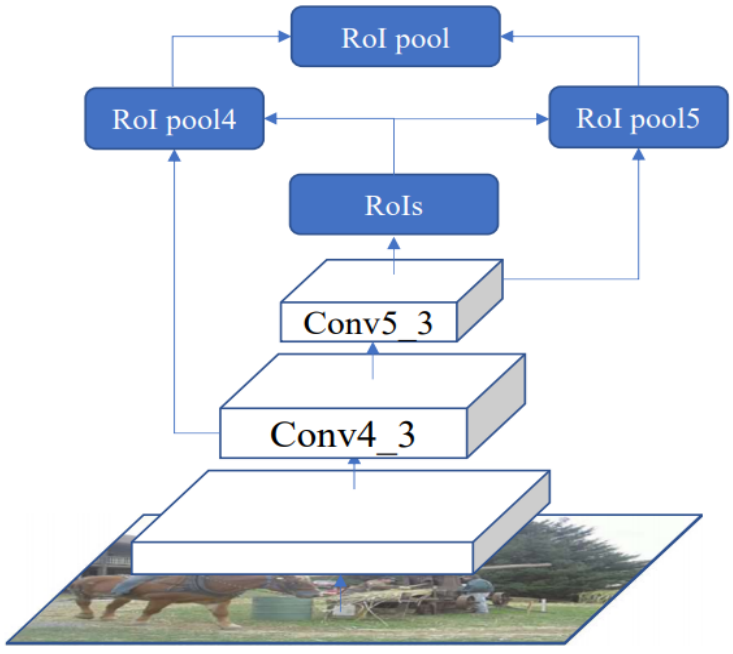

Figure 3: The concatenation architecture of the convolution layers.

recommend the candidate regions and generate their scores, using a non-maximum suppression strategy to select the candidate regions with scores greater than 0.7 and scores less than 0.3; RoI pooling layer, like Fast R-CNN, converts the different sized inputs to the fixed-length outputs; classification and regression layer, which is used to get the classification score of the regions and regress bounding box. The RPN network layer is trained end-to-end and its loss function is defined as:

$$L\left(\{p_i\}+\{t_i\}\right)=\frac{1}{N_{cls}}\cdot\sum_i L_{cls}\left(p_i,p_i^*\right)+\frac{\lambda}{N_{reg}}\cdot\sum_i p_i^*\cdot L_{reg}\left(t_i,t_i^*\right) \quad (2)$$

$$L_{cls}\left(p_i,p_i^*\right) = -\log\left(P_{p_i^*}\right) \quad (3)$$

$$L_{reg}\left(t_i,t_i^*\right) = smooth_{L1}\left(t_i-t_i^*\right) \quad (4)$$

$$smooth_{L1}(x) = \begin{cases}0.5x^2, & if\ |x|<1\\ |x|-0.5, & otherwise\end{cases} \quad (5)$$

where, i represents the i-th anchor's point, P_i is the probability of the anchor is an object, p_i^* is 1 if the anchor is labelled positive, or is 0 if the anchor is negative, t_i represents the predicted bounding box's four parameterized coordinates and t is the ground-truth box associated with the positive anchor. L_{cls} is log loss, N_{cls} and N_{reg} represent the normalization values, N_{cls} values 256 and N_{reg} values 2400 as default, but λ is adopted to balance regression and classifier accordingly. The regression calculation of the anchor box to the adjacent ground truth value bounding box is specified as follows:

$$\begin{cases}t_x = \frac{x-x_a}{w_\alpha}, t_y = \frac{y-y_\alpha}{h_\alpha}, t_w = \log\frac{w}{w_\alpha}, t_h = \log\frac{h}{h_\alpha}\\ t_x^* = \frac{x^*-x_\alpha}{w_\alpha}, t_y^* = \frac{y^*-y_\alpha}{h_\alpha}, t_w^* = \log\frac{w^*}{w_\alpha}, t_h^* = \log\frac{h^*}{h_\alpha}\end{cases} \quad (6)$$

where, x and y represent the coordinates of the proposed region, and w and h express the region's width and height. For a specific anchor, x represents the predicted bounding box, x_a expresses the anchor and the ground truth box is represented by x^*.

