Grasping Control of a Vision Robot Based on a Deep Attentive Deterministic Policy Gradient

XIANGXIN JI\textsuperscript{1}, FENG XIONG\textsuperscript{1}, WEICHANG KONG\textsuperscript{2}, DONGFEI WEI\textsuperscript{1}, AND ZEYAN SHEN\textsuperscript{1}

\textsuperscript{1}School of Mechatronics Engineering and Automation, Shanghai University, Shanghai 200444, China
\textsuperscript{2}Jerry Zhaoxin Information Technology Company Ltd., Shanghai 201210, China

Corresponding author: Feng Xiong (xfeng@mail.shu.edu.cn)

This work was supported in part by the Pudong New Area Science and Technology Development Fund, Shanghai, China, under Grant PKX2020-R16.

ABSTRACT Reinforcement learning can achieve excellent performance in the field of robotic grasping if the grasping target is stable. However, during applications in the real world, robot needs to overcome the effects of a complex working environment with different types of target objects, so it is more difficult to maintain the quality of action planning, even in the same scene. In order to make an agent have the ability to plan actions in a more adaptive way, the deep attentive deterministic policy gradient algorithm is applied in this article. An attention region proposal network is used to select the message of the pre-exploration area. Then this message is calculated using the adaptive exploration method to regulate the strategy as the target changes. Furthermore, a stratified reward function, which is used to reduce the negative influence of miscellaneous information brought by the sparse reward matrix, is defined according to the distance between the end effector and the center of the pre-exploration area. The results show that the DADPG is able to produce a robust strategy with noise interference, and can train in a more efficient way due to the hierarchical reward function.

INDEX TERMS Robotic grasping, reinforcement learning, attentive exploration method, stratified reward function.

I. INTRODUCTION

With the continuous integration of informatization and industrialization, robotics technology has also continuously developed, and distinct types of robots are widely used in different areas, such as military, industry and medicine. In these application fields, it is necessary for agents to be robust with completely different task scenarios, and to specify the control strategy as the environment changes. The grasping task is one of the basic tasks in the field of robot control, and many works have studied the task. Traditional ways to accomplish grasping missions include analytical methods and empirical methods. When addressing practical problems, analytical methods have limitations because it is usually difficult to establish a high precision mathematical model that can describe the entire task scene, but the quality of the result are highly dependent on the model. The empirical method forces robots to simulate human’s grasping strategy, which avoids the need for high accuracy mathematical and physical models. Common empirical methods include knowledge based methods and supervised learning methods. Early scholars manually extracted series of scale invariant local feature shapes from the images of target objects, and then used these features for object detection [1]. Shotton from Cambridge University used texture based features to detect objects [2], and then created a data set that contained mapping relations from data collected from robots (e.g. actions, positions, and postures) to object features. This dataset then acted as a data source for the subsequent supervised learning. However, when an agent is used to solve problems with complex environments or dense objects, such mapping relations cannot be established in a clear way. Then, an obstacle-clear method [3] appeared to overcome the problem. When vertical pixel of an object coincides with the barrier’s pixel, the agent will take actions to clean up these nontarget objects. This type of method will also encounter difficulties when there is no contact between objects, and the agent is unable to plan action correctly if the camera angle dose not allow the image quality to reach the requirement. Then Dogar and Srinivasa proposed an action sequence planning algorithm [4]. The action sequence did not rely on the object position; however, regarding the
performance in simulation environments, when intricate tasks were encountered, the agent first determined the sequence of actions and then implemented these actions according to the established plan. Empirical methods achieve impressive performance when the mission environment is relatively fixed but can be limited easily when environments are easily influenced by external factors such as light, shadows, etc. Later, along with the development of deep learning (DL), traditional methods gradually lagged behind in training time and accuracy. Therefore, most of the current grasping strategies are generated by deep learning methods.

Deep reinforcement learning (DRL) has achieved rapid development since 2013; and classical algorithms such as the DQN [5], DDPG [6], DDQN [7], TRPO [8], PPO [9], etc. appeared gradually. DRL is a type of self-learning method that has splendid advantages, especially in high dimension problems. Currently, missions aimed at continuously controlling robots have already become Open AI Gym fundamental simulation tasks; and DRL structures have been proven to be efficient in solving missions such as grasping [10], [11], moving [8], [12], [13], and targeting. These tasks also become major methods to verify the quality of DRL algorithms. Using DRL endows agents with the ability to make decisions automatically when encountering changing environments due to different tasks or external factors. changing environments due to different tasks or external factors. Key information is extracted from environmental images and sensors by a deep neural network (DNN), and then the agent searches for the optimal strategy using the chosen DRL algorithm. Currently, in most RL algorithm training processes, an action will obtain a reward equals 1 if task is finished while all other rewards are all equal 0. Then, deep Q function training is conducted using a pre-collected action-reward dataset. The Q function is used here to evaluate the position and posture of a robot, and the final result achieves a grasping success rate of 96%. Breyel et al. also performed relevant research on the relationship between the reward function, pretrained model and DRL to determine the correlation between the success rate and training speed. The model [14]. Furthermore, Clavera et al. proposed a reward guiding method to prevent the training mask R-CNN can achieve good performance only when the quality of the algorithm [18]. Since the mask R-CNN can achieve good performance only when the quality of the training mask R-CNN can recognize only the difference between the foreground and background; therefore, large amounts of unrelated information will be fused in the subsequent training phase, making the network environment crowded.

