Article

Vision-Based Robotic Object Grasping—A Deep Reinforcement Learning Approach

Ya-Ling Chen, Yan-Rou Cai and Ming-Yang Cheng *

Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan
* Correspondence: mycheng@mail.ncku.edu.tw

Abstract: This paper focuses on developing a robotic object grasping approach that possesses the ability of self-learning, is suitable for small-volume large variety production, and has a high success rate in object grasping/pick-and-place tasks. The proposed approach consists of a computer vision-based object detection algorithm and a deep reinforcement learning algorithm with self-learning capability. In particular, the You Only Look Once (YOLO) algorithm is employed to detect and classify all objects of interest within the field of view of a camera. Based on the detection/localization and classification results provided by YOLO, the Soft Actor-Critic deep reinforcement learning algorithm is employed to provide a desired grasp pose for the robot manipulator (i.e., learning agent) to perform object grasping. In order to speed up the training process and reduce the cost of training data collection, this paper employs the Sim-to-Real technique so as to reduce the likelihood of damaging the robot manipulator due to improper actions during the training process. The V-REP platform is used to construct a simulation environment for training the deep reinforcement learning neural network. Several experiments have been conducted and experimental results indicate that the 6-DOF industrial manipulator successfully performs object grasping with the proposed approach, even for the case of previously unseen objects.

Keywords: 6-DOF industrial manipulator; deep reinforcement learning; soft actor-critic; robotic object grasping; YOLO

check for updates

Citation: Chen, Y.-L.; Cai, Y.-R.; Cheng, M.-Y. Vision-Based Robotic Object Grasping—A Deep Reinforcement Learning Approach. *Machines* **2023**, *11*, 275. https://doi.org/10.3390/machines11020275

Academic Editor: Shabnam Sadeghi Esfahlani

Received: 5 January 2023
Revised: 8 February 2023
Accepted: 10 February 2023
Published: 12 February 2023

1. Introduction

In most conventional approaches for vision-based pick-and-place tasks of industrial manipulators used in a production line, a 3D model of the object to be grasped must be known in advance. With the known 3D model, one can either analyze the geometric shape of the object and find a proper way for the industrial manipulator to grasp that object or exploit methods such as feature matching and shape recognition to find an appropriate pose for the industrial manipulator to perform object grasping as well as pick-and-place tasks. However, this kind of approach is sensitive to the illumination conditions and other types of disturbances in the ambient environment. If the 3D model of the object to be grasped is not known in advance or if there are a variety of objects to be grasped, the aforementioned conventional approaches may fail. With the machine learning paradigm becoming popular, more and more research has been focused on applying the deep learning technique to deal with automatic object grasping tasks [1]. For example, Johns et al. [2] used a deep neural network to predict a score for the grasp pose of a parallel jaw gripper for each object in a depth image, through which a physical simulator was employed to obtain simulated depth images of objects as the training data set. Lenz et al. [3] used two deep neural networks to detect robotic grasps from images captured by an RGBD camera. One deep neural network having a simpler structure and requiring fewer computation resources was mainly used to retrieve candidate bounding rectangles for grasping. Another deep neural network was used to rank the candidate bounding rectangles for a parallel gripper [3]. In [4], 700 h were spent collecting data from 50,000 grasping attempts of robot manipulators, and a Convolutional Neural Network (CNN) was combined with a multi-stage learning approach to predict an appropriate grasping pose for robot manipulators. In [5], Levine et al. exploited the deep learning paradigm to train fourteen 7-DOF robot manipulators to perform object grasping using RGB images. A total of 800,000 grasp attempts by robot manipulators were recorded within two months to train the deep neural network. Experimental results indicate that the robot manipulator can successfully grasp 1100 objects of different sizes and shapes. Goldberg and his colleagues have done a series of studies on robot grasping using deep learning [6–9]. Mahler et al. proposed the Dex-Net 1.0 deep learning system for robot grasping [6]. More than 10,000 independent 3D models and 2.5 million samples of grasping data for the parallel gripper are used in Dex-Net 1.0. In order to shorten the training time, 1500 virtual cores in the Google cloud platform are used. In 2019, Mahler et al. proposed the Dex-Net 4.0 [9], for which five million depth images had been trained by a GQ-CNN. After the training is complete, the dual arm robot with a suction nozzle and a parallel-jaw gripper is able to empty a bin with an average grasping rate of 300 objects/hour [9]. Several past studies utilized CNN to produce suitable grasping poses to perform object grasping tasks [10–12]. All of the aforementioned studies demonstrated good performance in automatic object grasping, even for cases in which the objects to be grasped did not appear in the training data set. However, the subjects of these past studies all have common drawbacks, in that they are very time consuming and not cost effective in generating a grasping data set for training the deep neural network.

Recently, the research topic of exploiting reinforcement learning in training robot manipulators to perform object grasping has received much attention [13–15]. Gualtieri et al. used deep reinforcement learning algorithms to solve the robotic pick-and-place problems for cases in which the geometrical models of the objects to be grasped are unknown [16]. In particular, using the deep reinforcement learning algorithm, the robot manipulator is able to determine a suitable pose (i.e., optimal action) to grasp certain types of objects. In [17], the image of an arbitrary pose of an object is used as an input to a distributed reinforcement learning algorithm. After learning, the robot is able to perform grasping tasks for objects that are either occluded or previously unseen. Deep reinforcement learning is also used in training robotic pushing/grasping/picking [18,19]. Kalashnikov et al. developed the QT-Opt algorithm and focused on scalable, off-policy deep reinforcement learning [20]. Seven real robot manipulators were used to perform and record more than 580 k grasp attempts for training a deep neural network. Once the learning process is complete, the real robot manipulator can successfully perform grasping, even for previously unseen objects [20]. Chen and Dai used a CNN to detect the image features of an object. Based on the detected image features of the object of interest, a deep Q-learning algorithm was used to determine the grasp pose corresponding to that object [21]. In [22], Chen et al. used a Mask R-CNN and PCA to estimate the 3D pose of objects to be grasped. Based on the estimated 3D object pose, a deep reinforcement learning algorithm is employed to train the control policy in a simulated environment. Once the learning process is complete, one can deploy the learned model to the real robot manipulator without further training.

This paper proposes an object grasping approach that combines the YOLO algorithm [23–26] and the Soft Actor-Critic (SAC) algorithm [27,28]. It is well known that YOLO is capable of rapidly detecting, localizing and recognizing objects in an image. In particular, YOLO can find the location of the object of interest inside the field of view of a camera and use this location information as the input to a reinforcement learning algorithm. Since the search of an entire image in not essential, training time can therefore be substantially reduced. SAC is based on the Actor-Critic framework and exploits Off-Policy to improve the sample efficiency. SAC maximizes the expected return as well as the entropy of policy simultaneously. Since SAC exhibits excellent performance and is suitable for real-world applications, this paper employs SAC to train the robot manipulator to perform object grasping through self-learning.

2. Framework

This paper develops a robotic object grasping technique that combines computer vision-based object detection/recognition/localization and a deep reinforcement learning algorithm with self-learning capability. Figure 1 shows the schematic diagram of the robotic pick-and-place system developed in this paper. As shown in Figure 1, YOLO will detect the object of interest from the image captured by the camera. SAC will provide the desired grasping point in the image plane based on the depth image information of the object bounding box. The grasping point on the 2D-image plane is converted to a desired 6D grasping pose in the Cartesian space so as to control the robot manipulator to grasp objects of interest and place them at a desired position. The system will return the reward information based on the reward mechanism.

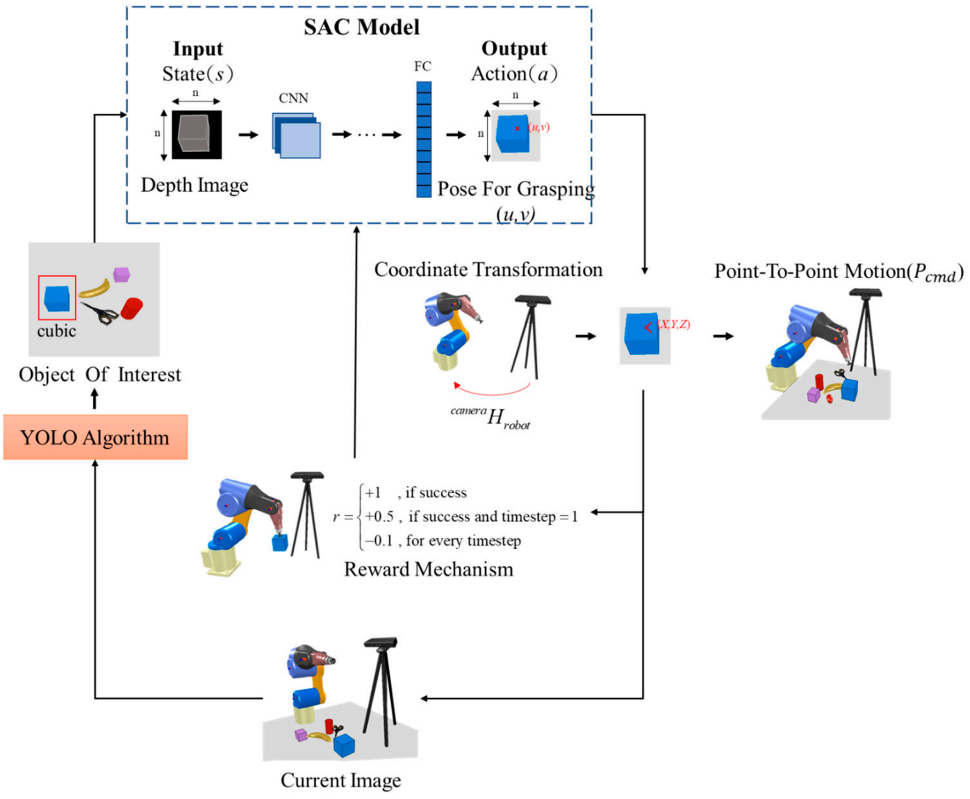

SAC Model
Input
State(s)
Output
Action(a)
CNN
FC
Depth Image
Pose For Grasping (u,v)
Coordinate Transformation
Point-To-Point Motion(P_{cmd})
cubic
Object Of Interest
YOLO Algorithm
${}^{camera}H_{robot}$
(X,Y,Z)
$r = \begin{cases} +1, & \text{if success} \\ +0.5, & \text{if success and timestep} = 1 \\ -0.1, & \text{for every timestep} \end{cases}$
Reward Mechanism
Current Image

Figure 1. Schematic diagram of robotic pick-and-place based on computer vision and deep reinforcement learning.