The non-maximum suppression (NMS) algorithm is used to find the best bounding box for each object and eliminate redundant bounding boxes. The NMS algorithm select the bounding box A with the highest score from the generated series of bounding boxes B, puts A in the final detection result D, removes A from B and sets an Intersection over Union (IoU) threshold, the remaining boxes with the IoU of M greater than the threshold is removed from B. The NMS algorithm repeats the above operation by selecting the bounding box with the highest score in the remaining boxes B until B is empty and finally outputs the result R. The NMS algorithm can be defined as follows [6]:

$$s_i = \begin{cases}s_i, IoU\left(M,B_i\right)<T\\ 0, \mathrm{IoU}\left(M,B_i\right)\geq T\end{cases} \quad (7)$$

where T is the threshold of IoU defined above. It can be known that boxes with IoU higher than the threshold is directly removed from the remaining boxes B, which may result in some overlapping objects being missed. To solve this problem, an optimized NMS algorithm is used, which is called soft-NMS and holds the boxes with IoU higher than the threshold, and sets them to a lower score instead of deleting them directly. The function is defined as follows [6]:

$$s_i = \begin{cases}s_i, IoU\left(M,B_i\right)<T\\ s_i\cdot\left(1-IoU\left(M,B_i\right)\right), \mathrm{IoU}\left(M,B_i\right)\geq T\end{cases} \quad (8)$$

where T is the threshold of IoU, when the IoU of M is greater than the threshold T, the score of the bounding box is linearly attenuated. In this case, the bounding box that is close to M is attenuated to a large extent, and the bounding box far from M remains unaffected.

In order to further verify the performance effect of the algorithm applied in the humanoid manipulators, the simulation software V-REP and 3D max were used to simulate the whole system and a simulation test platform including image acquisition, object recognition and corresponding humanoid manipulators' motion was built. V-REP is a strong 3D integrated environment for robots and has several universal calculation modules (inverse kinematics, physics, dynamics, collision detection, minimum distance calculation, path planning), distributed control architecture (control scripts of unlimited number, thread or non-thread), and several extension mechanisms (plug-in, client application program and so on.) [37,38]. V-REP also supports many programming languages such as C++, Python, Matlab and so on. We use the built-in Kinect model of V-REP as the sensor of the virtual environment, use 3D max as an auxiliary tool to design different objects, and then import them into V-REP as the target object for recognition. Our object detection models are implemented on the Google TensorFlow platform using Python and the results can be used by the simulation platform by Python APIs built-in the V-REP software. The platform is shown in Figure 4.

Experiment Schemes and Results

In order to verify the effects and the performance of the optimized algorithm proposed in this paper, three datasets are mainly used to conduct the experiments, namely PASCAL VOC 2012 [39], MS COCO [40] and another self-built dataset containing many small objects.

The PASCAL VOC 2012 dataset contains 20 object categories including people, animals (such as cats, dogs and birds), vehicles (such as cars, ships and planes), furniture (such as chairs, tables, and sofas) and is one of the most widely used benchmark datasets for generic object detection.

MS COCO is a well-known large-scale object detection, segmentation, and captioning dataset built by Microsoft. COCO has several features including object segmentation, 330K images (> 200 K labeled), 80 object categories, 5 captions per image, etc.

Our self-built dataset consists of some images of the Cornell Grasping Dataset and photos taken by ourselves. There are many small objects in these pictures, which is convenient for us to test the performance of the algorithm.