Regarding the issues above, an improved deep deterministic policy algorithm based on the attentive exploration method (DADPG) was proposed in this paper. Figure1 shows a flowchart of the proposed method for grasping control missions. The DADPG uses the attentive exploration method (AE method) to extract pre-exploration messages from input images and concatenates them with gripper features as the new state, which is sent into the following actor-critic network (AC network). For the agent, the state is redefined, and a pre-exploration hierarchical reward function is implemented. The originality and contributions of this study are as follows:

- An AE network that conducts pretreatment for every image taken at the beginning of each episode, calculates the object information with an attention RPN structure, obtains pre-exploration information, and reduces the network load is designed.
- A new state, which includes image features, pre-exploration features and gripper features, that serves as the input of the AC network is designed.
- A pre-exploration hierarchical reward function is designed, and the reward for each action is calculated according to the distance between the center position of the pre-exploration area and the gripper’s end effector.

The remainder of this paper is organized as follows: The DADPG network structure is described in Section 2. The environment that an agent used to learn the grasping strategy is proposed in Section 3. Section 4. specifically introduces the comparison and verification experiment measuring the effect of DADPG. The conclusion is given in Section 5.

II. DADPG
A. BACKGROUND KNOWLEDGE
1) RL
Reinforcement learning considers interaction tasks between an agent and the environment. These tasks include a series of actions, observations and rewards. Actions represent the interactions between an agent and the environment in order to complete task. An observation is a message that is generated by an agent from the environment in every time slice.
A reward defines the goodness of an action after an agent applies it to the environment. RL problems can all be turned into a Markov determination process problems (MDPs). In an MDP problem, the state of the next time slice is correlated only with the state of the current time slice and is describe’d as follows:

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_t, s_{t-1}, \ldots, s_1, s_0)$$ (1)

where $s_t$ is the environmental state at time $t$. A basic MDP problem can be represented as $(S, A, P)$, where $S$ is the state, $A$ is the action and $P$ is the state transition probability matrix. Considering that multiple actions can be chosen in each given state, and that the state will change after the chosen action is executed, the RL agent pays more attention to the value of the action. It is obvious that agents will always choose actions with higher action values to execute. The value of an action is defined as an action value function, written as follows:

$$Q^\pi(s, a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s, a]$$

$$= E[r + \gamma Q^\pi(s', a')|s, a]$$ (2)

### 2) DDPG

The deep deterministic policy gradient (DDPG), a type of off-policy RL method proposed [6] by Lillicrap et al., is used to train the decision-making ability of agents. The DDPG uses a neural network structure to approximate the action value function with the current state, and the actor network exports an action with the maximum action value. Then, the output action and current state are sent into the critic network, and the agent obtains an evaluation from the critic network. The optimal policy proceeds in the direction of minimizing the following temporal difference error $\delta$ under a greedy policy:

$$\delta = [r + \gamma \max_{a'} Q^\pi(s', a') - Q^\theta(s, a)]^2$$ (3)

where $r$ is the reward generated after an action is applied to the environment, $\gamma$ is a discount factor, $Q_\theta(s, a)$ is a Q network with parameter $\theta$, that generates an action value function with current state $s$ and chooses action $a$, and $\max_{a'} Q^\pi(s', a')$ is the max action value of the next state generated by network Q with parameter $\theta'$.

### B. DADPG STRUCTURE

The DADPG network structure includes three parts: a feature fusion network for the input image and end effector, an AE network, an AC network. Figure 2 shows the entire network structure of the DADPG. The environmental images generated by robot camera $I_{query}$, target image $I_{support}$ and end effector state $grip_{state}$ are regarded as inputs of the DADPG. First, a feature fusion network is proposed here to extract the features from the input data. Second, $I_{query}$ and $I_{support}$ are processed by a weight-sharing network and then sent as input into the AE network to calculate the pre-exploration information. These pre-exploration data and the output of the feature fusion network are concatenated as the input state of the AC network, and the AC network calculates value Q with the state and action.

### C. FEATURE FUSION NETWORK

Since the input of the DADPG is mixed with image features and gripper end effector features, a feature extraction network is necessary. The feature extraction network in this paper includes two parts: a matrix feature extraction network and a vector feature extraction network.