3. Object Recognition and Localization Based on YOLO Algorithms

In computer vision-based object recognition/localization applications, many past studies have adopted a two-step approach. The first step focuses on detecting and segmenting out the region that contains objects of interest within the image. The second step proceeds to object recognition/localization based on the region detected in the first step. Such an approach often consumes enormous computation resources and time. Unlike the two-step approach, YOLO can simultaneously detect and recognize objects of interest [23–26]. The schematic diagram of the YOLO employed in this paper is shown in Figure 2, where "Input" is the image input, "Conv" is the convolution layer, "Res_Block" is the residual block, and "Upsample" is the upsampling of image features. YOLO uses the Darknet-53 network structure to extract image features. In general, Darknet-53 consists of a series of 1×1 and 3×3 convolution layers. Each convolution layer has a Leaky ReLU activation function, a batch normalization unit and a residual block to cope with the problem of gradient disappearance/explosion caused by the large number of layers in the deep neural network. In addition, to improve the detection accuracy of small objects, YOLO adopts the

Feature Pyramid Network structure to perform multi-scale detection. The image input after processing by the Darknet-53 will output three different sizes of image features—13 × 13, 26 × 26 and 52 × 52. Object detection will be performed on these image features and the anchor box will then be equally distributed to three outputs. The final detection results will be the sum of the detection results of these three image features of different sizes.

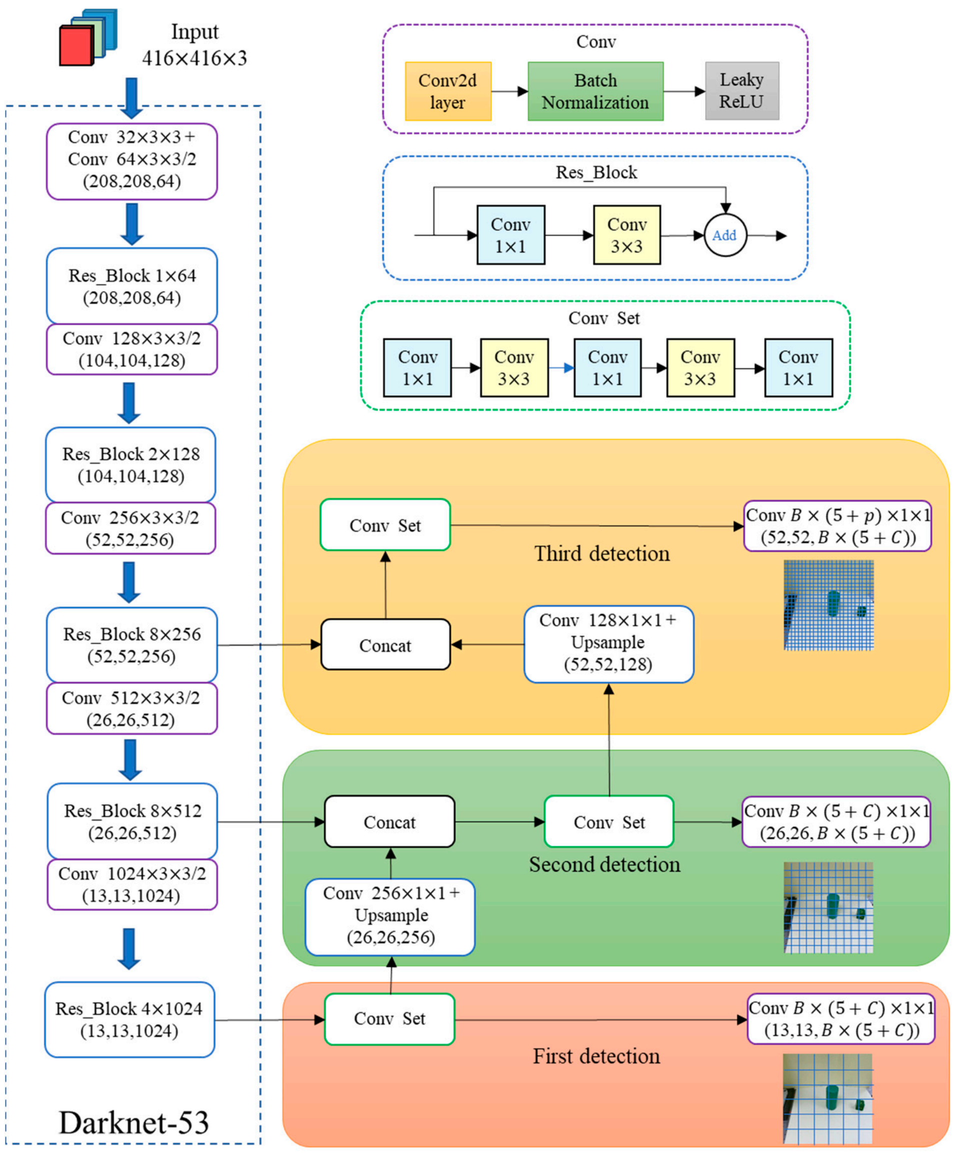

Input
416×416×3
Conv 32×3×3 + Conv 64×3×3/2 (208,208,64)
Res_Block 1×64 (208,208,64)
Conv 128×3×3/2 (104,104,128)
Res_Block 2×128 (104,104,128)
Conv 256×3×3/2 (52,52,256)
Res_Block 8×256 (52,52,256)
Conv 512×3×3/2 (26,26,512)
Res_Block 8×512 (26,26,512)
Conv 1024×3×3/2 (13,13,1024)
Res_Block 4×1024 (13,13,1024)
Darknet-53
Conv
Conv2d layer
Batch Normalization
Leaky ReLU
Res_Block
Conv 1×1
Conv 3×3
Add
Conv Set
Conv 1×1
Conv 3×3
Conv 1×1
Conv 3×3
Conv 1×1
Conv Set
Conv B × (5 + p) ×1×1 (52,52, B × (5 + C))
Third detection
Concat
Conv 128×1×1 + Upsample (52,52,128)
Concat
Conv Set
Conv B × (5 + C) ×1×1 (26,26, B × (5 + C))
Second detection
Conv 256×1×1 + Upsample (26,26,256)
Conv Set
Conv B × (5 + C) ×1×1 (13,13, B × (5 + C))
First detection

Figure 2. Schematic diagram of YOLO.

4. Object Pick-and-Place Policy Based on SAC Algorithms

SAC is a deep reinforcement learning algorithm [27,28] that can enable a robot to learn in the real world. The attractive features of SAC include: (1) it is based on the Actor-Critic framework; (2) it can learn based on past experience, i.e., off-policy, to achieve improved efficiency in sample usage; (3) it belongs to the category of Maximum Entropy Reinforcement Learning and can improve stability and exploration; and (4) it requires fewer parameters.

In this paper, both the state and action are defined in the continuous space. Therefore, SAC uses neural networks to parametrize the soft-action value function and the policy function as $Q_\theta(s_t, a_t)$ and $\pi_\phi(a_t|s_t)$, respectively. A total of five neural networks are constructed—two soft action-value networks $Q_{\theta_1}(s_t, a_t)$ and $Q_{\theta_2}(s_t, a_t)$; two target soft action-value networks $Q'_{\theta_1'}(s_t, a_t)$ and $Q'_{\theta_2'}(s_t, a_t)$; and one policy network $\pi_\phi(a_t|s_t)$, where θ_1,θ_2,$\theta_1{}'$,$\theta_2{}'$ and ϕ are the parameter vectors of the neural networks as shown in Figure 3. In particular, the policy function and the soft action-value function are the actor and the critic in the Actor-Critic framework, respectively. Under state s, the soft action-value function will output the expected reward $Q_\theta(s_t, a_t)$ for selecting action a, thus guiding the policy function $\pi_\phi(a_t|s_t)$ to learn. Based on the current state, the policy function will output an action to yield the system state for the next moment. By repeating these procedures, one can collect past experience to be used in training the soft action-value function. Since SAC is a random policy, the outputs of SAC are therefore the mean and standard deviation of probability distribution of the action space.

