ITC 1/53
**Information Technology and Control**
**Vol. 53 / No. 1 / 2024**
**pp. 5-18**
**DOI 10.5755/j01.itc.53.1.34708**

**Learning Sliding Policy of Flat Multi-target Objects in Clutter Scenes**

Received 2023/07/25 | Accepted after revision 2023/10/12

**HOW TO CITE:** Wu, L., Wu, J., Li, Z., Chen, Y., Liu, Z. (2024). Learning Sliding Policy of Flat Multi-target Objects in Clutter Scenes. *Information Technology and Control*, 52(4), 5-18. https://doi.org/10.5755/j01.itc.53.1.34708

# Learning Sliding Policy of Flat Multi-target Objects in Clutter Scenes

**Liangdong Wu**
School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

**Jiaxi Wu**
Institute of Automation, Chinese Academy of Sciences, Beijing, China

**Zhengwei Li**
School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

**Yurou Chen**
Institute of Automation, Chinese Academy of Sciences, Beijing, China

**Zhiyong Liu**
School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China;
Institute of Automation, Chinese Academy of Sciences, Beijing, China;
Cloud Computing Center, Chinese Academy of Sciences, Dongguan, Guangdong, China

**Corresponding author:** zhiyong.liu@ia.ac.cn

In clutter scenes, one or several targets need to be obtained, which is hard for robot manipulation task. Especially, when the targets are flat objects like book, plates, due to limitation of common robot end-effectors, it will be more challenging. By employing pre-grasp operation like sliding, it becomes feasible to rearrange objects and shift the target towards table edge, enabling the robot to grasp it from a lateral perspective. In this paper, the proposed method transfers the task into a Parameterized Action Markov Decision Process to solve the problem, which is based on deep reinforcement learning. The mask images are taken as one of observations to the network for avoiding the impact of noise of original image. In order to improve data utilization, the policy network predicts the parameters for the sliding primitive of each object, which is weight-sharing, and then the Q-network selects the optimal execution target. Meanwhile, extra reward mechanism is adopted for improving the efficiency of task actions to cope with multiple targets. In addition, an adaptive policy scaling algorithm is proposed to improve the speed and adaptability of policy training. In both simulation and real system, our method achieves a higher task success rate and requires fewer actions to accomplish the flat multi-target sliding manipulation task within clutter scene, which verifies the effectiveness of ours.

KEYWORDS: Deep Learning in Manipulation, Reinforcement Learning, Robot Control, Intelligent system, sliding policy.

## 1. Introduction

Robot grasping constitutes a pivotal focal point within the domain of robotics research, with numerous algorithms having been devised to propel advancements in this field [12]. Conventionally, prior to the robot's initiation of a grasping operation, the targeted object is prepositioned in a condition conducive to grasping, thereby enabling the robot's gripper to grasp it from one or multiple directions. However, in the face of a flat object placed horizontally on a table, a direct grasp is most likely not possible. Typical objects are common books, dishes etc. If such objects are thin and wide, robots equipped with ordinary gripper cannot grasp them. At this time, a simple and direct idea is that the "pre-grasp" action like sliding or pushing is taken to shift flat object to the table edge [5]. Currently, some manual or learning-based methods have emerged for processing individual object [28]. However, the target surrounded by other objects is more common. It means that the target is likely to collide with other objects before being moved to the table edge, resulting in a failure of the pre-grasp operation.

The target can be graspable through non-prehensile actions rearranging the surrounded obstacles, such as pushing, sliding, etc. [31]. Current approaches to clutter scenarios revolve around partially or completely isolating objects from other obstacles, enabling top-down access to obtain small objects [30]. For flat objects, however, such manipulations are still insufficient. Shift the target towards the table's edge sufficiently so that it is exposed for side grasping, rather than solely to separate it from obstacles. Furthermore, it is imperative to ensure that objects are not inadvertently dropped from the table to prevent potential damage. Consequently, the task of acquiring flat objects within a cluttered scene becomes notably more demanding. In particular, if there are a large number of obstacles and targets, it will be harder to rearrange each object to make a target move to the table smoothly and keep each object from falling [18]. Moreover, multiple targets in cluttered scenarios also add complexity to the task. Because it is not only necessary to ensure the graspable-ability of a single target, but also to fully consider the overall action efficiency of the task. The manipulation policy is put forward higher requirements.

In this paper, we aim to push each target in the cluttered scene to the table's side with a small number of actions to make them graspable, which is the focus of our research and also to eliminate difficulty for subsequent grasping. We introduce a technique grounded in deep reinforcement learning to accomplish sliding manipulation within cluttered scenes. The operation task is redefined as a Parameterized Action Markov Decision Process (PAMDP) [17], where the object to be slid is denoted as a discrete action space. The sliding primitive composed of sliding distance and execution angle is a continuous action space. Hence, the whole action space is both discrete and continuous. For avoiding the impact of noise of original image, the mask images are taken as one of observations to the network. The policy network predicts the parameters for the sliding primitive of each individual object, which is weight-sharing, and then the Q-network selects the optimal execution target. Furthermore, with the aim of enhancing the efficiency of completing multi-target task, the policy needs to identify the execution of each action as a necessary step in the overall minimum number of actions as much as possible. Therefore, we design an additional reward mechanism to intensify the policy's performance. Meanwhile, to boost the speed and adaptability of policy training, an adaptive policy scaling algorithm is proposed. The trained agent generates a sequence of sliding actions to rearrange individual objects, clear obstructions for a collision-free path, and subsequently slides the target towards the table edge. This process repeats, ensuring that each target progressively reaches a graspable pose.

