


Article

# Designing a Supermarket Service Robot Based on Deep Convolutional Neural Networks

Aihua Chen , Benquan Yang, Yueli Cui, Yuefen Chen, Shiqing Zhang \* and Xiaoming Zhao

School of Electronics and Information Engineering, Taizhou University, Taizhou 318017, China; chen\_aihua1980@163.com (A.C.); yangbq266@sina.com (B.Y.); cuiyueli@tzc.edu.cn (Y.C.); chen-yuefen@163.com (Y.C.); tzxyzxm@163.com (X.Z.)

\* Correspondence: tzcqsq@163.com

Received: 9 January 2020; Accepted: 15 February 2020; Published: 2 March 2020



**Abstract:** In order to save people's shopping time and reduce labor cost of supermarket operations, this paper proposes to design a supermarket service robot based on deep convolutional neural networks (DCNNs). Firstly, according to the shopping environment and needs of supermarket, the hardware and software structure of supermarket service robot is designed. The robot uses a robot operating system (ROS) middleware on Raspberry PI as a control kernel to implement wireless communication with customers and staff. So as to move flexibly, the omnidirectional wheels symmetrically installed under the robot chassis are adopted for tracking. The robot uses an infrared detection module to detect whether there are commodities in the warehouse or shelves or not, thereby grasping and placing commodities accurately. Secondly, the recently-developed single shot multibox detector (SSD), as a typical DCNN model, is employed to detect and identify objects. Finally, in order to verify robot performance, a supermarket environment is designed to simulate real-world scenario for experiments. Experimental results show that the designed supermarket service robot can automatically complete the procurement and replenishment of commodities well and present promising performance on commodity detection and recognition tasks.

**Keywords:** supermarket service robot; single shot multibox detector; deep convolutional neural networks; robot operating system; Raspberry PI

## 1. Introduction

In recent years, with the rapid development of computer, automation and artificial intelligence, robot technology has drawn tremendous attractions, since robots can be widely used for human-robot collaboration [1], robotic laser welding [2], robotic precision farming [3], and medical treatment [4]. As an important carrier of artificial intelligence, how to design intelligent robots has become an active research subject. With the continuous improvement of people's living standards, more and more service robots have been employed in people's lives [5]. For instance, in large shopping malls, service robots are adopted as an assistant for shopping, thereby improving the customers' service efficiency [6,7].

The current supermarket service robots mainly use cameras to capture images of commodities in the supermarket. Commodities are detected and identified by image processing technology. Therefore, the detection and recognition of commodities has become a key part in a supermarket service robot [8,9]. Traditional methods of object detection and recognition are achieved by manually extracting hand-crafted features of objects, such as scale invariant feature transform (SIFT) [10,11], histograms of oriented gradients (HOG) [12], etc. However, these hand-crafted features are low-level, so that they are not discriminative enough for object detection and recognition. Due to diversity of

objects and complexity of background in real-world sceneries, it is more difficult to obtain promising performance on object detection and recognition tasks for the supermarket service robots.

To address the above-mentioned issue, the recently-emerged deep learning techniques may present an alternative approach. Among them, deep convolutional neural networks (DCNNs) [13] are one of typical deep learning methods, which aims to learn high-level discriminative features with multiple convolutional and pooling layers. Owing to the powerful feature learning ability, DCNNs have exhibited its excellent performance for object detection and recognition [14,15]. At present, the representative DCNNs models for object detection and recognition include YOLO [16], Faster RCNN (region-based CNN) [17], single shot multibox detector (SSD) [18], and so on.

YOLO [16] aims to utilize the entire image as an input of the network. The convolutional layer at the beginning of the model extracts image features, and returns directly to the position of the bounding box and the category of the bounding box. Its network structure is similar to Google Net [19]. YOLO is simple and fast, but the object detection accuracy is relatively low when dealing with dense and small objects. To overcome this drawback, several improved versions of YOLO are proposed in recent years, such as YOLO 9000 [20] and YOLO v3 [21].

Faster RCNN [17] is an another typical algorithm based on region proposal for object detection and recognition. This method is developed on the base of RCNN [22,23] and Fast RCNN [24]. It first uses a series of convolutional layers to extract feature maps. Then, each object proposal is mapped to a region of interest from feature maps. The region of interest (ROI) pooling layer adopts max-pooling to convert the features of interest with different size into a fixed spatial extent of  $7 \times 7$ . The feature vector is fed into a sequence of fully connected layer that finally branch into two sibling output layers such as SoftMax and bounding box regression. Faster RCNN has high performance on object detection and recognition tasks, but it runs too slowly due to its large computation complexity. Recently, the R-FCN [25] further optimizes network structure of Faster RCNN, in which the VGG or ZF networks in Faster RCNN are replaced by using a more powerful ResNet network [26].

SSD [18] is proposed on the basis of the CNN and YOLO network. Similar to YOLO, SSD completes both the generation of object candidate area and the recognition of object candidate area. The object detection time for SSD is thus greatly reduced. In order to overcome the problem of inaccurate positioning and difficulty in detecting small objects, SSD makes improvements in two aspects. Firstly, SSD extracts feature maps of different scales for detection. Large-scale feature maps are used to detect small objects, whereas small-scale feature maps are employed to detect large objects. Secondly, SSD leverages a priori frames of different scales and aspect ratios. Compared with YOLO and Faster RCNN, the detection accuracy of SSD is higher than YOLO, but slightly lower than Faster RCNN. Nevertheless, the detection speed of SSD is the fastest. Therefore, this work adopts SSD for object detection and recognition to design a supermarket service robot.