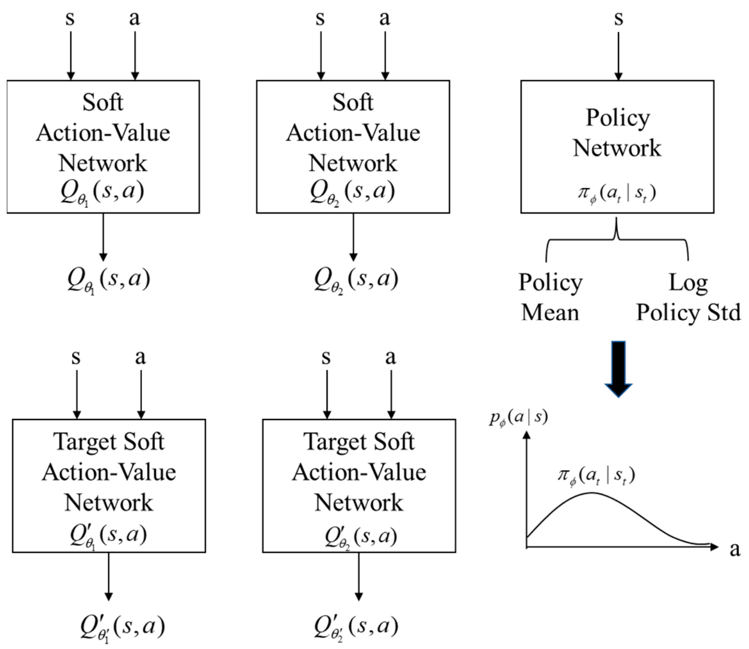

s
a
Soft Action-Value Network $Q_{\theta_1}(s,a)$
$Q_{\theta_1}(s,a)$
Soft Action-Value Network $Q_{\theta_2}(s,a)$
$Q_{\theta_2}(s,a)$
Policy Network $\pi_\phi(a_t|s_t)$
Policy Mean
Log Policy Std
$p_\phi(a|s)$
$\pi_\phi(a_t|s_t)$
Target Soft Action-Value Network $Q'_{\theta_1}(s,a)$
$Q'_{\theta'_1}(s,a)$
Target Soft Action-Value Network $Q'_{\theta_2}(s,a)$
$Q'_{\theta'_2}(s,a)$

Figure 3. Neural network architecture of SAC.

The objective function for the Soft Action-Value Network is described by Equation (1), while Equation (2) is the learning target. The Mean-Square Error (MSE) is employed to update the network parameters. The action-value network $Q_\theta(s_t, a_t)$ and the target action-value network $Q'(s_t, a_t)$ have the same network structure. The action-value network is used to predict the expected reward for executing action a under state s. The target action-value network is used to update the target so as to help train the action-value network. During training, only the action-value network will be trained, while the target action-value network will remain unchanged. In short, the target will change if the target action-value network updates, which will make it difficult for the learning of the neural network to converge.

$$J_Q(\theta) = E_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \tag{1}$$

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1}\sim p}[V_{\theta'}(s_{t+1})] \tag{2}$$

In this paper, the Stochastic Gradient Descent (SGD) method is employed to calculate the derivative of the objective function, as described by Equation (3):

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) - r(s_t, a_t) + \gamma(Q_{\theta'}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})))) \tag{3}$$

The weights of the target soft action value network are updated using Equation (4), where τ is a constant:

$$\theta'_{t+1} \leftarrow \tau\theta_t + (1-\tau)\theta'_t \tag{4}$$

The objective function of the policy network is described by Equation (5). To improve the policy, one should maximize the sum of action value and entropy:

$$\begin{aligned} J_\pi(\phi) &= E_{s_t\sim D, a_t\sim \pi_\theta}[\alpha \log(\pi_\phi(a_t|s_t)) - Q_\theta(s_t, a_t)] \\ a_t &= f_\phi(\varepsilon_t; s_t) \end{aligned} \tag{5}$$

where ε_t is the noise and Equation (5) can be rewritten as Equation (6):

$$J_\pi(\phi) = E_{s_t\sim D, \varepsilon_t\sim N}[\alpha \log(\pi_\phi(f_\phi(\varepsilon_t; s_t)|s_t)) - Q_\theta(s_t, f_\phi(\varepsilon_t; s_t))] \tag{6}$$

The derivative of the objective function of the policy network is described by Equation (7):

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \alpha \log(\pi_\phi(a_t|s_t)) + (\nabla_{a_t} \alpha \log(\pi_\phi(a_t|s_t)) - Q(s_t, a_t))\nabla_\phi f_\phi(\varepsilon_t; s_t) \tag{7}$$

The SAC reinforcement learning algorithm is illustrated in Figure 4.

```
Soft Actor-Critic
1.  Input: $\theta_1, \theta_2, \phi$                                  ➔Initial parameters
2.    $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2$           ➔Initial target network weights
3.    $D \leftarrow \varnothing$                                       ➔Initial empty replay buffer
4.    for each iteration do
5.      for each environment step do
6.        $a_t \sim \pi_\phi(a_t | s_t)$                               ➔Sample action from the policy
7.        $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$                         ➔Sample transition from the environment
8.        $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$   ➔Store the transition in the replay buffer
9.      end for
10.     for each gradient step do
11.       $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1,2\}$   ➔Update the Q-function parameters
12.       $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$         ➔Update policy weights
13.       $\theta_i' \leftarrow \tau\theta_i' + (1-\tau)\theta_i'$ for $i \in \{1,2\}$   ➔Update target network weights
14.     end for
15.   end for
16. Output: $\theta_1, \theta_2, \phi$                                 ➔Optimized parameters
```

Figure 4. SAC reinforcement learning algorithm.

4.1. Policy

This paper applies SAC to robotic object grasping. The learning agent is the 6-DOF robot manipulator, while the policy output is the coordinate (u,v) of the object grasping point on the image plane. The state, action and reward mechanism are designed as follows.

4.1.1. State (State *s*)

By exploiting YOLO, one can detect the objects of interest. The state of the SAC algorithm is defined to be the depth image of the object of interest. The state input designed in this paper is the depth information. Therefore, after obtaining the position of the object of interest in the RGB image, one needs to find its corresponding position in the depth image. Note that this depth image will be scaled to a size of 64 × 64. To be precise, the state used in this paper is a 64 × 64 × 1 depth image as shown in Figure 5.

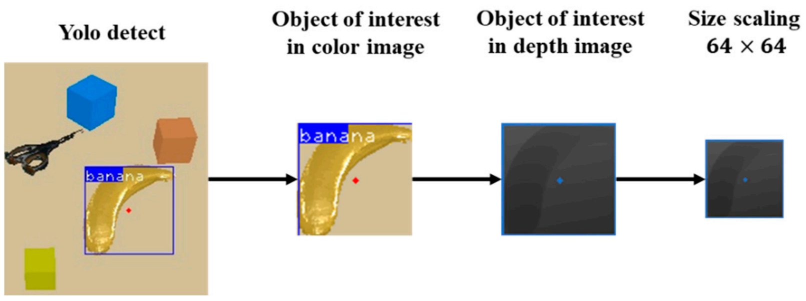

Yolo detect
Object of interest in color image
Object of interest in depth image
Size scaling 64 × 64
banana
banana

Figure 5. Illustrative diagram of state acquisition.

4.1.2. Action (Action *a*)

The action of SAC is defined to be the input displacement vector of the object of interest on the image plane as described by Equation (8), for which its unit is a pixel. The length and width of the bounding box obtained by YOLO are denoted as x and y, respectively. In addition, the coordinate of the center of the bounding box is denoted as (u_c, v_c). Equation (9) gives the displacement vector of the object of interest on the image plane corresponding to the action by the SAC. The coordinates of the object grasping point on the image plane as shown in Figure 6 are calculated using Equation (10). With the calculated image coordinates of the object grasping point, by using coordinate transformation, depth information and inverse kinematics, one can obtain the joint command for the 6-DOF robot manipulator to perform object grasping.

$$a = (a_1, a_2), \begin{cases} a_1 \in [-1, 1] \\ a_2 \in [-1, 1] \end{cases} \tag{8}$$

$$\begin{aligned} \Delta u &= 1 + (a_1 * x/2 - 0.99) \\ \Delta v &= 1 + (a_2 * y/2 - 0.99) \end{aligned} \tag{9}$$

$$\begin{aligned} u &= \Delta u + u_c \\ v &= \Delta v + v_c \end{aligned} \tag{10}$$

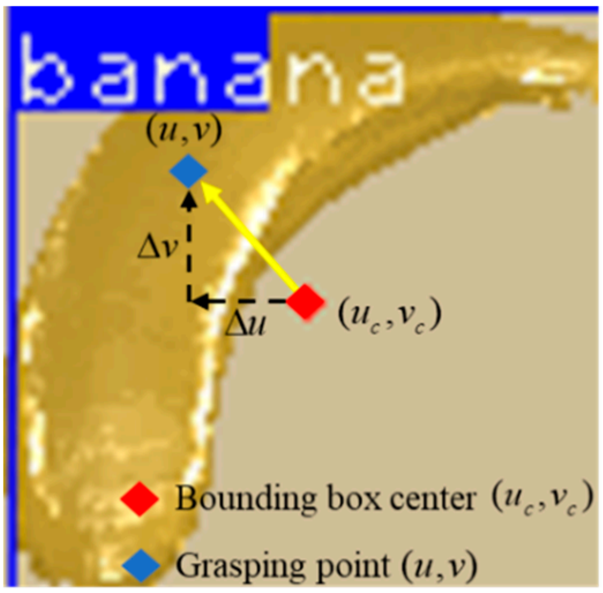


Figure 6. The displacement vector and the object grasping point on the image plane.