The VGG16 model pre-trained on ImageNet is em-

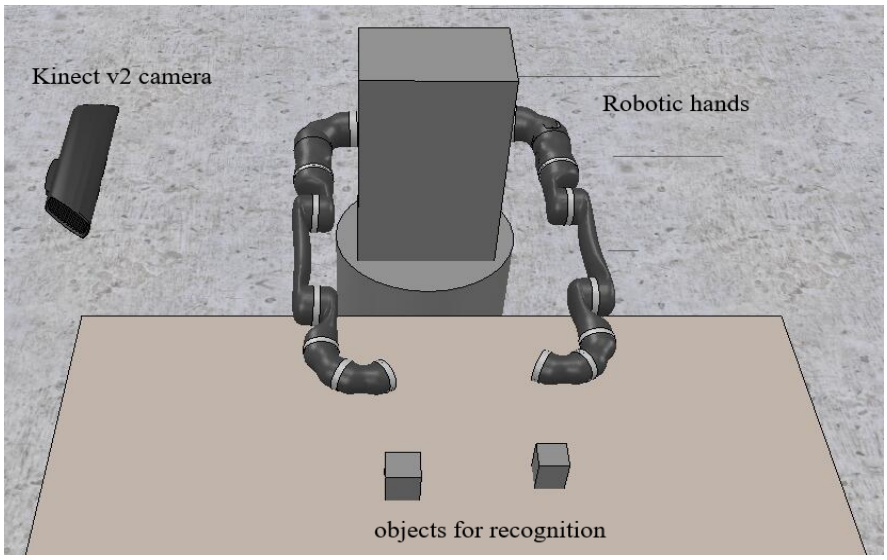

Figure 4: Simulated system of the humanoid robot manipulators platform.

Table 3: The results for the networks VGG16net on MS COCO dataset.

Metrics	COCO VGG 2000 Proposals	COCO VGG 1800 Proposals	COCO VGG 1000 Proposals
AP	0.428	0.420	0.340
AP.50	0.590	0.579	0.465
AP.75	0.472	0.453	0.380
APS	0.254	0.237	0.172
APM	0.433	0.421	0.290
APL	0.511	0.490	0.383
ARS	0.451	0.438	0.391
ARM	0.592	0.586	0.520
ARL	0.650	0.647	0.578

Where, AP: Average Precision; AP.50: Average Precision at IoU: 0.5; AP.75: Average Precision at IoU: 0.75; APS: Average Precision for small objects; APM: Average Precision for medium objects; APL: Average Precision for large objects; ARS: Average Recall for small objects; ARM: Average Recall for medium objects; ARL: Average Recall for large objects.

ployed as the backbone network. The code is based on the official Faster R-CNN code implemented on the TensorFlow platform. And the backbone networks are pre-trained on the large dataset ImageNet and then fine-tuned on the detection dataset. The network is trained with the stochastic gradient descent (SGD), and two images are applied per minibatch for training and minibatch size is set to 128. Each dataset is trained for 11 epochs with an initial learning rate of 0.001, which is then divided by 10 at epoch 7 and again at epoch 10. The weight decay of 0.0001 and the momentum of 0.9 are used respectively. The dropout method is also adopted to avoid overfitting and the probability of random culling is set to 0.5.

The first experiment is to test the performance of the different numbers of the region proposals. The following are the results for the networks VGG16net on MS COCO for different region proposals, shown in Table 3. The following is the graph of mAP values of the VGG16 net at 1800 proposals on PASCAL VOC 2012 mAP values of each object class Figure 5.

In order to achieve the required performance, it is necessary to appropriately reduce the size of the model and the operation numbers in the forward pass. Therefore, different experiments are conducted, including removing the convolutional layer, replacing two fully connected layers with a single layer, replacing the fully connected layer with a full convolutional layer, and reducing the width of the convolutional layer (i.e., the filter size) and compare their performance, the final results tested on MS COCO dataset are shown in Table 4.

Figure 6 shows the performance of the original method and the method proposed in this paper, and the comparison of the two kinds of pooling methods, Figure 6a shows the effect of the normal method, and Figure 6b shows the effect of the optimized method proposed in this paper. It can be seen that some objects with more overlapping areas (the woman in the right corner) can also be correctly identified. Figure 7 shows the overall comparison, the two images show the effect diagrams of Ren, et al. and the effect diagrams in this paper. It can be noticed that in Figure 7a the plant in the red pot was wrongly detected as two plants, and in Figure 7b this mistake is corrected. Figure 8 shows some test result on our self-built dataset. Objects such as cups, bowls, bottles can be correctly detected in our experiments. There are about 400 pictures in our self-built dataset, and the accuracy rate is about 95% in our experiments. Light and resolution of the pictures will affect the accuracy of detection. Figure 9 is the example case of the simulated recognition in the simulated system, the left image shows that the humanoid manipulator detects a cube, and the right image shows the humanoid manipulator with two hands moves precisely to the cube and can grasp it.