The matrix feature extraction network consists of 2 types of convolution layers and a fully connected layer. Each convolution layer is followed by a pooling layer, which is used here to compress the amount of data and reduce the probability of overfitting. The vector feature extraction network includes 2 types of fully connected layers. Each fully connected layer is activated by a ReLU function, and then two outputs are concatenated together as the input data of the later training step. Figure 3 shows the structure of the feature fusion network.
The AE method includes 2 parts: the attention RPN and the adaptive exploration method.

1) ATTENTION RPN
In an actual environment, messages collected by sensors are easily influenced by various types of noise. This message leads to an image with too much unrelated information, and the relevant message only includes a small ratio. If this type of image is considered as the input of the training network, it will largely limit the training speed and even negatively affect the grasping success rate. Therefore, aiming at improving the training efficiency, it is essential to pretreat images and extract as much information that is related to target objects as possible before they are sent into the RL network. In this paper, the attention RPN is proposed to address this type of problem.

The attention RPN was first proposed by Qi Fan et al. in a few shot object detection network structure (FSOD), which was first applied in few shot object detection problems. Query images and support images are first pretreated by the depth correlation method. Target information (foreground) is separated from non-target information (background) by the depth correlation method. Then, the processed image is sent into the region proposal network (RPN) to perform the object detection task. This type of work reduces the amount of information to a certain extent, improves the training efficiency in a valid way, and provides the model with the ability to detect target objects with only a small amount of support data [19]. Figure 4 shows the structure of the attention RPN. In the DADPG task, input data also needs to be pretreated to alleviate the problem of inefficient learning. Background information and noise are separated from environmental information and target information, so the attention RPN is introduced here to preprocess the input image. $I_{query}$ and $I_{support}$ are sent into the attention RPN as inputs, and the features of these 2 types of images are extracted by a weight-sharing network first. Then, $I_{query}$ is classified into different layers by the depth wise convolution. The depth wise convolution is part of Xception which was first proposed by Chollet to reduce the number of parameters and operating costs [20]. $I_{support}$ is turned into several $1 \times 1$ kernels by the depth correlation method and average pooling method, and each kernel performs a convolution operation on the corresponding layer of $I_{query}$. The depth wise convolution is proposed here to compress the model and reduce the amount of pending data. The specific formula is written as follows:

$$G_{h,w,c} = \sum_{i,j} X_{i,j,c} \cdot Y_{h+i-1,w+j-1,c} , \quad i, j \in 1, \ldots, S$$

where $G$ is the result map of the attention feature, $X$ is a $1 \times 1$ kernel generated by $I_{support}$, and $Y$ is the pending data from $I_{query}$. If $I_{query}$ has areas that are very similar to $I_{support}$, the activated value in these areas will increase, which makes it easier for the network to detect it from the entire image. Then, the RPN is introduced to generate the regions of interest [21] and select the anchor areas that have the three largest IoU.
values as the pre-exploration areas. Figure 5 shows the feature maps generated by the attention RPN.

2) ADAPTIVE EXPLORATION METHOD
Since the pre-processing method mentioned above is a few shot training method, inaccurate messages will appear at the beginning. However, in the DDPG process, inaccurate information will lead to a series of problems, such as long stay times in non-target areas, inefficient training and network convergence problems, etc. In order to address such difficulties, an adaptive method based on the result of the attention RPN is proposed here.

The key point generated by the result of the attention RPN is considered to be the center of the pre-exploration area, and the radius of this area will adaptively increase along as the number of manipulator operating steps increases. Once the distance between the target object and the end effector is less than the settled threshold, the agent is deemed to have reached the target position.

In the early training step, if the end effector has difficulty reaching the target object after a certain number of steps, this means that the pre-exploration is inaccurate, and the pre-exploration radius will enlarge until the maximum value is reached as an offset. The update function is shown as follows:

\[
R_{\text{predict}} = \begin{cases} 
R_{\text{min}}, & e_n < e_{\text{thr}}, \\
\min(R_{\text{min}} + \delta^+, R_{\text{max}}), & e_n > e_{\text{thr}}.
\end{cases}
\]

(5)

where \(R_{\text{predict}}\) is the radius of the pre-exploration area; \(R_{\text{min}}\) and \(R_{\text{max}}\) are the extremum of the exploration radius; \(e_n\) is the number of current steps; \(e_{\text{thr}}\) is the threshold number of operation steps, which is equal to 5 in this paper; and \(\delta^+\) is the growth rate of the exploration radius. The exploration process terminated when the end effector reaches the target position, and the absolute distance between the actual target coordinates and predicted coordinates is recorded. Figure 6 shows the results of the adaptive exploration method used on those work spaces from Figure 5. In each picture slice of Figure 6, a smaller circle is the initial exploration area of every grasping task, and will increase to the widest exploration area as the larger one.