4.1.3. Reward (Reward, r)

A positive reward of 1 will be given if a successful object grasping occurs. In contrast, a negative reward -0.1 (i.e., penalty) will be given if failure occurs. As a result, the accumulated reward for an episode will be negative if the first ten attempts of object grasping fail. In order to help the learning agent find the optimal object grasping point as soon as possible, an extra positive reward 0.5 will be given if the first attempt of object grasping is successful. In addition, two termination conditions are adopted for the learning of SAC. To prevent the learning agent from continuously learning the wrong policy, if none of the first 100 object grasping attempts is successful, this episode will be terminated immediately. In addition, when the learning agent successfully performs object grasping, this episode will also be terminated immediately. The reward mechanism is described by Equation (11).

$$r = \begin{cases} +1 & , \quad \text{if successful} \\ +1.5 & , \quad \text{if successful and the number of attempts in object grasping} = 1 \\ -0.1 & , \quad \text{for each failure attempt in object grasping} \end{cases} \tag{11}$$

4.2. Architecture Design of SAC Neural Network

Since state s adopted in this paper is a $64 \times 64 \times 1$ depth image, a CNN is amended to the SAC so that the SAC can learn directly from the depth image. The hyperparameters of SAC are listed in Table 1 and its network architecture is shown in Figure 7. The input to the policy network is the depth image of the object of interest as detected by YOLO. The inputs to the soft action-value network and the target soft action-value network are comprised of the depth image of the object of interest as detected by YOLO and the policy outputted by the policy network. As shown in Figure 7, the policy network, the soft action-value network and the target soft action-value network all consist of three CNNs and four full connected neural networks. The activation functions used in the soft action-value network and the target soft action-value network are ReLU. As for the policy network, the activation functions for the three CNNs and the first three full connected neural networks are ReLU. The output of the last layer of the policy network is the displacement vector on the image plane, having both positive and negative values. Therefore, the hyperbolic tangent function (i.e., Tanh) is chosen as the activation function for the last layer of the policy network. Note that the three CNNs and the first fully connected neural network are used to extract image features.

Table 1. Hyperparameters of SAC neural network.

Hyperparameter	**Title 2**
optimizer	Adam
learning rate	0.001
replay buffer size	200,000
batch size	64
discount factor (γ)	0.99
target smoothing coefficient (τ)	0.005
entropy temperature parameter (α)	0.01

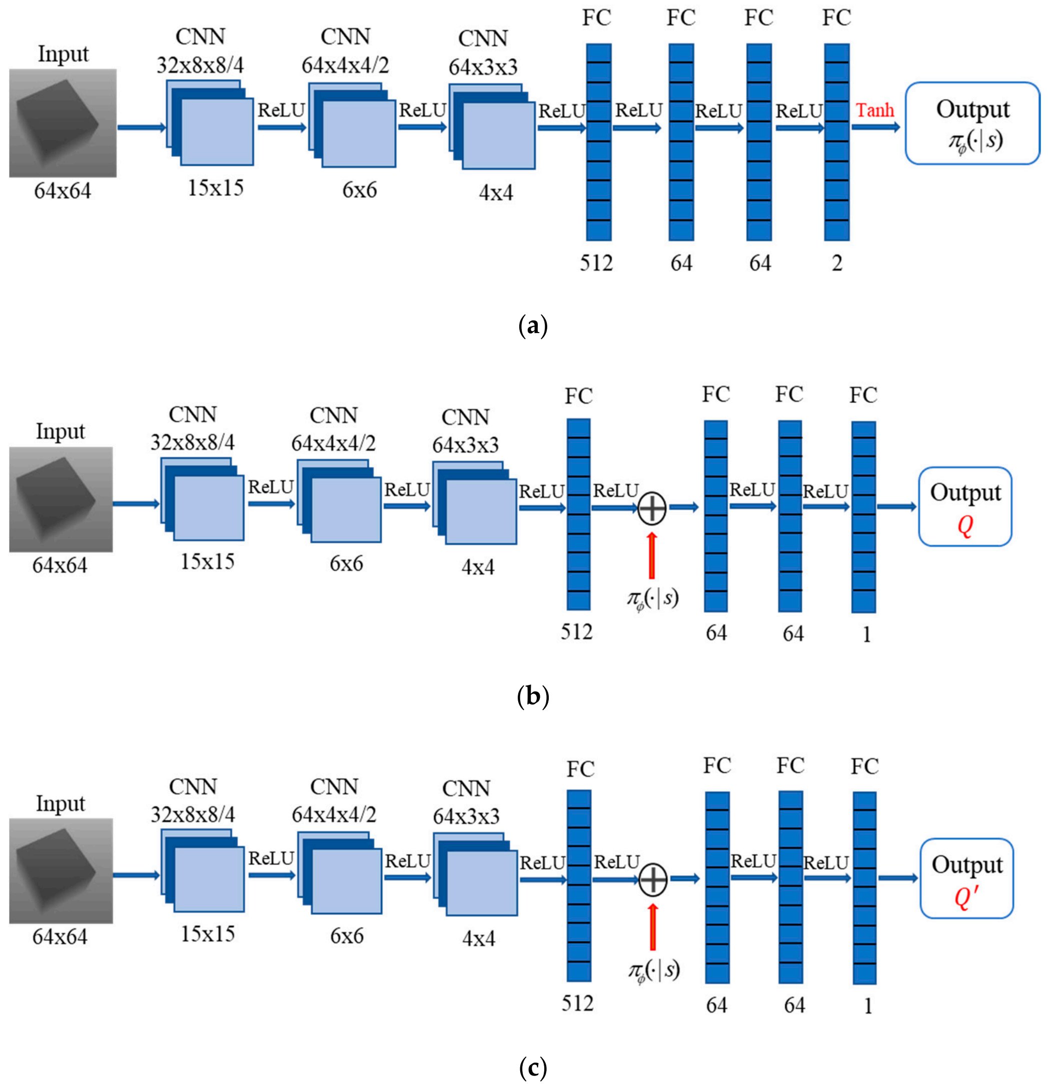
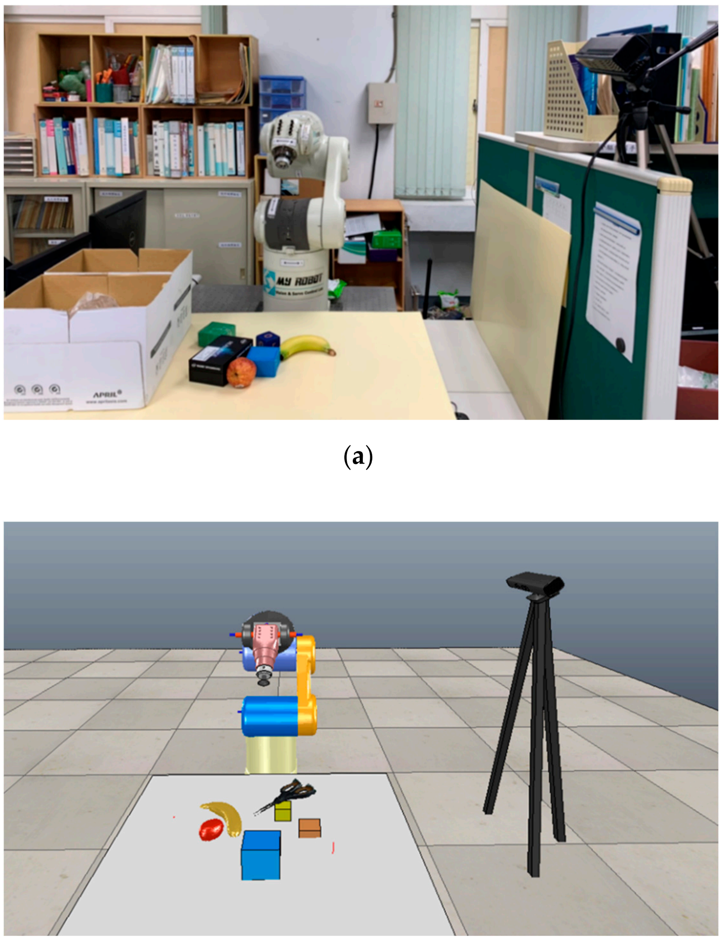


Figure 7. Architecture of SAC neural network. (**a**) Policy Network; (**b**) Soft Action-Value Network; (**c**) Target Soft Action-Value Network.