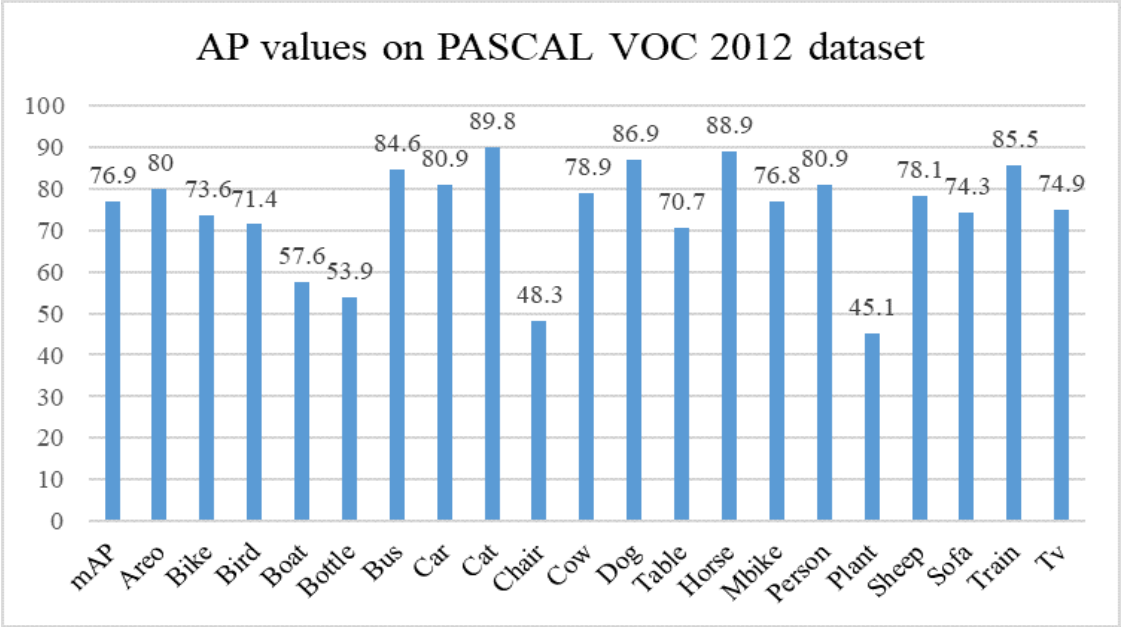

Figure 5: AP values for VGGnet on PASCAL VOC 2012 dataset.

Table 4: Various performance on MS COCO dataset.

Faster R-CNN	proposals	net	AP@ 0.5	AP	mAP	Test time (sec/img)
Baseline from He, et al.	RPN, C4	2fc	47.3	26.3	21.9	3.36
Baseline with conv5	RPN, C5	2fc	51.7	30.6	30.9	3.31
Baseline with conv4 and conv5	RPN, C4, C5	2fc	53.6	32.3	31.6	3.02
Baseline with conv4 and con5	RPN, C4, C5	2 fully conv	53.8	32.1	31.8	2.83