The RPN is updated by a binarized anchor, and it performs a regression on a positive anchor. However, the accuracy of the position is more important in the DADPG, so the gradient descent direction of the absolute distance mentioned above is considered a more important part of the attention RPN loss function. The new loss function is as follows:

\[
L(\theta, w) = \frac{1}{N} \sum L_{\text{reg}}(||P_{\text{predict}}(\theta, w) - P_{\text{actual}}||^2)
\]

(6)

where \(L_{\text{reg}}\) is the regression loss used to calculate the distance between the \(i\)th predicted position \(P_{\text{predict}}(\theta, w)\) and the actual position \(P_{\text{actual}}\).

The AE method addresses with the prepossessing job for the input image, and obtains a predicted position and a loss value of the absolute distance as result. The predicted position is then sent into the AC network, while the loss value is only used to update the AE network and dose not participate in the future training of the DADPG agent.

E. ACTOR-CRITIC NETWORK
The AC network is the key to the DDPG algorithm. The AC network includes the actor network, target actor network, critic network and target critic network. Figure 7 shows the AC network structure of the DADPG. The details of each network are shown as follows:

- **Actor:** The actor network is responsible for the iterative update of the policy network parameter \(\theta_A\) and helps agent select action \(A\) according to the current state \(S\).
- **Target Actor:** The target actor network is responsible for selecting the “best” action \(A_{\text{next}}\) according to the next state \(S_{\text{next}}\) that is sampled from the replay buffer. The parameters of this network will not update in real time but will copy parameters from the actor network after a certain episode.
- **Critic:** The critic network is responsible for the iterative update of the value network parameter \(\theta_C\), uses the current state and selected action as input, and calculates the Q value \(Q(S, A, \theta_C)\).
- **Target Critic:** The target critic network is responsible for calculating the target Q value. The parameter of this network will not update in real time but will copy parameters from the critic network after a certain episode.

III. LEARNING ENVIRONMENT
The simulation of the DADPG algorithm is established by a bullet3 module. A simulation environment named the KukaDiversityObjectGraspEnv, which is created by Google.
Mind with the PyBullet toolbox, is introduced here as the basement of the DADPG simulation environment. The initial environment includes a 6 DOF Kuka robot, an adaptive workbench and a tray to place objects.

With respect to the original DDPG algorithm, there are 2 reasons for the low training speed: one is that in the training process, under complex conditions, the random exploration period of each training step will be extended; and the second is that the action reward is sparse, and the agent often needs to go through a long exploration period before it can obtain a positive reward. In order to fix these two negative influences and match the environment to the DADPG algorithm, the existing environment needs to be redeveloped. In this paper, the state and reward are redefined.

A. STATE
In RL problems, the state represents the total message in a real time environment, including all historical information of the current time slice. According to the Markov nature, only the current state is necessary if the agent is going to derive the future state. However, not all information plays a positive role in the agent training process, and some information even has side effects. This type of information makes the agent unable to train in the expected way. Therefore, the state data need to be filtered to reserve the information that can be observed by the agent as the current state. The main purpose of modifying the status definition here is to shorten the exploration time at the beginning and let the agent have a higher convergence speed.

Since the comparison of image position and end effector position is required, the camera angle needs to be adjusted. In this paper, each grasping is completed at the same height, which is independent of the change of the z axis, so the top view of the z axis is chosen as the camera shooting direction, turning a 3-dimensional coordinate problem into a 2-dimensional coordinate problem. The new state used here includes camera images, end effector information and pre-exploration area messages. First the AE method in Section 2 is introduced to pretreat the input image and calculate the key points of the pre-exploration area. Then the image features and end effector features are mixed by a feature fusion network. Finally, the predicted information from the AE network is concatenated with the mixed features as a new state. This new state is regarded as the input of later training.

The redefined state has several advantages in the learning process. First, since most of the background noise has been separated before training, the load of the network decreases. Second, when changing the grasping target, there is no need to prepare a large number of new training samples for object detection. The AE network only needs to change the $I_{support}$ as a new convolution kernel, and then the network can gradually approach the characteristics of new goals while learning the grasping strategy, which considerably decreases the training time. Third, the pre-exploration produced by the AE method narrows the scope of agent exploration in the early training stage efficiently, which leads to a higher convergence speed. Finally, adding the position data of the end effector as part of the input state can force the AC network to lean the correlation between the robot state and target object state. That is, the agent can make corrections based on the image and corresponding state of the end effector.