5. Experimental Setup and Results

The real experimental environment used in this paper is shown in Figure 8a, while Figure 8b shows the simulated environment constructed using the simulation platform V-REP. The simulated environment is mainly used to train and test the deep neural network. The 6-DOF A7 industrial articulated robot manipulator used in the real experiment is manufactured by ITRI. The Mitsubishi AC servomotors installed at each joint of the robot manipulator are equipped with absolute encoders and are set to torque mode. A vacuum sucker (maximum payload 3 kg) manufactured by Schmalz is mounted on the end-effector of the robot manipulator. The vision sensor used in the experiment is a Kinect v2 RGBD camera (30 Hz frame rate) manufactured by Microsoft. The maximum resolution for the RGB camera is 1920 $\times$ 1080 pixels, while the maximum resolution for the depth camera is 512 $\times$ 424 pixels. The Kinect v2 camera is located at the upper right side of the 6-DOF robot manipulator to capture the images of the objects. These object images will be used for YOLO to classify their categories. Two desktop computers are used in the experiment. The computer for controlling the 6-DOF robot manipulator and the vacuum sucker is equipped with Intel(R) Core TM i7-2600 CPU @3.40 Ghz and 12 GB RAM. It runs under Microsoft Windows 7 and uses Microsoft Visual Studio 2015 as its programming development platform. The computer responsible for computer vision, the training of the deep reinforcement learning network, and the V-REP robot simulator is equipped with a NVIDIA GeForce RTX 2080 Ti and 26.9 GB RAM. It runs under Microsoft Windows 10 and uses PyCharm as its development platform. The Python and the tool kit of the PyTorch are used in training the deep reinforcement learning network.


(**a**)

(**b**)

Figure 8. Experimental and simulated environment: (**a**) real experimental environment; (**b**) simulated environment.

5.1. Training Results of YOLO

As shown in Figure 9, the objects of interest used in the experiment included apples, oranges, a banana, a cup, a box and building blocks.

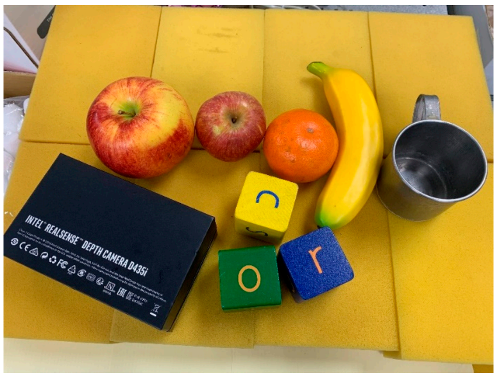


Figure 9. Objects of interest used in the experiment.

The COCO Dataset was used to train the YOLOv3 in this paper. However, the COCO Dataset does not include objects such as the building blocks used in the experiment. As a result, it was necessary to collect a training data set for the building blocks. In particular, a total of 635 images of the building blocks were taken. The transfer learning technique [29] was employed in this paper to speed up the training process, in which the weights provided by the authors of YOLO were adopted as the initial weights for training the YOLOv3. Figure 10 shows the training results of YOLO. The total number of iterations was 45,000. The value of the loss function converged to 0.0391. To test the performance of the trained YOLOv3, several objects were randomly placed on the table, with the detection results shown in Figure 11. Clearly, YOLOv3 can successfully detect and classify the objects of interest.

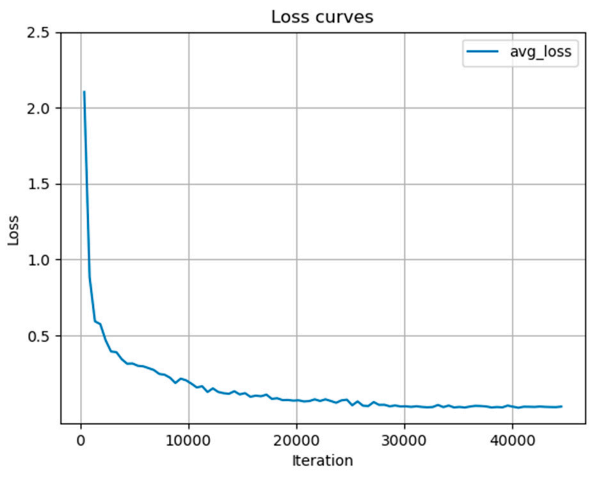


Figure 10. Training results of YOLOv3.

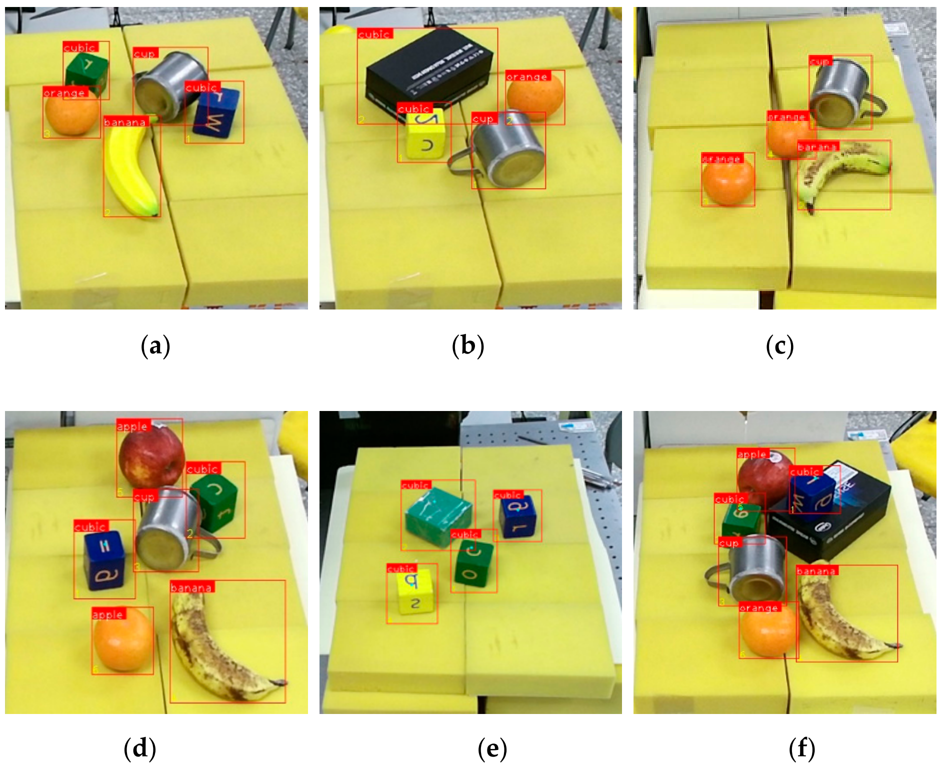


Figure 11. Detection/classification results of YOLOv3 after training (**a**) 1st test (**b**) 2nd test (**c**) 3rd test (**d**) 4th test (**e**) 5th test (**f**) 6th test.

5.2. Training and Simulation Results of Object Grasping Policy Based on SAC

Figure 12 illustrates the flowchart of the training process for the proposed object grasping approach based on SAC. At the beginning of each episode, the experimental/simulation environment was reset, namely, the robot manipulator was returned to the home position, objects were placed on the table, and the camera took images of the environment. Based on the image captured by the camera, the object recognition/localization approach based on YOLO developed in Section 3 was used to find the position of the object of interest so as to obtain its current state (s) (detailed procedures are indicated by the red dash block in Figure 12). According to its current state, the SAC would output an action (a), i.e., the input displacement vector of the object of interest on the image plane. The joint command of the robot manipulator could be obtained by using coordinate transformation, depth information and inverse kinematics. According to the obtained joint command, the end-effector was controlled to move to a desired position and a suction nozzle was turned on to perform object grasping. A positive reward was given for a successful grasp. The termination conditions for an episode occurred either when the total number of object grasping attempts was more than 100, or when an object grasping attempt was successful.