The main contributions of this work can be summarized as follows:

- (1) We develop a supermarket service robot on the basis of DCNNs, in which hardware and software systems are designed. In order to verify the working stability of the robot, a supermarket simulation environment is built, and two working modes of automatic commodity procurement and replenishment are verified.
- (2) We develop a small-scale image dataset containing 12 supermarket commodities, and compare three different methods for commodities detection and recognition with the designed supermarket service robot. Experiment results demonstrate the effectiveness of the proposed method.

The structure of this paper is as follows. Section 2 introduces the hardware and software system of a supermarket service robot. Section 3 presents the simulated experimental environment and an analysis of experimental results. Finally, the discussion and conclusions are given in Sections 4 and 5, respectively.

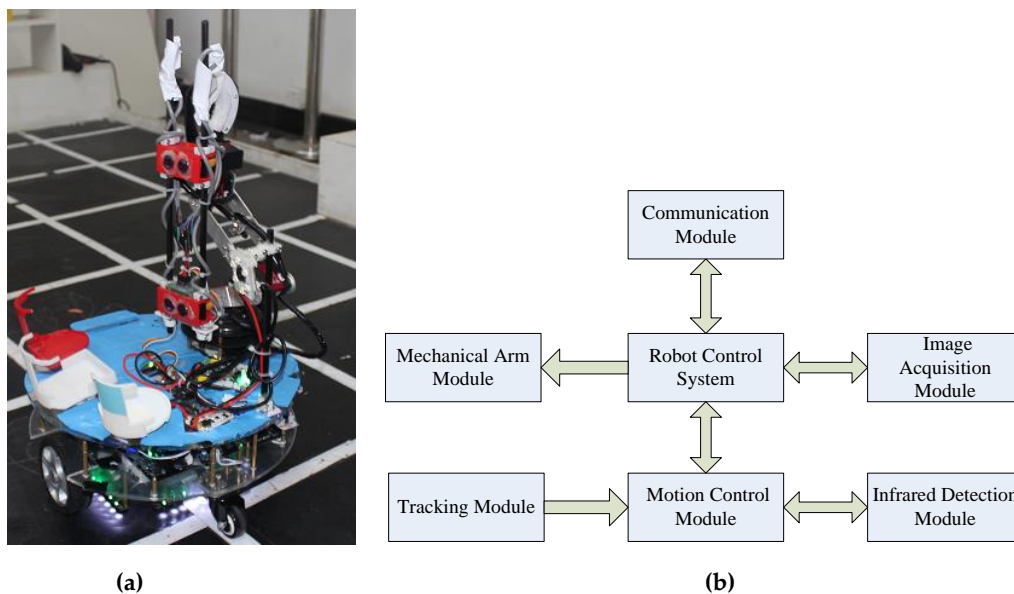
## 2. Our Method

Our designed supermarket service robot comprises of two key parts: A robot hardware system and a robot software system, as described below.

### 2.1. Robot Hardware System

#### 2.1.1. The Hardware Framework

Figure 1 shows the appearance as well as the robot hardware system of designed supermarket service robot. It can be seen from Figure 1b, the whole hardware system mainly includes seven parts: Robot control module, motion control module, tracking module, infrared detection module, communication module, image acquisition module, and implementation module.



**Figure 1.** The hardware framework: (a) The appearance of the designed supermarket service robot, (b) The hardware framework of the designed supermarket service robot.

The robot control module is the popular Raspberry PI which is used as control kernel. It is an open source hardware platform. The bottom layer is a complete Linux operating system in which all the basic functions of a computer are included. The robot control module receives signals sent by other parts, and performs a processing analysis. It also sends control instructions to coordinate various parts of the robot to complete the tasks specified by the users.

The motion control module includes motor drive module, four stepper motors and four omnidirectional wheels [27]. The motor drive module is implemented by a STM32. The stepper motor and omnidirectional wheels are symmetrically installed under the robot chassis. Each omnidirectional wheel is connected to a stepper motor independently, which ensures the stable and flexible operation of robot.

In order to improve the stability of robot and reduce cost, this work uses infrared detection to control the robot movement. The infrared ray has different reflection properties on the surface of objects with different colors. During robot movement, the infrared light is continuously emitted to ground. When the infrared light meets white floor, it is diffusely reflected. The reflected light is received by the receiver installed on robot. If the black part is encountered, infrared light is absorbed and the receiving tube cannot receive infrared light. Therefore, the control system determines robot's walking route based on whether infrared light is received or not.

The infrared detection module works similarly to tracking module. The infrared sensor emits a beam of infrared light. The infrared light strikes surface of an object to form reflected light. The sensor

receives reflected signals. The robot can calculate distance from object to robot based on the time difference between emitted light and reflected light. This module is used to detect whether there are commodities on the shelf or not.