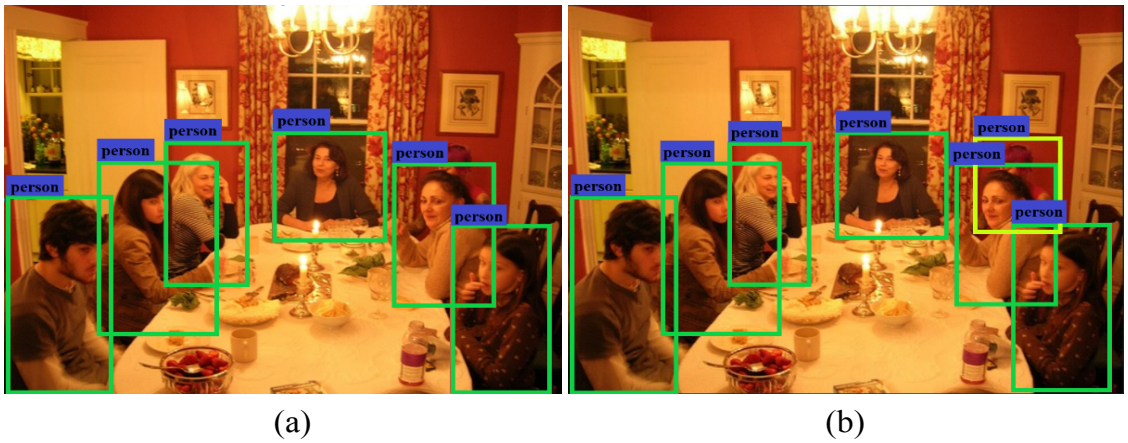

Figure 6: Comparison of the two kinds of pooling methods: a) The effect of the normal method; b) The effect of the optimized method proposed.

Conclusions

In this paper, the region-based Faster R-CNN is used for object detection, and a system that combines Faster R-CNN with the humanoid robot simulation platform for object recognition is proposed. Certain performance improvement is obtained by optimizing the original convolutional network, and further experimental research is carried out through the simulation platform, and obtained relatively good performance. In the meantime, further research and experiments are needed to improve the reliability and practicability of the entire simulation platform in order to make our research practical. We also need to do further research on the motion trajectory planning of a humanoid robot to improve its flexibility.

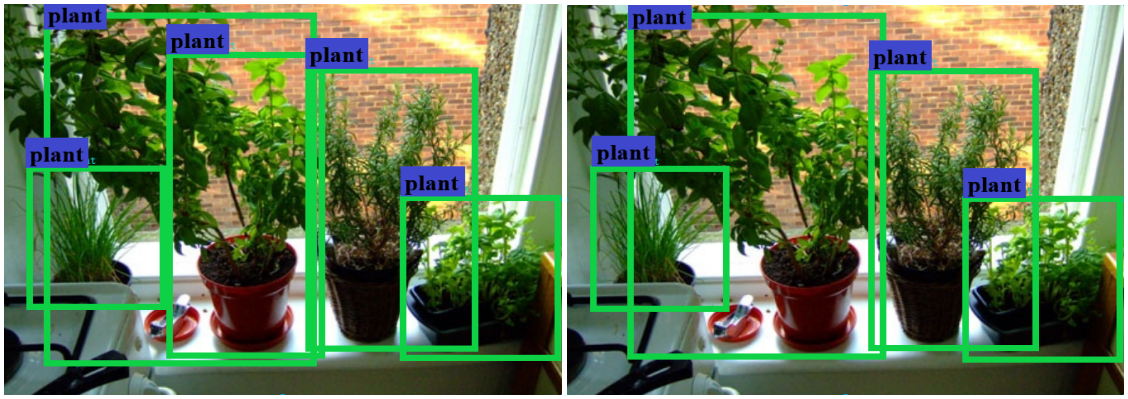

Figure 7: Overall comparison: a) The effect diagrams of Ren, et al. [9]; b) The effect of the optimized method proposed.

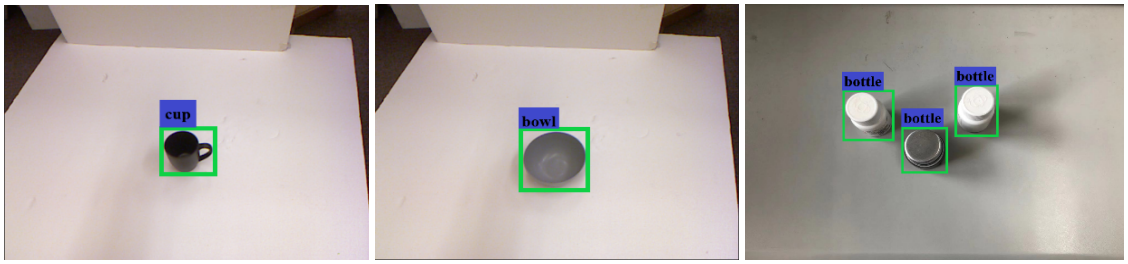

Figure 8: Some test results on the test set of our self-built dataset.