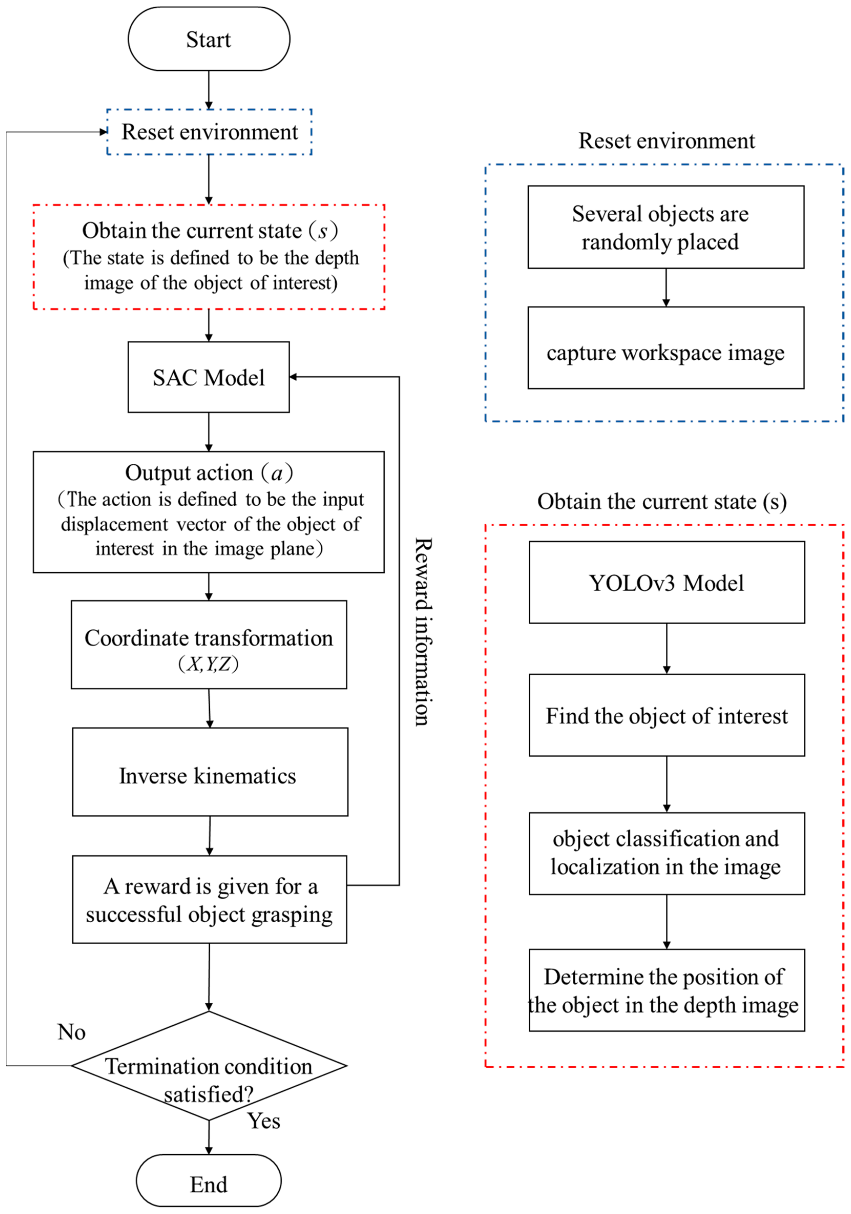
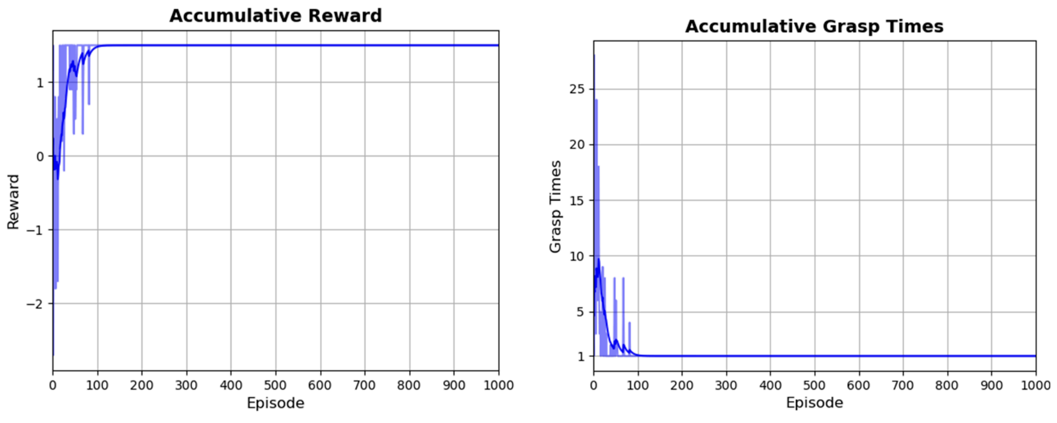

Start
Reset environment
Obtain the current state (s)
(The state is defined to be the depth image of the object of interest)
SAC Model
Output action (a)
(The action is defined to be the input displacement vector of the object of interest in the image plane)
Coordinate transformation
(X,Y,Z)
Inverse kinematics
A reward is given for a successful object grasping
Reward information
No
Termination condition satisfied?
Yes
End
Reset environment
Several objects are randomly placed
capture workspace image
Obtain the current state (s)
YOLOv3 Model
Find the object of interest
object classification and localization in the image
Determine the position of the object in the depth image

Figure 12. Flowchart of the training process for the proposed object grasping approach based on SAC.

In the real world, objects to be grasped are randomly placed. However, if the objects to be grasped are randomly placed for each episode in the training initially, the training time for learning object grasping successfully could be very long. In order to speed up the learning process, the idea of incremental learning is exploited in this paper to set up the learning environment. For instance, a building block was the object of interest for grasping. Firstly, the pose of the building block on the table was fixed and the deep reinforcement neural network was trained over 1000 episodes in the simulated environment constructed by the V-REP robot simulator. The training results are shown in Figure 13.



(**a**) (**b**)

Figure 13. Training results of a fixed pose building block (**a**) accumulated reward for each episode (**b**) number of grasping attempts for each episode.

As described in Equation (11), a positive reward of 1 will be given if the robot successfully grasps an object. In contrast, a negative reward -0.1 (i.e., penalty) will be given if the robot fails to grasp an object. That is, the accumulated reward for an episode will be negative if the robot needs more than ten attempts to successfully grasp an object. In addition, since an extra positive reward 0.5 will be given if the robot successfully grasps an object on its first attempt, the maximum accumulated reward for an episode will be 1.5. From the results shown in Figure 13, it was found that after 100 episodes of training, the 6-DOF robot manipulator was able to find a correct grasping pose for the case of a building block with a fixed pose.

After the 6-DOF robot manipulator could successfully grasp the building block with a fixed pose, the deep reinforcement neural network was retrained for another 1000 episodes. This time, the building block as well as other objects (used as the environmental disturbance) were randomly placed on a table. By exploiting the paradigm of transfer learning, the weights of the deep reinforcement neural network after learning for the case of fixed object poses were used as the initial weights for the deep reinforcement neural network in the retraining process. By taking into account the fact that objects of the same category may have different sizes or colors, for every 100 episodes in the retraining process, the colors and sizes of objects in each category were changed. This strategy served to enhance the robustness of the trained policy toward environmental uncertainty during verification in the real world. Figure 14 shows the training results for the case of randomly placed objects, where the yellow line represents the results of exploiting transfer learning (i.e., using the weights for the case of fixed object poses as the initial weights) and the purple line shows the results without using transfer learning. The results shown in Figure 14b indicate that the number of grasping attempts required to find correct grasping points without using transfer learning was much larger than that for using transfer learning over the first 200 episodes. Table 2 shows similar results in total training time and total number of grasping attempts for 1000 episodes.

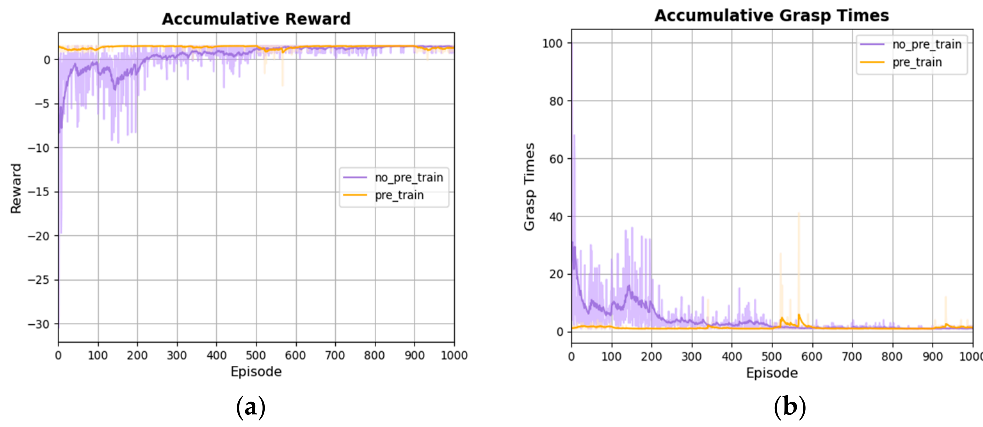


Figure 14. Training results for the case of randomly placed objects: the yellow line represents the results of exploiting transfer learning (i.e., use the weights for the case of fixed object poses as the initial weights), while the purple line shows the results without using transfer learning. (**a**) Accumulated reward for each episode; (**b**) number of grasping attempts for each episode.

Table 2. Total training time and total number of grasping attempts.

	Pre_Train (Use Transfer Learning)	**No_Pre_Train**	**Without_YOLO**
Training time	6443 (s)	15,076 (s)	102,580 (s)
Number of grasping attempts	1323 (attempts)	3635 (attempts)	38,066 (attempts)

Figure 15 shows the results of directly using the entire image (rather than using the object of interest detected by YOLOv3) as the input state for the deep reinforcement learning network. The results shown in Figure 15 indicate that correct grasping points cannot be obtained after 1000 episodes of training. Table 2 indicates that the training time for the case of using the entire image as the input is 15.9 times longer than that for using the proposed approach (i.e., transfer learning + YOLO + SAC). In addition, the number of grasping attempts for the case of using the entire image as the input is 28.8 times larger than that for using the proposed approach. The above simulation results reveal that the proposed approach indeed can effectively reduce the total training time and total number of grasping attempts.

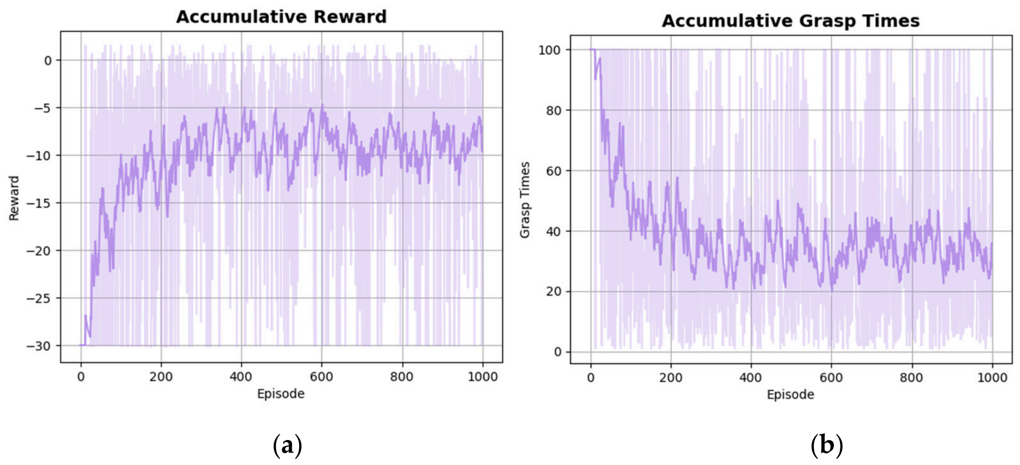


Figure 15. Results of the V-REP robot simulator without combining YOLOv3: (**a**) accumulated reward for each episode; (**b**) number of grasping attempts for each episode.