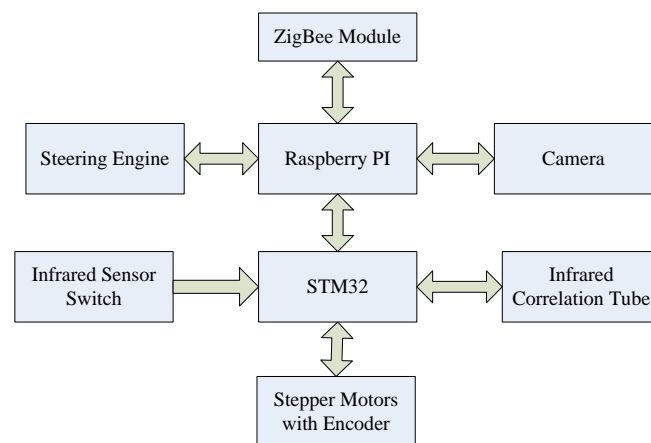
The communication module is used to communicate between service robots and users in which the ZigBee wireless transmission technology is leveraged. It receives work instructions sent by users to service robot. After robot completes working, it sends instructions to users.

The image acquisition module is utilized to collect image information of commodities and upload them to Raspberry PI.

The implementation module contains a mechanical arm. It has six degrees of freedom, and can flexibly complete grasping and placing of commodities.

### 2.1.2. Robot Hardware Components

The detailed hardware components of supermarket service robot are shown in the Figure 2. The robot hardware is mainly composed of the Raspberry PI and STM32, which communicate with each other through serial ports.



**Figure 2.** Robot hardware components.

The Raspberry PI has three functions: (1) It uses the ZigBee module to realize the communication between robots and customers; (2) it controls a camera to collect commodity images and identify them; (3) it controls the manipulator movements through steering gears.

The STM32 also has three functions: (1) It controls the robot movements which are derived by a motor with an encoder; (2) it controls eight infrared correlation tubes to achieve the robot tracking; (3) it controls infrared sensor switch to achieve blank shelves detection. Note that the power supply is powered by a lithium battery placed on the robot chassis. In order to ensure the stability of robot movement, the components of the robot are symmetrically assembled together.

### 2.2. Robot Software System

The software system in designed supermarket service robot runs on the Ubuntu Mate operating system, in which the ROS framework is used to develop robot control program. The ROS framework is designed for robot to carry out a variety of commands, including moving, grabbing, and positioning. The entire control and processing processes are implemented by a Python programming language.

For commodity inspection, the work is mainly divided into two parts. Firstly, robot collects commodity images to builds a small-scale image dataset. In the supermarket service robot system, the pre-trained deep model is obtained in advance by the GPU training. Then, the Raspberry PI controls the cameras to collect commodity images, and uses the pre-trained model to perform a real-time commodity detection and recognition task. Note that when the robot works, the detection and recognition of commodities are carried out on a CPU in the Raspberry PI since the Raspberry PI

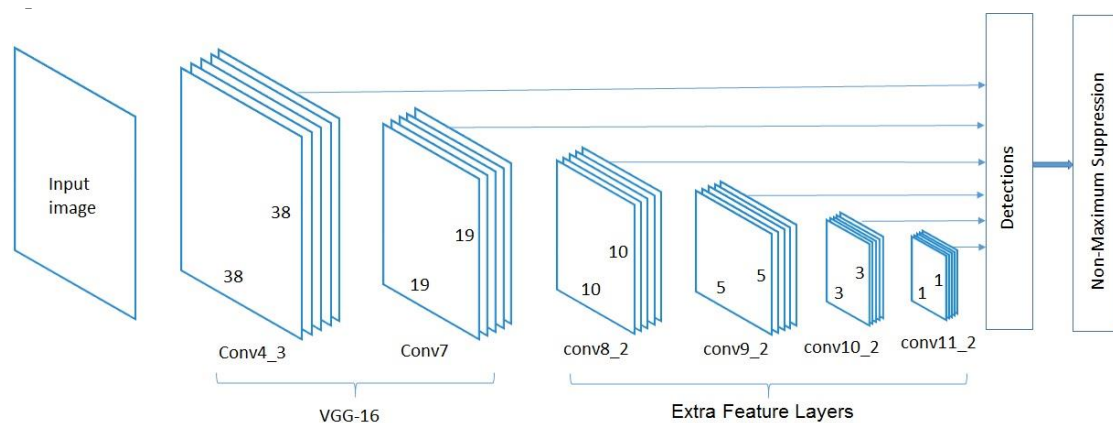
does not have an integrated GPU. In the followings, we will describe the principle of SSD and the ROS framework in detail.

### 2.2.1. The Principle of SSD

In the robot software system, we employ SSD to implement object detection and recognition. SSD [18] is an end-to-end network model based on the feed-forward convolutional neural network. All detection and recognition processes are integrated in the same network structure. As the number of network layers continues to increase, the feature maps become smaller and smaller. The bottom feature maps contain more detailed information which is used to detect small-sized objects. The high-level feature maps contain richer semantic information which is good at large-size object detection. SSD employs comprehensive multi-layer feature maps to give detection results and enlarges the scale range of the detected objects. At the back of the feature maps, SSD adopts the convolution sum of  $3 \times 3$  and  $1 \times 1$  to output the position, category and confidence of the objects. With the help of weight-sharing features of the convolution kernel, parameters are reduced.

In order to better detect objects of various shapes, SSD adopts the anchor mechanism of Faster R-CNN, and sets default boxes with different sizes and different aspect ratios as the center of each unit of the feature maps. The SSD network output is composed of two parts: the location of the detection box and the confidence of category. SSD filters redundant candidate box through non-maximum suppression, and then produces final detection results.