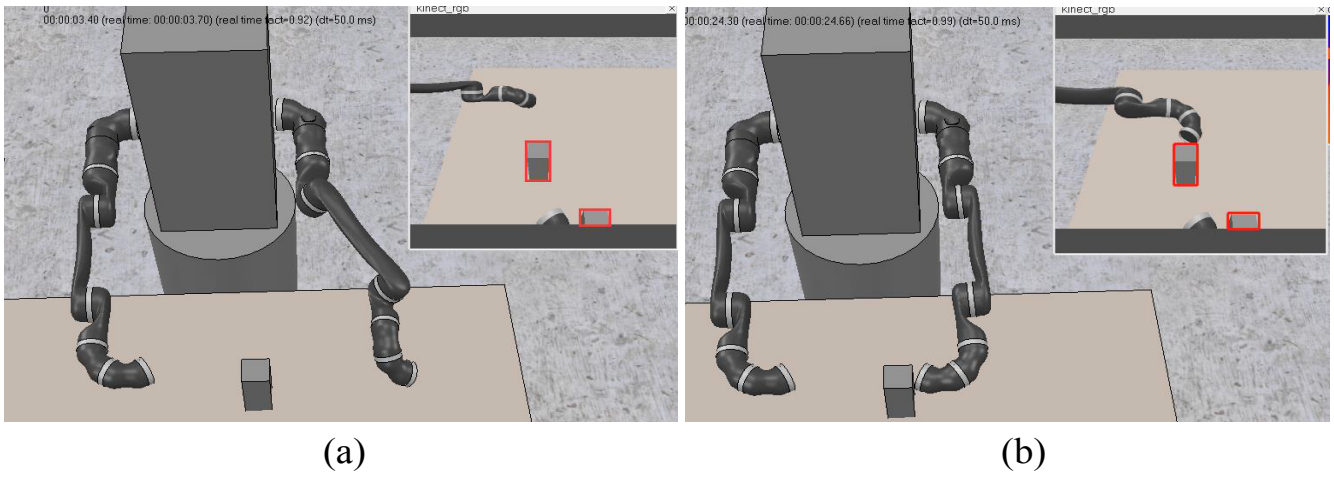

Figure 9: Example case of the simulated recognition in the simulated system: a) The humanoid manipulator detects a cube; b) The humanoid manipulator with two hands moves precisely to the cube and can grasp.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61573145), the Public Research and Capacity Building of Guangdong Province (Grant No. 2014B010104001) and the Basic and Applied Basic Research of Guangdong Province (Grant No. 2015A030308018).

References

1. Pierson HA, Gashler MS (2017) Deep learning in robotics: A review of recent research. Advanced Robotics 31: 821-835.
2. Jiang Q, Tan D, Li Y, et al. (2020) Object detection and classification of metal polishing shaft surface defects based on convolutional neural network deep learning. Applied Sciences 10: 87.
3. Ren Y, Zhu C, Xiao S (2018) Small object detection in optical remote sensing images via modified faster R-CNN. Applied Sciences 8: 813.
4. Dong C, Liu J, Xu F (2018) Ship detection in optical remote sensing images based on saliency and a rotation-invariant descriptor.

Remote Sensing 10: 400.