5.3. Object Grasping Using a Real Robot Manipulator

As mentioned previously, the input to the proposed deep reinforcement learning-based object grasping approach is the depth image (provided by Kinect v2) of the objects of interest detected by YOLOv3. Since YOLOv3 uses the RGB image (provided by Kinect v2) to detect the objects of interest, there is a need to construct the correspondence between the depth image and the RGB image so that the depth information of a point on the object of interest can be retrieved. In this paper, such a correspondence is constructed by using SDK accompanied with Kinect v2. In addition, with camera calibration [30] and the obtained depth information, the 3D information of a point on the object of interest in the camera frame can be retrieved. Hand-eye calibration [31] is then conducted to obtain the coordination transformation relationship between the camera frame and the end-effector fame. Using the results of hand-eye calibration and robot kinematics, the 3D information of a point on the object of interest in the camera frame can be converted into 3D infor-mation in the robot base frame. Moreover, using robot inverse kinematics, the joint com-mands for the robot to perform the task of grasping the object of interest can be obtained.

Figure 16 illustrates the flowchart for grasping a specific object. In this experiment, several different types of objects were randomly placed on a table. Note that the vacuum sucker mounted on the end-effector rather than a gripper is used in this paper to grasp the object of interest. In order to perform a successful grasp, the suction force needs to overcome the gravity force of the object of interest. As a result, the rim of the cup is not facing up in the experiment. The Kinect v2 camera took an image of the environment. The user assigned a specific object of interest for the robot manipulator to grasp. The SAC outputted a prediction of the position coordinate of the assigned object to be grasped. The joint command of the robot manipulator was obtained by using coordinate transformation, depth information and inverse kinematics. According to the obtained joint command, the end-effector was controlled to move to a desired position and a suction nozzle was turned on to perform object grasping. If the attempt for object grasping failed, the Kinect v2 camera took an image at the environment again and the object grasping process was repeated. If the attempts of object grasping failed three consecutive times, the task for grasping an assigned specific object was regarded as a failure.

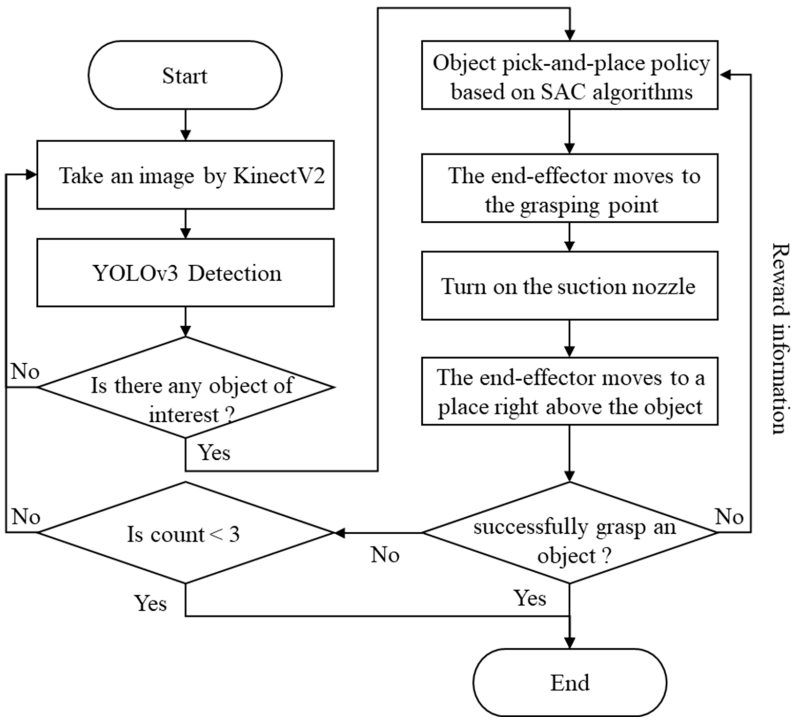

Start
Take an image by KinectV2
YOLOv3 Detection
Is there any object of interest ?
No
Yes
Object pick-and-place policy based on SAC algorithms
The end-effector moves to the grasping point
Turn on the suction nozzle
The end-effector moves to a place right above the object
Reward information
successfully grasp an object ?
No
Yes
Is count < 3
No
Yes
No
End

Figure 16. Flowchart for grasping a specific object.

In particular, SAC was employed to train a 6-DOF robot manipulator to grasp building blocks and bananas in a simulated environment constructed by a V-REP robot simulator. By exploiting the concept of Sim-to-Real [32], the trained network was deployed to the real 6-DOF robot manipulator to perform object grasping in the real world. In addition, in real-world experiments, objects such as apples, oranges and cups which are not in the training data set were added to the list of objects of interest. From the experimental results shown in Figure 17, it is evident that the trained SAC can indeed provide correct object grasping points for objects of interest in real-world environments. Experimental results for the success rate of grasping different objects are listed in Table 3.

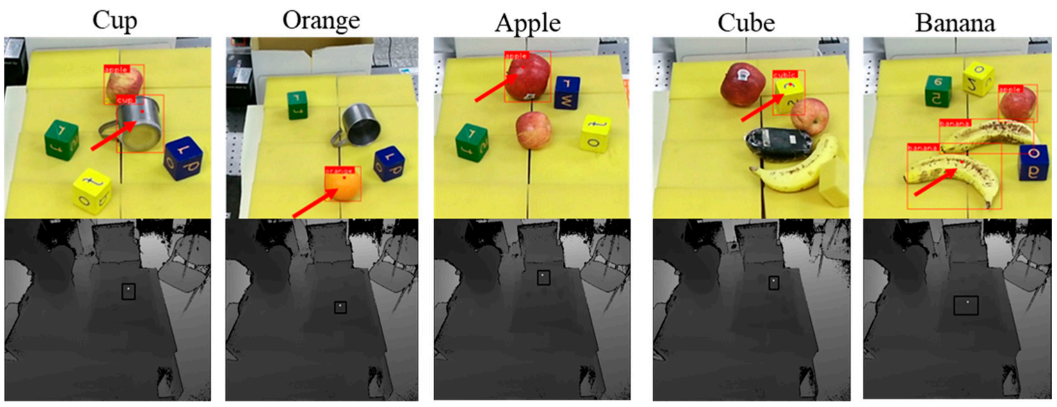


Figure 17. Object grasping point provided by SAC for different objects of interest (red point inside the bounding box in the upper environment image; white point in the lower depth image); the "arrow" sign is used to indicate the object of interest.

Table 3. Rate of successful grasping for different objects.

Object of Interest	Building Block	Apple	Banana	Orange	Cup
Rate of successful grasping	19/20	6/10	6/10	8/10	9/10
Object is in the training set	yes	no	yes	no	no

The results listed in Table 3 indicate that for the objects in the training set, the building block has a much higher rate of being successfully grasped than the banana. The reason for this discrepancy is that in the simulated environment, the banana has a fixed shape and smooth surface. However, the bananas used in real-world experiments have different shapes/sizes and their surfaces are not smooth enough. Therefore, the significant differences between the simulated environment and that of the real-world experiment lead to a lower rate of successful grasping for bananas. As for the objects not in the training set, the apples had the lowest rate of being successfully grasped. One possibility is that the two apples used in the real-world experiments have significant differences in size/shapes. In addition, in real-world experiments, hand-eye calibration error and robot calibration errors all contribute to the fact that the end-effector cannot 100% accurately move to the grasping position determined by the proposed deep reinforcement learning-based object grasping approach. Since bananas and apples require a more accurate grasping point, it is not surprising that their rates of being successfully grasped are lower.

In summary, there are several interesting observations from the experimental results. First of all, the suction nozzle used in this paper requires a smooth object surface to achieve successful grasping. That explains why apples and bananas have lower successful grasping rates. Secondly, without further training, the proposed approach exhibits decent grasping performance, even for cases in which the objects of interest are previously unseen. Thirdly, experimental results indicate that the SAC can be trained in the robot simulator and the trained SAC can be deployed to the real 6-DOF robot manipulator to successfully perform object grasping in the real world.

The next experiment was to grasp and classify all the objects randomly placed on the table and to put the grasped objects into the bin where they belonged. First of all, several objects were randomly placed on the table, after which YOLOv3 detected and classified all of the objects on the table. The SAC then provided information for the grasping points corresponding to all the objects of interest to the robot manipulator. The 6-DOF robot manipulator then performed the grasping task and put the grasped objects into their respective bins. Note that during the grasping process, the robot manipulator may collide with other objects so that their poses may change and result in grasping failures. In order to deal with the aforementioned problem, after performing the object grasping task, if some objects remained on the table, the object grasping tasks were repeated until all of the objects on the table had been grasped and correctly put into the bin. Figure 18 shows an image sequence of the object grasping/classification experiment.