The detailed network structure of SSD is presented in Figure 3. The SSD input is the entire image. Feature maps of the input image are extracted through a convolutional neural network based on VGG-16, and then CNN layers of different sizes are added to network for extracting multiple scale features. Due to the strong similarity between adjacent pixels in images, SSD performs convolution operations at intervals of several pixels, which removes redundancy while expanding the effective receptive field.



**Figure 3.** Detailed network structure of single shot multibox detector (SSD).

In order to detect objects of different sizes, SSD selects conv4\_3, conv7, conv8\_2, conv9\_2, conv10\_2, and conv11\_2 feature maps for classification and border regression. Different feature maps have different sizes and different effective receptive fields, so the corresponding default box sizes are also different. SSD redesigns the generation scheme of the default boxes. The default boxes of different scale parameters are designed for the feature maps of different scales. Assuming there are  $N$  feature maps, the scale parameters of the default box of the corresponding feature maps are as follows:

$$s_i = s_{\min} + \left( \frac{s_{\max} - s_{\min}}{N-1} \right) \cdot (i-1) \quad i \in [1, N] \quad (1)$$

where  $s_{\max}$  and  $s_{\min}$  are the maximum and minimum scale parameters, respectively. SSD designs different default boxes based on the object size and aspect ratio. The default box usually has 5 values

for the aspect ratio, i.e., 1, 2, 3, 1/2, and 1/3. There are two default boxes with a ratio of 1, with sizes  $s_i$  and  $\sqrt{s_i s_{i+1}}$ .

In order to establish the relationship between ground truth boxes and default boxes, SSD selects the default boxes with threshold of ground truth boxes. The default boxes are higher than 0.5. In this way, SSD obtains multiple candidate boxes that overlap each other and score higher. After matching, most of the default boxes will become negative samples which are used to stabilize the training and fast convergence.

According to the matching strategy of SSD, define  $x_{ij} = 1$  to indicate that the  $i$ -th default box matches the  $j$ -th real box of the category  $p$ . There may be multiple default boxes with thresholds greater than 0.5. The overall loss is a weighted sum of the localization loss ( $loc.$ ) and the confidence loss ( $conf.$ ), as defined below:

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \quad (2)$$

where  $N$  is the number of matched default boxes.  $\alpha$  is used to adjust the scale, which is default to 1.  $c$  is the predicted value of class confidence.  $l$  is the position parameter of the predicted box, and  $g$  is the position parameter of ground truth box.

$L_{loc}(x, l, g)$  is the smooth L1 loss which is defined as:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & |x| \geq 1 \end{cases} \quad (3)$$

where  $(cx, cy)$ ,  $w$  and  $h$  are the center, width and height of the default bounding box ( $d$ ), respectively.

The calculation expression of the confidence function is as follows:

$$L_{conf}(x, c) = - \sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad (4)$$

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (5)$$

### 2.2.2. ROS

The ROS framework integrates robot components and provides a communication architecture and interface [28,29]. It presents hardware description, low-level driver management, communication, program distribution package management, and other operating system-like functions. In addition, it provides some tools and libraries which are used to obtain, build, write, and run multi-machine integrated programs. In this operating environment, the robot perception, decision-making, and control algorithms can be better organized.

ROS uses a distributed processing framework to design various functions of robot. The point-to-point design allows the robot's process to run separately. As a result, it facilitates modular modification and customization, and improves the fault tolerance of the system. ROS supports the development of multiple programming languages. No matter which programming language is used, information interaction between nodes can be connected through a language-independent interface.

It also complies with the Berkeley Software Distribution (BSD) protocol which is completely free for personal and commercial applications and modifications. These advantages make the ROS system gradually become a widely-used platform in the field of robotics research.

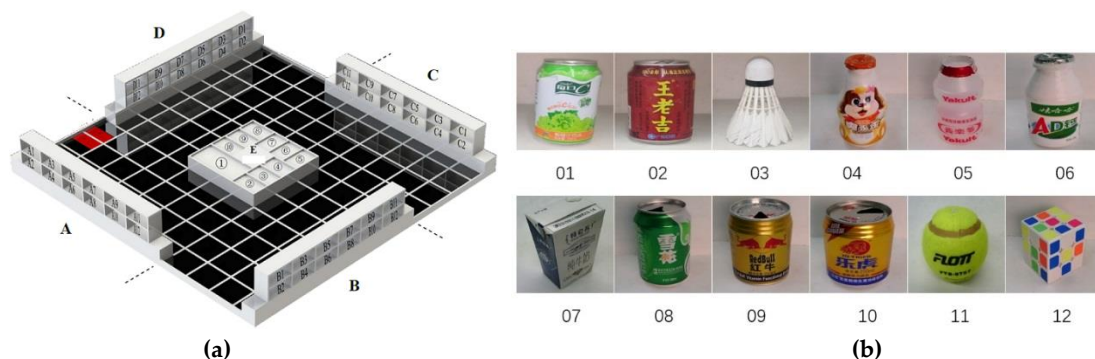
The ROS system includes three parts: File system level, computational graph level, and community level. The file system level is to organize files according to their functions and gather files with certain functions together. This is used to facilitate the software reuse. The computational graph level is a point-to-point network form for processing data. It describes the communication between nodes and systems. The community level is a form of the ROS code distribution, which is used for sharing of knowledge.