B. REWARD
The reward represents the action quality, and the final purpose of the RL agent is to maximize the accumulated rewards. In the original Kuka simulation, the reward is defined as follows:

$$ r = \begin{cases} 1, & \text{if grasping succeed}, \\ 0, & \text{if grasping failed} \end{cases} \quad (7) $$
In particular, at the end of each action, if the target rises a certain distance along the \( z \) axis of the world coordinate system, the grasping action is considered successful, and the reward of the current action equals 1; conversely, the reward equals 0. Due to the time spent exploring in the early training stage, the probability of a sparse reward matrix is high. In order to solve the problem caused by a sparse reward matrix, Andrychowicz \textit{et al.} proposed a module called hindsight experience replay (HER), which can improve the learning efficiency by storing the “failed” action in a designed replay buffer [22]. On the basis of HER, Colas \textit{et al.} introduced an algorithm called CURIOUS, and proposed the replacement method for multiple tasks and multiple objects in training [23]. CURIOUS, which was an extension of HER, used a learning process to measure the priority of tasks. These methods are all based on the replay buffer method. In this paper, the agent uses stratified reward function to make some improvements.

The first consideration in the grasping task is how to let the robot get sufficiently close to the target position, which means a “reach” task. In this way, a stratified reward function aiming at forcing the agent to learn as much as possible from actions is defined. The reward function is shown as follows:

- Whether the grasping task is completed judged. If it is, the feedback will obtain a positive value.

\[
r_1 = \begin{cases} 
2, & \text{if grasping succeed} \\
0, & \text{if grasping failed}
\end{cases}
\]  

(8)

- Whether the agent exploration occurs in the pre-exploration area judged. If the absolute distance between the robot end effector and the center of the pre-exploration area is smaller than the radius of the pre-exploration area, which is defined in Section 2.4, feedback will obtain the product of the absolute distance and a penalty coefficient; conversely, feedback will obtain the product of the absolute distance and a comparatively large penalty coefficient.

\[
r_2 = \begin{cases} 
-\frac{1}{100}d, & \text{if is in predicted area} \\
-\frac{1}{200}d, & \text{if is not in predicted area}
\end{cases}
\]  

(9)

where \( d \) is the distance between the chosen points.

\[
d = (x_g - x_p)^2 + (y_g - y_p)^2
\]  

(10)

- The absolute distance between the robot end effector and the actual position of the target represents the exploration accuracy, and the product of this absolute distance and a certain penalty coefficient is deemed part of the reward function.

\[
r_3 = -\frac{1}{200}[(x_g - x_r)^2 + (y_g - y_r)^2]
\]  

(11)

- The total reward equals the sum of the above 3 reward values.

\[
r = r_1 + r_2 + r_3
\]  

(12)

where \( x_g \) is the \( x \) axis coordinate of the robot end effector after action is executed, \( x_p \) is the \( x \) axis coordinate of the center point in the pre-exploration area, and \( x_r \) is the \( x \) axis coordinate of the center point in the real object area. The \( y \) value is the same.

The advantage of this stratified reward is that the agent can evaluate the value of each action more quantitatively during the exploration process which compensates for the shortcomings of not using the HER module to a certain extent. Additionally, the update of the AE network parameters can be fed back according to the action reward, which will improve the training efficiency because the feedback forces the coordinates of the predicted exploration center to be closer to the actual coordinates.
C. LEARNING PROCESS

At the beginning of training, each variable in the environment is initialized, and the target object is randomly placed in the grasping operation area. After the camera takes the real-time image, first, the camera image \( I_{\text{query}} \) and target object image \( I_{\text{support}} \) are sent to the AE network to generate the pre-exploration information; then the feature fusion network is used to merge the features of the camera image, robot end effector data and pre-exploration data are regarded as the current state \( s_t \). If the number of steps is less than the threshold, the agent will choose an action from the action space of the environment; conversely, the action will be selected by the actor network. When an action is chosen, the agent executes the action with a step function from a self-defined Kuka environment. The current reward \( r_t \) and robot state under the next time slice \( s_{t+1} \) are generated by the environment, and then all data \( (s_t, a_t, r_t, s_{t+1}) \) are stored in a replay buffer. When the replay buffer has enough samples, \( s_t \) is sent into the critic network to calculate \( Q \) while \( s_{t+1} \) is sent into the target critic network at the same time to calculate \( \hat{Q} \). Then, the \( \text{MSE} \) between \( Q \) and \( \hat{Q} \) is calculated, and the parameters of the critic network will be updated in the direction of the gradient of this \( \text{MSE} \). The environment will be reset when the agent finishes the grasping task or exceeds the allowed number of grasping attempts. The entire learning process is shown in Table 1.

IV. EXPERIMENTS

In this paper, end to end training is conducted on GTX1660s. The training network is set up using PyTorch. For each training, a training group with 64 samples is selected from the memory unit. The maximum number of samples that can be accommodated is 100000, and the learning rate is 0.0001. In order to fully verify the effectiveness of the DADPG, different reinforcement learning algorithms are used here to conduct comparative experiments with the same task target.