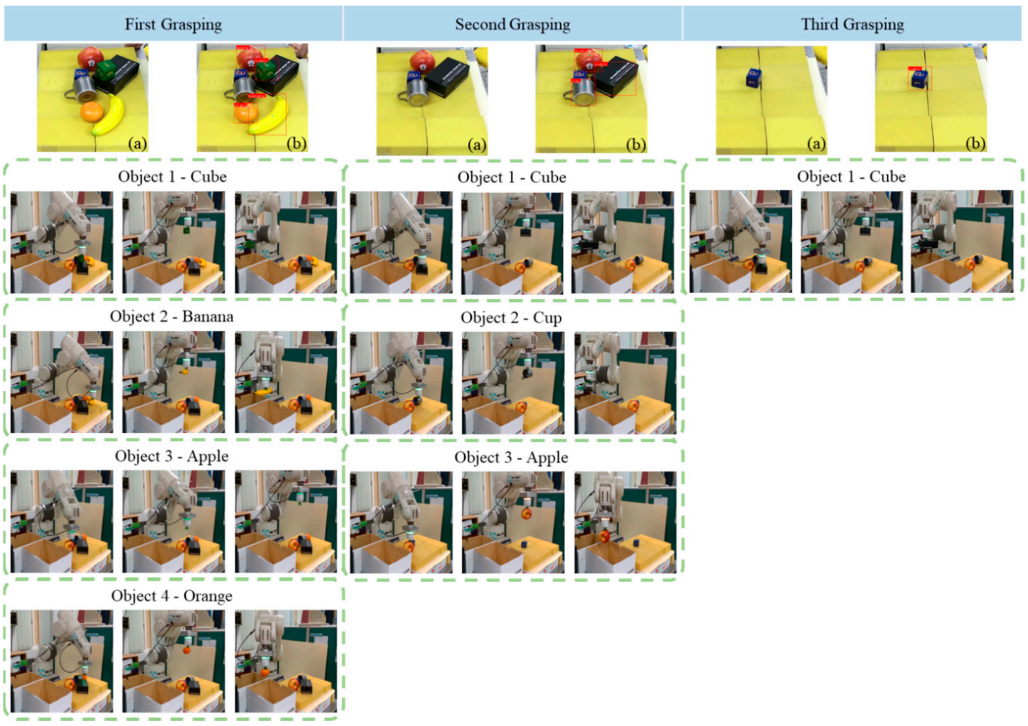


Figure 18. Image sequence of object grasping/classification experiment (**a**) original image (**b**) classification results of YOLOv3.

6. Conclusions

This paper proposes an approach that combines YOLO and deep reinforcement learning SAC algorithms for the 6-DOF robot manipulator to perform object grasping/classification through self-learning. In particular, the objects of interest in this paper are detected by YOLOv3. By considering the fact that objects of the same type may have different colors, only their depth images provided by Kinect v2 are thus used as the inputs for the proposed deep reinforcement learning-based object grasping approach. In this way, the exploration space can be substantially reduced so as to improve the success rate and enable SAC to converge quickly. Moreover, to speed up the training process, a V-REP robot simulator is employed to construct a simulated environment to train the SAC. Simulation results indicate that the proposed approach can indeed effectively reduce the total training time and the total number of grasping attempts compared with an approach that directly uses the entire image as the input state for the deep reinforcement learning network. In addition, to further speed up the training process, the paradigms of transfer learning and incremental learning are employed in the proposed approach. Moreover, the trained SAC was transferred to a real 6-DOF robot manipulator for real-world verification. Experimental results indicate that in using the proposed approach, the real 6-DOF robot manipulator successfully performed object grasping/classification, even for previously unseen objects.

Author Contributions: Conceptualization, Y.-L.C., Y.-R.C. and M.-Y.C.; methodology, Y.-L.C. and Y.-R.C.; software, Y.-L.C. and Y.-R.C.; validation, Y.-L.C. and Y.-R.C.; formal analysis, Y.-L.C. and Y.-R.C.; writing original draft preparation, Y.-L.C., Y.-R.C. and M.-Y.C.; writing review and editing, M.-Y.C.; project administration, M.-Y.C.; Funding acquisition, M.-Y.C.; supervision, M.-Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan, grant number MOST 108-2221-E-006-217-MY2.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Kyprianou, G.; Doitsidis, L.; Chatzichristofis, S.A. Collaborative Viewpoint Adjusting and Grasping via Deep Reinforcement Learning in Clutter Scenes. *Machines* **2022**, *10*, 1135. [CrossRef]
2. Johns, E.; Leutenegger, S.; Davison, A.J. Deep learning a grasp function for grasping under gripper pose uncertainty. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Republic of Korea, 9–14 October 2016; pp. 4461–4468. [CrossRef]
3. Lenz, I.; Lee, H.; Saxena, A. Deep learning for detecting robotic grasps. *Int. J. Robot. Res.* **2015**, *34*, 705–724. [CrossRef]
4. Pinto, L.; Gupta, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 3406–3413. [CrossRef]
5. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **2018**, *37*, 421–436. [CrossRef]
6. Mahler, J.; Pokorny, F.T.; Hou, B.; Roderick, M.; Laskey, M.; Aubry, M.; Kohlhoff, K.; Kröger, T.; Kuffner, J.; Goldberg, K. Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 1957–1964. [CrossRef]
7. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv* **2017**, arXiv:1703.09312. [CrossRef]
8. Mahler, J.; Matl, M.; Liu, X.; Li, A.; Gealy, D.; Goldberg, K. Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 5620–5627. [CrossRef]
9. Mahler, J.; Matl, M.; Satish, V.; Danielczuk, M.; DeRose, B.; McKinley, S.; Goldberg, K. Learning ambidextrous robot grasping policies. *Sci. Robot.* **2019**, *4*, eaau4984. [CrossRef]
10. Zhang, H.; Peeters, J.; Demeester, E.; Kellens, K. A CNN-Based Grasp Planning Method for Random Picking of Unknown Objects with a Vacuum Gripper. *J. Intell. Robot. Syst.* **2021**, *103*, 1–19. [CrossRef]
11. Morrison, D.; Corke, P.; Leitner, J. Learning robust, real-time, reactive robotic grasping. *Int. J. Robot. Res.* **2020**, *39*, 183–201. [CrossRef]
12. Fang, K.; Zhu, Y.; Garg, A.; Kurenkov, A.; Mehta, V.; Li, F.F.; Savarese, S. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *Int. J. Robot. Res.* **2020**, *39*, 202–216. [CrossRef]
13. Ji, X.; Xiong, F.; Kong, W.; Wei, D.; Shen, Z. Grasping Control of a Vision Robot Based on a Deep Attentive Deterministic Policy Gradient. *IEEE Access* **2021**, *10*, 867–878. [CrossRef]
14. Horng, J.R.; Yang, S.Y.; Wang, M.S. Self-Correction for Eye-In-Hand Robotic Grasping Using Action Learning. *IEEE Access* **2021**, *9*, 156422–156436. [CrossRef]
15. Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; Levine, S. How to train your robot with deep reinforcement learning: Lessons we have learned. *Int. J. Robot. Res.* **2021**, *40*, 698–721. [CrossRef]
16. Gualtieri, M.; Ten Pas, A.; Platt, R. Pick and place without geometric object models. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018; pp. 7433–7440. [CrossRef]
17. Fujita, Y.; Uenishi, K.; Ummadisingu, A.; Nagarajan, P.; Masuda, S.; Castro, M.Y. Distributed reinforcement learning of targeted grasping with active vision for mobile manipulators. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 9712–9719. [CrossRef]
18. Zeng, A.; Song, S.; Welker, S.; Lee, J.; Rodriguez, A.; Funkhouser, T. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 4238–4245. [CrossRef]

19. Deng, Y.; Guo, X.; Wei, Y.; Lu, K.; Fang, B.; Guo, D.; Liu, H.; Sun, F. Deep reinforcement learning for robotic pushing and picking in cluttered environment. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019; pp. 619–626. [CrossRef]
20. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. QT-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv* **2018**, arXiv:1806.10293. [CrossRef]
21. Chen, R.; Dai, X.Y. Robotic grasp control policy with target pre-detection based on deep q-learning. In Proceedings of the 2018 3rd International Conference on Robotics and Automation Engineering, Guangzhou, China, 17–19 November 2018; pp. 29–33. [CrossRef]
22. Chen, Z.; Lin, M.; Jia, Z.; Jian, S. Towards generalization and data efficient learning of deep robotic grasping. *arXiv* **2020**, arXiv:2007.00982. [CrossRef]
23. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
24. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
25. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. [CrossRef]
26. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
27. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
28. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2019**, arXiv:1812.05905. [CrossRef]
29. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
30. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
31. Cai, C.; Somani, N.; Nair, S.; Mendoza, D.; Knoll, A. Uncalibrated stereo visual servoing for manipulators using virtual impedance control. In Proceedings of the 13th International Conference on Control Automation Robotics & Vision, Singapore, 10–12 December 2014; pp. 1888–1893.
32. Peng, X.B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018. [CrossRef]