### 3. Experiments

#### 3.1. Experimental Setup

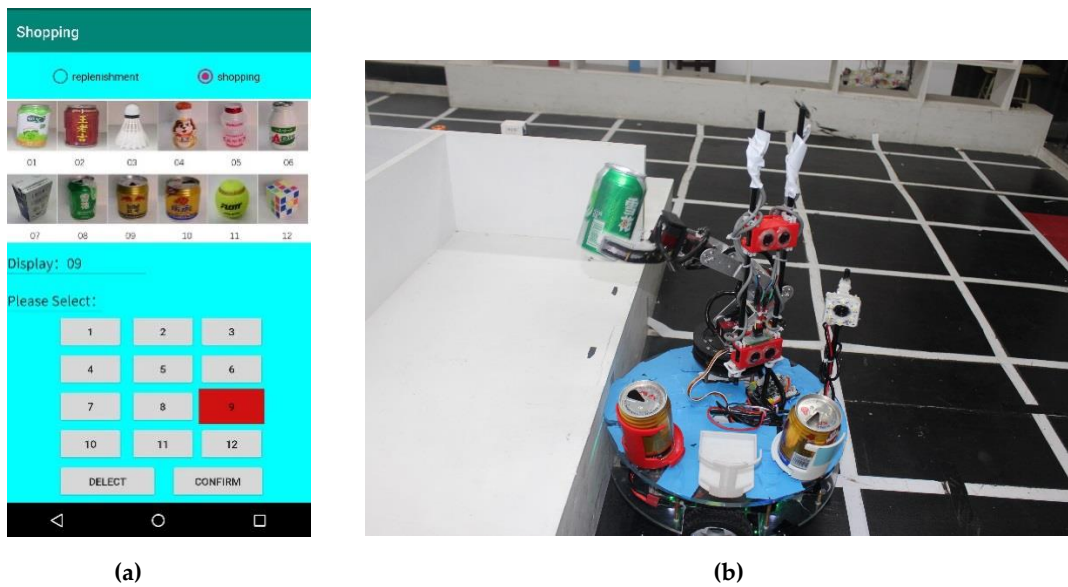
In order to verify effectiveness of the designed supermarket service robot, we present two testing methods of the supermarket service robot: the commodity procurement and replenishment. The commodity procurement aims to capture commodities from shelves that customers want to buy and put them in the specified position in the shopping cart. The commodity replenishment means that the supermarket robot places commodities in warehouse to the blank window of shelf.

The supermarket shopping environment we simulated is given in Figure 4a. In Figure 4a, surrounding areas, such as A, B, C, and D, are shopping shelves, each of which has a double-deck shelf. The middle E area is used as a shopping cart or warehouse. The conventional 12 commodity items labeled from 01 to 12 are shown in Figure 4b. Note that the labels of commodities from 01 to 12 in Figure 4b are only used to make the numbers more readable. But in the Python execution experiments, the real labels for training deep models are annotated from 01 and 12.



**Figure 4.** The experiment environment and commodities: (a) The simulated supermarket shopping environment, (b) 12 commodities on the shelf or warehouse.

In addition, we have developed a software interface of supermarket robot facing with customers, as depicted in Figure 5a. When purchasing commodities, the customer is demanded to click the serial number corresponding to the commodities in turn, and confirm this command. The mobile phone then sends this command to the supermarket service robot through wireless network. The supermarket service robot starts from the area A, and sequentially detects each window of the shelf. Meanwhile, it identifies the purchased products, grabs and places them in the shopping cart. When commodities are replenished, the supermarket staffs click the corresponding number of commodities in turn, and confirm this command. Figure 5b shows a working picture of grabbing commodities with the designed supermarket service robot.



**Figure 5.** The software interface and a working show: (a) Commodities selection interface, (b) a working picture of grabbing commodities with the designed supermarket service robot.

### 3.2. Experimental Results and Analysis

In the experiments, the recognition accuracy of commodities is a key step for the designed supermarket service robot. This work uses SSD algorithm to detect and identify commodities. For this purpose, we construct a small-scale dataset including conventional supermarket commodity images for experiments. This dataset contains 12 types of commodities. Each of the commodities from 01 to 11 has about 6000 images. Note that the labeled number 12 is a Rubik's cube. We changed the color order of the Rubik's cube, and collected a total of 10,000 images. We employ 60% of the collected data as a training set, and the remaining 40% as a testing set.