A. TASK TARGET

1) REACH

In the reach task, the robot end effector only needs to reach a preset point. In this type of task, the target point position is determined by the world coordinates of a random object. The agent attempts to fetch the object in the range of the search area with the actions selected by the actor network. The training goal is to enable the agent to reach the target point position within a certain number of steps. Figure 8 shows the process of a reach task in the Kuka environment.

2) PICK

The pick mission is deemed successful when the target object rises along the \( z \) axis of the world coordinate. In this type of task, the target object is placed in a tray, and the agent selects actions according to the actor network. The first step is letting the agent finish the “reach” task in the workspace and then execute the grasping and rising actions. The target of training is to create an agent that can finish the “pick” mission in a certain number of steps. Figure 9 shows the process of a pick task in the Kuka environment.

B. COMPARISON OF INPUT STATE

1) NON IMAGE STATE

This part of the experiment is introduced here to verify the influence of different one-dimensional state quantities on agent training. In Open AI Gym, the agent is required to produce a \( 21 \times 1 \) tensor as its observation. This tensor includes the object information of position, speed, Euler angle etc. In this paper, adjustments were made to the original observations in order to reduce the total amount of information processed by the network, and only the group positions of the end effector and object were collected here. The controlled variable method is used to conduct comparative experiments to verify the effect of this adjustment. A Kuka robot agent is trained 6000 times under the same PyBullet environment with the same DDPG structure, and the training target is the “pick” mission with a randomly placed object. Figure 10 demonstrates the results of the non image data comparison experiment.

Curves of the input state with all information, state without coordinate information and state only with coordinate information all experienced extreme fluctuations in the early training stage. This occurs because the agent did not acquire any information about how to accomplish tasks, thus, exploration in the working area was necessary at the beginning of training. Then, as the number of training steps increased, the curve gradually tended to be stable. With regard to the performance on success rate, the state only with coordinates performed best. The grasping success rate at 6000 steps reached approximately 0.617, followed by the state with all information reached 0.397. However, the curve of the input state without coordinate information finally reached 0.078, which is the lowest among all the results. Therefore, the evident positive input state in this work is coordinate information while other input does not...
have obvious positive significance for convergence, and some factors may be negative, which means that the adjustment of the input state is meaningful.

2) IMAGE STATE
In the Google Mind experiment, 472 × 472 images were used as the input states. In order to verify the influences of different image sizes, comparison experiments were proposed here; and three kinds of image sizes, 720 × 720, 472 × 472 and 48 × 48 acted as the input state. The experimental environment is the same as that of the non image state comparison experiment, and a feature extraction network is set to generate features from images. Table 2 shows the time costs of a single training step for different input image sizes. Figure 11 shows the training results of image state comparison experiment.

From Figure 11, it is clear that with the 720 × 720 and 472 × 472 images improved the success rate faster, both reaching 0.27 in the early training stage. The watershed appeared

![Image 1](image1.png)

**FIGURE 10.** Comparison of non image state training results.

![Image 2](image2.png)

**FIGURE 11.** Comparison of image state training results.

| Image Resolution | 720 × 720 | 472 × 472 | 48 × 48 |
|------------------|----------|----------|--------|
| Times of single step | 0.69113 | 0.49873 | 0.41874 |

**TABLE 2.** Training time of different image inputs.
after 500 training steps. The $720 \times 720$ image experienced severe fluctuations, and the success rate curve showed an overall downward trend between 500 and 1000 training steps. Considering the entire process, the curve experienced an overall upward trend and peaked at approximately 0.487 in the end. However, $472 \times 472$ image had more stable performance during the entire training process; and the final result was better, peaking at 0.623 at approximately 6000 steps. The $48 \times 48$ image showed an unstable performance in the early training stage, but became robust after several steps, and the success rate remained stable at approximately 0.611. The results demonstrates that the input image with a high resolution had too much noise that deeply affected the ability of the agent to extract useful information from the input data, which led to a lower final score. In addition, a higher resolution means a longer training time, which is shown in Table 2. The $48 \times 48$ image lagged behind in convergence speed since there was less information that could be collected in the initial training stage. In contrast, the $472 \times 472$ image acted well during the entire training stage, and was only slightly inferior to the $48 \times 48$ image in training time. Therefore, $472 \times 472$ images were selected as the input for the following experiments.