5. Liu L, Ouyang W, Wang X, et al. (2020) Deep learning for generic object detection: A survey. International Journal of Computer Vision 128: 261-318.
6. Cao C, Wang B, Zhang W, et al. (2019) An improved faster R-CNN for small object detection. IEEE Access 7: 106838-106846.
7. Zhan C, Duan X, Xu S, et al. (2007) An improved moving object detection algorithm based on frame difference and edge detection. Fourth International Conference on Image and Graphics (ICIG 2007), 519-523.
8. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05).
9. Ren S, He K, Girshick R, et al. (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 91-99.
10. Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: A comprehensive review. Neural comput 29: 2352-2449.
11. Levine S, Pastor P, Krizhevsky A, et al. (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research 37: 421-436.
12. Liu W, Anguelov D, Erhan D, et al. (2016) Ssd: Single shot multibox detector. European Conference on Computer Vision, 21-37.
13. Ju M, Luo J, Zhang P, et al. (2019) A simple and efficient network for small target detection. IEEE Access 7: 85771-85781.
14. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 1097-1105.
15. He K, Zhang X, Ren S, et al. (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 37: 1904-1916.
16. Girshick R (2015) Fast r-cnn. 2015 IEEE International Conference on Computer Vision, 1440-1448.
17. Redmon J, Divvala S, Girshick R, et al. (2016) You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition, 779-788.
18. Redmon J, Farhadi A (2017) YOLO9000: Better, faster, stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition, 7263-7271.
19. Redmon J, Farhadi A (2018) Yolov3: An incremental improvement.
20. Sun X, Wu P, Hoi SC (2018) Face detection using deep learning: An improved faster RCNN approach. Neurocomputing 299: 42-50.
21. Saqib M, Khan SD, Sharma N, et al. (2019) Crowd counting in low-resolution crowded scenes using region-based deep convolutional neural networks. IEEE Access 7: 35317-35329.
22. Tzelepi M, Tefas A (2017) Human crowd detection for drone flight safety using convolutional neural networks. 2017 25th European Signal Processing Conference (EUSIPCO), 743-747.
23. Wu L, Li H, He J, et al. (2019) Traffic sign detection method based on Faster R-CNN. Journal of Physics: Conference Series. IOP Publishing 1176: 032045.
24. Xi X, Yu Z, Zhan Z, et al. (2019) Multi-task cost-sensitive-convolutional neural network for car detection. IEEE Access 7: 98061-98068.
25. Ruiz-del Solar J, Loncomilla P, Soto N (2018) A survey on deep learning methods for robot vision.
26. Lenz I, Lee H, Saxena A (2015) Deep learning for detecting robotic grasps. The International Journal of Robotics Research 34: 705-724.
27. Pinto L, Gupta A (2016) Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. 2016 IEEE International Conference on Robotics and Automation (ICRA), 3406-3413.
28. Noda K, Arie H, Suga Y, et al. (2014) Multimodal integration learning of robot behavior using deep neural networks. Robotics and Autonomous Systems 62: 721-736.
29. Yu J, Weng K, Liang G, et al. (2013) A vision-based robotic grasping system using deep learning for 3D object recognition and pose estimation. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), 1175-1180.
30. Zhang F, Leitner J, Milford M, et al. (2015) Towards vision-based deep reinforcement learning for robotic motion control.
31. Gu S, Holly E, Lillicrap T, et al. (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. 2017 IEEE International Conference on Robotics and Automation (ICRA), 3389-3396.
32. Lokanath M, Sai KK, Sanath KE (2017) Accurate object classification and detection by faster-RCNN. IOP Conference Series: Materials Science and Engineering 263: 052028.
33. Zhang D, Li J, Xiong L, et al. (2019) Cycle-consistent domain adaptive faster RCNN. IEEE Access 7: 123903-123911.
34. Zhang Z, Zhou X, Chan S, et al. (2017) Faster R-CNN for small traffic sign detection. CCF Chinese Conference on Computer Vision, 155-165.
35. Roh MC, Lee JY (2017) Refining faster-RCNN for accurate object detection. 2017 Fifteenth IAPR International Conference on Machine Vision Applications, 514-517.
36. Ji H, Gao Z, Mei T, et al. (2019) Improved faster R-CNN with multiscale feature fusion and homography augmentation for vehicle detection in remote sensing images. IEEE Geoscience and Remote Sensing Letters 16: 1761-1765.
37. Rohmer E, Singh SP, Freese M (2013) V-REP: A versatile and scalable robot simulation framework. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1321-1326.
38. Xie M, Zhou D, Shi Y, et al. (2018) Virtual experiments design for robotics based on V-REP. IOP Conference Series: Materials Science and Engineering 428: 012069.
39. Everingham M, Van Gool L, Williams CK, et al. (2010) The PASCAL visual object classes (voc) challenge. Int J Comput Vis 88: 303-338.
40. Lin TY, Maire M, Belongie S, et al. (2014) Microsoft coco: Common objects in context. European Conference on Computer Vision, 740-755.

DOI: 10.36959/673/363