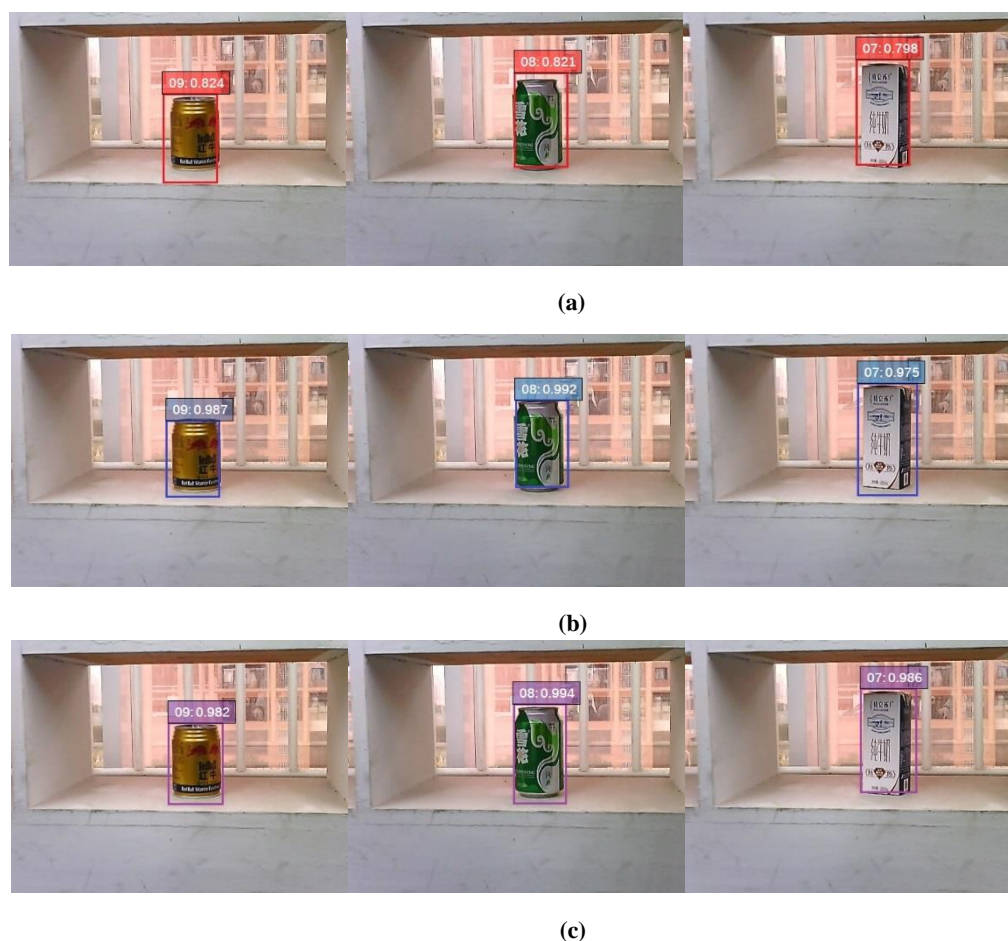
#### 3.2.1. Performance Analysis of SSD

In order to evaluate the performance of the used SSD algorithm, we compare three deep CNN models such as YOLO, Faster RCNN and SSD on commodity detection and recognition tasks. Figure 6 presents an example of the used three deep CNN models. Table 1 shows the detailed mAP (mean average precision) and FPS (frames per second) of the used three deep CNN models when IOU (intersection over union) is set to 0.5 for testing. The results in Figure 6 and Table 1 indicate that SSD has the fastest speed with 17 FPS, followed by YOLO, and Faster RCNN. Nevertheless, SSD obtains the second mAP of 77.6%, which is a little lower than the first mAP of 78.1% achieved by Faster RCNN. Considering the tradeoff between mAP and FPS, this work thus adopts SSD to implement commodity detection and recognition tasks for our designed supermarket service robot.

**Table 1.** Comparison of different deep models.

Models	mAP	FPS
YOLO	67.3	16
Faster RCNN	78.1	3
SSD	77.6	17





**Figure 6.** An example of commodity detection and recognition: (a) the obtained results of YOLO, (b) the obtained results of Faster RCNN (region-based CNN), (c) the obtained results of SSD.

### 3.2.2. Comparisons between Hand-Crafted Features and SSD

In order to verify performances of the used SSD algorithm, we compare SSD with two hand-crafted features including SIFT and HOG. We adopt SIFT and HOG features for commodity detection, followed by the linear support vector machine (SVM) for commodity classification. Table 2 lists the recognition results of different commodity detection and recognition methods. From Table 2, we can see that SSD performs better than SIFT and HOG. In particular, SSD obtains the best accuracy of 94.98%, whereas SIFT and HOG yield an accuracy of 86.72% and 85.90%, respectively. The results in Table 2 clearly indicate the advantages of learned features obtained by SSD over hand-crafted features on object detection and recognition tasks.

**Table 2.** Performance comparison (%) of different commodity detection and recognition methods.

Method	01	02	03	04	05	06	07	08	09	10	11	12	Avg.
SIFT	84.14	85.83	91.37	87.48	85.75	88.26	89.54	85.27	80.53	81.41	93.86	87.27	86.72
HOG	83.65	85.86	90.76	86.85	84.36	86.63	86.78	84.64	82.36	80.87	92.17	85.86	85.90
SSD	94.88	96.35	96.75	96.14	94.87	95.52	95.81	95.62	90.14	90.63	98.31	94.69	94.98

In order to further analyze the recognition accuracy of each commodity, Figure 7 shows the confusion matrix of commodity recognition results obtained by SSD. Note that in Figure 7 the diagonal bold digits represent the correct recognition rates for the corresponding labeled commodity, whereas the other digits in parallel denote the wrong recognition rates in which this labeled commodity is wrongly identified as other commodities. For instance, Figure 7 shows that the commodity 01 is correctly

classified with accuracy of 94.88%, and identified as the commodity 03 with a wrong recognition rate of 0.08%, as well as the commodity 04 with a wrong recognition rate of 0.13%, respectively. It can be seen from Figure 7 that the labeled 11 commodities are classified well with the highest accuracy of 98.31%. Note that the commodities 09 and 10 are too similar in appearance so that they are easily confused to each other. Therefore, these two commodities are more difficult to detect and recognize than others. This results in the obtained low recognition performance for these two commodities.