C. COMPARISON WITH OTHER NETWORK STRUCTURE

In this part, the DADPG algorithm is compared with different RL methods, including HER-DDPG, PPO and DADPG with a partial improvement, in a certain mission. The HER-DDPG algorithm introduces a hindsight experience replay module, which considers the impact of each failed action on the final result. This improvement significantly increases the training efficiency. PPO is a commonly used RL method treated as one of the baseline Open AI algorithms. PPO is proposed to address the problem of how to use existing data to make the greatest improvement in strategy without causing accidental training collapse, which is the same as the target of TRPO. However, PPO is more efficient and has a wider trial range than TRPO by using a first-order approach to solve the problem [9]. Furthermore, in order to verify the effects of the improvements proposed in this paper, the DDPG with AE methods and the DDPG with a stratified reward function were tested separately. These algorithms are introduced here since they all include AC network baselines, so the fairness of comparative experiments can be guaranteed to a great extent. Additionally all agents are trained with the same basic parameters under the same PyBullet environment for a fair comparison. Table 3 provides the training details. In order to verify the effectiveness of DADPG algorithms, the Kuka grasping mission is introduced here. There is a randomly chosen object placed in the workspace. The task is deemed a success when the target object rises in a certain distance.

Figure 12 shows the training results of the comparison experiment for the DADPG, the partially improved DDPG and the original DDPG. The improvements here are the AE method and stratified reward. It is clear that the DADPG algorithm had a higher rate of increase in the success rate at the early training stage. The success rate reached 77.92% after 10000 training episodes, and the entire training process was more stable, reaching 90.83% at the end of training. While the DDPG with the stratified reward also possessed a stable training process, only small fluctuations occurred in the early training stage; furthermore, the rate of increase was much slower, reaching 62.18% at the 10000th episode, and the final result converged at 81.74% after approximately 40000 training steps. This occurs because even though the improvement in the reward function was efficient, the agent still needed to take a long time to conduct target area exploration, which reflected the slow convergence speed in the early training stage. The DDPG with the AE method achieved better performance on the final result than the DDPG with the stratified reward function, but the performance was not superior to that of the DADPG, which reached 85.42% at the end of training. Furthermore, the DDPG with the AE method had a more intense fluctuation before the 10000th training step and took more steps to make the result converge to a stable result. This occurred because the AE method was unable to generate accurate information in the early training stage, but it improved slowly along with the training process. With regard to DDPG,
TABLE 3. Training details for different RL methods.

| Method             | Image State | Reward | HER Module | AE method |
|--------------------|-------------|--------|------------|-----------|
| DDPG               | Non         | Sparse | No         | No        |
| HER-DDPG           | Non         | Sparse | Yes        | No        |
| DDPG with AE Method| 472 × 472   | Sparse | No         | Yes       |
| DDPG with Stratified Reward | 472 × 472 | Stratification | No | No |
| DADPG              | 472 × 472   | Stratification | No | Yes |

TABLE 4. Stability test.

| No. Groups | 1 | 2 | 3 | 4 | 5 |
|------------|---|---|---|---|---|
| No. Tests  | 50 | 50 | 50 | 50 | 50 |
| No. Successes | 42 | 44 | 42 | 41 | 44 |
| Success Rate | 0.84 | 0.88 | 0.84 | 0.82 | 0.88 |

it experienced a stable rate of increase in the success rate at the early training stage, which was superior to both of the partially improved DDPG. However, the success rate peaked at approximately 79.7% around the 37500th episode. For all, the DADPG achieved the best performance on convergence speed and stability, which proved that the combination of the AE method and stratified reward function was effective. In addition, the partially improved DDPG performed better than the original one on the convergence speed and the results of the grasping success rate. Therefore, these two partially improved methods can be both considered meaningful. Figure 13 shows the training result of the comparison experiment for the DADPG with the DDPG-HER and PPO. Similar to the comparison experiment for the DADPG with DDPG partial improvements, the DADPG here also had good performance on training speed and stability; furthermore the HER-DDPG had achieved a better convergence speed. The HER-DDPG agent reached its result success rate after approximately 34000 training steps, and the result was approximately 84.02%. However, the training process of PPO showed both a lower training result and a longer training process, reaching 80.81% after approximately 60000 training steps. Therefore, agents trained by the DADPG were clearly efficient.

Then 5 groups of stability experiments were performed to verify the performance of the DADPG model in a self defined Kuka environment [24]. In each group of tests, 50 targets were randomly selected, the maximum number of grasping attempts was 8, and the success conditions were the same as those in the previous experiments. The result are shown in Table 4. It is clear that the success rate of the stability test remained steady at over 80%, and the peak point of the test is 88%, which proves the stability of the training results generated by DADPG for grasping missions.

V. CONCLUSION

For robot control, the stability and accuracy of a method are always of great significance, furthermore, in practical applications, whether a task can be efficiently completed with limited support data is also an important indicator for evaluating the performance of agents. Therefore, DADPG based reinforcement learning, which has an adaptive exploration method, is proposed in this paper. The AE method is used here to train the strategy-selection ability to grasp the target object with little support data. Furthermore, a stratified reward function is defined to enable agent learning from each action as much as possible, and improve the inefficient exploration problem that the agent experienced at the early training stage. Training was conducted on a self defined bullet3 environment, and comparison experiment was proposed for different RL algorithms. The results show a grasping success rate of 90.83%, and verify the superiority of the algorithm in convergence speed. Then, 5 groups of stability experiments were conducted to test the robustness of the model generated by the DADPG, and the average grasping success rate reached 88%. These results show that the DADPG can overcome the poor stability and inefficient convergence that may occur during the training process.