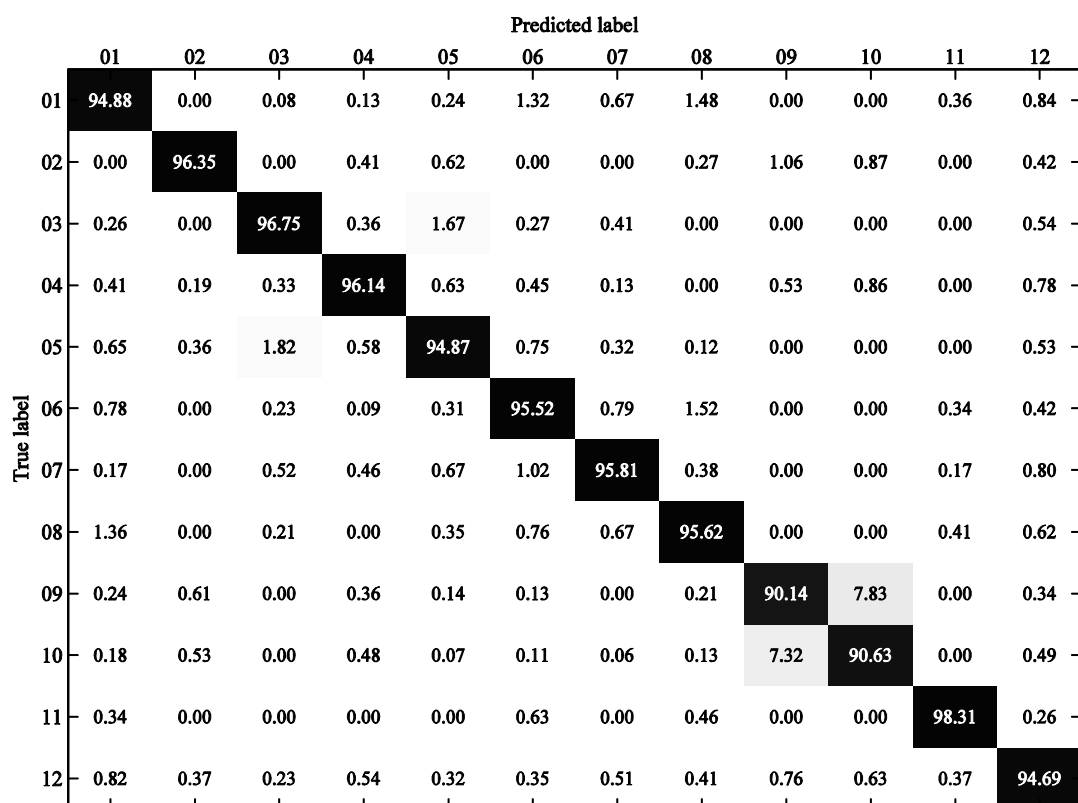


Figure 7. The confusion matrix of recognition accuracy (%) obtained by SSD.

#### 4. Discussion

In large warehouse/industrial inventory management systems, autonomous/programmable robots may adopt the similar idea for working. However, they usually work in a relatively single way. By contrast, supermarket service robots face larger challenges, such as complex supermarket environment, a large number of products, diverse needs of customers and staff. Our work is mainly motivated by the great success of recently-developed deep learning techniques. Experiment results demonstrate the validity of our designed supermarket service robot.

At present, our designed robot is in an initial stage for testing. We choose 12 supermarket commodities to construct the supermarket shopping environment for experiment simulation. Note that all the chosen 12 commodities are almost canned so that they are very similar in appearance. This will result in the difficulty of object detection and recognition. Once our work obtains promising performance for these similar 12 commodities, it is easy and scalable for our method to deal with a large supermarket with several hundreds of different commodities.

On object detection and recognition tasks, we compare our used SSD method with two typical deep learning methods such as YOLO, and Faster RCNN. Experiment results show the effectiveness of the used SSD method. In future, it is interesting to investigate the performance of other advanced deep learning methods such as R-FCN, YOLO 9000, and YOLO v3 on our designed robot.

However, the robot is in an initial stage of testing. The operation mode and control of the robot are relatively simple. In the actual supermarket environment, multiple robots need to work together,

which requires a server to connect robots to form a local area network. Therefore, the communication protocol [30,31], the security and anti-interference ability of the network [32,33] should be considered. In future, it is very meaningful to consider the security challenges, such as distributed denial of service attacks. In addition, the current work of this work only involves one robot. In the actual supermarket environment, multiple robots are needed to work together. It is thus a challenging subject to establish a complicated robot local area network with multiple robots to facilitate the resource sharing and robot management.

## 5. Conclusions

This work proposes a supermarket service robot based on SSD. We design the hardware and software structure and working principle of the supermarket service robot. The designed supermarket service robot employs the STM32 and Raspberry PI as the hardware framework. The software framework for commodity detection and recognition is implemented by the SSD algorithm. In order to test the stability and reliability of the designed supermarket robot, we have established a supermarket simulation environment and constructed a small-scale image dataset containing 12 supermarket commodities for experiments. Experimental results show the effectiveness of our designed supermarket service robot.

**Author Contributions:** Proposed method, robot software system design and Writing—original draft preparation: A.C. and S.Z.; Hardware system and structure design of robot: B.Y. and Y.C. (Yueli Cui); Robot simulation test: Y.C. (Yuefen Chen) and A.C.; Writing—Review and Editing: S.Z., A.C., and X.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by Zhejiang Provincial National Science Foundation of China and National Science Foundation of China (NSFC) under Grant No. LZ20F020002 and 61976149, Taizhou Science and Technology Project under Grant No. 1803gy08 and 1802gy06, and Outstanding Youth Project of Taizhou University under Grant No. 2018JQ003 and 2017PY026.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Liu, H.; Fang, T.; Zhou, T.; Wang, Y.; Wang, L. Deep learning-based multimodal control interface for human-robot collaboration. *Procedia CIRP* **2018**, *72*, 3–8. [[CrossRef](#)]
2. Fang, Z.; Weng, W.; Wang, W.; Zhang, C.; Yang, G. A Vision-Based Robotic Laser Welding System for Insulated Mugs with Fuzzy Seam Tracking Control. *Symmetry* **2019**, *11*, 1385. [[CrossRef](#)]
3. Kounalakis, T.; Triantafyllidis, G.A.; Nalpantidis, L. Deep learning-based visual recognition of rumex for robotic precision farming. *Comput. Electron. Agric.* **2019**, *165*, 104973. [[CrossRef](#)]
4. Li, X.; Cao, L.; Tiong, A.M.H.; Phan, P.T.; Phee, S.J. Distal-end force prediction of tendon-sheath mechanisms for flexible endoscopic surgical robots using deep learning. *Mech. Mach. Theory* **2019**, *134*, 323–337. [[CrossRef](#)]
5. Paulius, D.; Sun, Y. A Survey of Knowledge Representation in Service Robotics. *Robot. Auton. Syst.* **2019**, *118*, 13–30. [[CrossRef](#)]
6. Bertacchini, F.; Bilotta, E.; Pantano, P. Shopping with a robotic companion. *Comput. Hum. Behav.* **2017**, *77*, 382–395. [[CrossRef](#)]
7. Cheng, C.-H.; Chen, C.-Y.; Liang, J.-J.; Tsai, T.-N.; Liu, C.-Y.; Li, T.-H.S. Design and implementation of prototype service robot for shopping in a supermarket. In Proceedings of the 2017 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, Taiwan, 6–8 September 2017; pp. 46–51.
8. Pasquale, G.; Ciliberto, C.; Odone, F.; Rosasco, L.; Natale, L. Are we done with object recognition? The iCub robot's perspective. *Robot. Auton. Syst.* **2019**, *112*, 260–281. [[CrossRef](#)]
9. Cartucho, J.; Ventura, R.; Veloso, M. Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2336–2341.
10. Lowe, D.G. Distinctive image features from scale invariant key points. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [[CrossRef](#)]

11. Ke, Y.; Sukthankar, R. PCA-SIFT: A more distinctive representation for local image descriptors. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, 27 June–2 July 2004; Volume 4, pp. 506–513.
12. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
13. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 3–8 December 2012; pp. 1097–1105.
14. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)] [[PubMed](#)]
15. Zhang, S.; Zhang, S.; Huang, T.; Gao, W.; Tian, Q. Learning affective features with a hybrid deep model for audio–visual emotion recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 3030–3043. [[CrossRef](#)]
16. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
17. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
18. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
19. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
20. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
21. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
22. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
23. Cai, Z.; Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6154–6162.
24. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
25. Dai, J.; Li, Y.; He, K.; Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 379–387.
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
27. Li, Y.; Dai, S.; Zhao, L.; Yan, X.; Shi, Y. Topological Design Methods for Mecanum Wheel Configurations of an Omnidirectional Mobile Robot. *Symmetry* **2019**, *11*, 1268. [[CrossRef](#)]
28. Chang, Y.-H.; Chung, P.-L.; Lin, H.-W. Deep learning for object identification in ROS-based mobile robots. In Proceedings of the 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, Japan, 13–17 April 2018; pp. 66–69.
29. Bayar, V.; Akar, B.; Yayan, U.; Yavuz, H.S.; Yazici, A. Fuzzy logic based design of classical behaviors for mobile robots in ROS middleware. In Proceedings of the IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, Alberobello, Italy, 23–25 June 2014; pp. 162–169.
30. Sikeridis, D.; Tsiropoulou, E.E.; Devetsikiotis, M.; Papavassiliou, S. Context-Aware Wireless-Protocol Selection in Heterogeneous Public Safety Networks. *IEEE Trans. Veh. Technol.* **2018**, *68*, 2009–2013. [[CrossRef](#)]

31. Butt, T.A.; Phillips, I.; Guan, L.; Oikonomou, G. Adaptive and context-aware service discovery for the internet of things. In Proceedings of the 13th International Conference, NEW2AN 2013 and 6th Conference, ruSMART 2013, Petersburg, Russia, 28–30 August 2013; pp. 36–47.
32. Tsiropoulou, E.E.; Baras, J.S.; Papavassiliou, S.; Qu, G. On the mitigation of interference imposed by intruders in passive RFID networks. In Proceedings of the International Conference on Decision and Game Theory for Security, New York, NY, USA, 2–4 November 2016; pp. 62–80.
33. Zhang, K.; Liang, X.; Lu, R.; Shen, X. Sybil attacks and their defenses in the internet of things. *IEEE Internet Things J.* **2014**, *1*, 372–383. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).