REFERENCES

[1] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, “Groups of adjacent contour segments for object detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 1, pp. 36–51, Jan. 2008.
[2] J. Shotton, J. Winn, C. Rother, and A. Criminisi, “TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation,” in Proc. Eur. Conf. Comput. Vis., 2006, pp. 1–15.
[3] J. Won, Y. Park, and I. H. Suh, “Scattering for robotic grasping in cluttered environments,” in Proc. Academ. Conf. Korea Inst. Electron. Eng., 2018, pp. 920–923.
[4] M. R. Dogar and S. S. Srinivasa, “A planning framework for non-prehensile manipulation under clutter and uncertainty,” Auto. Robots, vol. 33, no. 3, pp. 217–236, Oct. 2012.
[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” 2013, arXiv:1312.5602.
[6] D. Silver, G. Lever, N. Heess, T. Degris, and M. Riedmiller, “Deterministic policy gradient algorithms,” in Proc. 31st Int. Conf. Mach. Learn., 2014, pp. 387–395.
[7] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” 2015, arXiv:1509.06461.
[8] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” in Proc. 32nd Int. Conf. Mach. Learn., 2015, pp. 1889–1897.
[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.
[10] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2016, pp. 3406–3413.
[11] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” Int. J. Robot. Res., vol. 37, nos. 4–5, pp. 421–436, 2017.
[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2015, arXiv:1509.02971.
XIANGXIN JI was born in Nanjing, Jiangsu, China, in 1997. He received the B.S. degree in vehicle engineering from the Yancheng Institute of Technology, in 2019. He is currently pursuing the M.S. degree in mechanical engineering with Shanghai University. From 2019 to 2021, he was a Research Assistant with the Shanghai Advanced Robot Laboratory, Shanghai University. He is the author of five articles and more than three inventions. His research interests include self-learned agent design, reinforcement learning algorithms in robot controlling applications, and robotic motion planning.

FENG XIONG was born in Wulumuqi, Xinjiang, China, in 1974. He received the B.S. and M.S. degrees in mechanical engineering from the Xinjiang Institution of Technology, in 1995 and 1998, respectively, and the Ph.D. degree in mechanical engineering from Chongqing University, in 2001. From 2003 to 2011, he was a Lecturer at the School of Mechatronics Engineering and Automation, Shanghai University, China. From 2005 to 2007, he was a Project Manager with the Science and Technology Commission of Shanghai Municipality. Since 2008, he has been the Director of the National Semiconductor Lighting Application System Engineering Technology Research Center. Since 2011, he has been an Associate Research Fellow with the School of Mechatronics Engineering and Automation, Shanghai University. He is the author of 48 articles and more than 30 inventions and formulated two items of “local standards for semiconductor lighting application engineering” in Shanghai. His research interests include the intelligent manufacturing and automation, the improvement of computing power of high-dimensional data, and abstract algebra.

Dr. Xiong was a recipient of the honorary title of “Advanced Individual in World Expo Science and Technology” awarded by the Ministry of Science and Technology, China, in 2011, and the Third Prize of the Shanghai Science and Technology Progress Award.

WEICHANG KONG was born in Shanghai, China, in 1990. He received the B.S. and Ph.D. degrees in control theory and control engineering from the School of Electronics and Information Engineering, Tongji University, Shanghai, in 2012 and 2020, respectively. He is currently working as a Senior Engineer with the Jiangsu Automation Research Institute, Shanghai Branch. His research interests include the scheduling and optimization of multi-line production systems and data value assessment for real industrial data.

DONGFEI WEI was born in Lianyungang, Jiangsu, China, in 1997. He received the B.S degree in mechanical engineering from the Huaihai Institute of Technology, in 2019. He is currently pursuing the M.S. degrees in mechanical engineering with Shanghai University. From 2019 to 2021, he was a Research Assistant with the Shanghai Advanced Robot Laboratory, Shanghai University. He is the author of three articles and more than five inventions. His research interests include lightweight computer vision and the development of robot embedded systems.

ZEYAN SHEN was born in Hangzhou, Zhejiang, China, in 1997. He received the B.S. degree in mechanical engineering from Wenzhou University, in 2019. He is currently pursuing the M.S. degree in mechanical engineering with Shanghai University. From 2019 to 2021, he was a Research Assistant with the Shanghai Advanced Robot Laboratory, Shanghai University. He is the author of one article and more than three inventions. His research interests include robot motion planning and robot path planning problem. Mr. Shen is owner of the Third Prize of Zhejiang Mechanical Design Competition, in 2018.