The effectiveness of our method has been verified through experiments conducted in both simulated and real-world systems, showcasing a superior task success rate and enhanced action efficiency. To sum up, this paper has three main contributions.

1. The sliding policy is learned to rearrange objects via PAMDP and reinforcement learning with weight-sharing network for making multiple target flat objects graspable in clutter scenes.
2. Incorporate the extra reward into the system for boosting action efficiency.
3. An adaptive policy scaling algorithm is proposed for flat multi-target sliding tasks in clutter.

## 2. Related Works

In the field of robot grasping, numerous studies have emerged [4, 15], enabling intelligent manipulation of unknown objects within unstructured environments to a certain degree. Nonetheless, in regular circumstances, robots often face challenges when attempting to directly grasp flat objects such as books and plates. Instead, they require the assistance of pre-grasp operations, such as pushing [7], sliding [9], and so on, to render these objects graspable.

A straightforward idea is to design professional tool to complete the operation task [27], when it is hard for flat objects to obtain via general gripper. A unique two-finger fixture gripper was designed via Babin et al. and trained a dedicated "shovel" policy for it, so as to achieve a flat desktop to scoop up objects and then grasp them [1]. Berlin Institute of Technology manufactures the RBO soft robotic hand using silicone gel and polyester fibers [20], and it is driven pneumatically.

Ordinary gripper can grasp flat objects to accomplish the grasping task by pushing, sliding and other pre-grasp operations [5]. Sarantopoulos et al. trained the robot to grasp from the side by identifying the gap between table top and steel plate [22]. King et al. adopted the idea of trajectory optimization to solve the pre-grasp problem and obtained the optimal solution through training policy [14]. Kappler et al. classified objects into categories, and made adaptive adjustments for objects to be executed based on their categories, then completed pre-grasp operations by sliding [13]. CBiRRT algorithm [9] and reinforcement learning method [24] are used to move flat object to grabable poses.

When the target object is in close proximity to surrounding obstacles, employing pre-grasp actions can expedite the acquisition of the target via robot, such as pushing [6]. Huang et al. moved the obstacle by pushing, so that the target object could be discovered by the camera and provide input state for the subsequent capture [11]. Baichuan et al. output the optimal pushing action sequences based on the predictive network to capture the target in a relatively efficient way [10]. Berscheid et al. introduced sliding into the algorithm framework of VPG [31] and applied it to cluttered environments with constraints around storage bins [2]. A hierarchical reinforcement learning approach was proposed by Sarantopoulos et al. [21] to incorporate pushing, sliding, and other pre-grasp actions within a continuous action space framework.

Currently, research on pre-grasp manipulation of flat objects is relatively scarce. Bingjie et al. mitigated task complexity to some extent by not positioning the plates completely flat, successfully accomplishing plate grasping in clutter [25].

## 3. Background

### 3.1. Reinforcement Learning (RL)

In general, we represent the Markov Decision Process (MDP) as a tuple $(S, A, p, r, \gamma)$, where $S$ is the set of states, $A$ is the set of actions, $p$ is the state transition probability function, $r$ is the reward function, and $\gamma \in [0,1]$ is the discount factor.

The main goal of reinforcement learning is to maximize the discounted reward sum $R_t = \sum_{i=t} \gamma^{i-t} r_i$, also known as the expected return maximization. The agent's objective is to train a policy $\pi_\theta$ with parameter $\theta$ that can perform an action with maximum reward. How to learn policy efficiently is the main research content of reinforcement learning, and many algorithms have been proposed, such as DDPG [16], A3C [19], PPO [23] and so on.

### 3.2. Markov Decision Process for Parameterized Action Space

In the field of reinforcement learning, action spaces are commonly categorized into discrete actions and continuous actions. Currently, various algorithms are designed to address different types of action spaces. For instance, The Deep Q-Network (DQN) algorithm is explicitly designed to address problems characterized by discrete action spaces, whereas DDPG is customized for situations involving continuous action spaces. On the other hand, PPO is a versatile approach that can effectively handle both discrete and continuous action spaces.

However, there is another kind of action space that is more special, that is, the parameterized action space that is both discrete and continuous. Simultaneously, in certain scenarios, each discrete action $k \in A_d$ ($A_d$ = {1, 2, ..., $K$}) can be represented by a set of continuous parameters denoted as $x_k$, where the total number of continuous parameters is denoted as $m_k$. The aggregate of all continuous parameters, denoted as $x_k$, collectively constitutes a continuous action

space referred to as $A_c$. Consequently, this type of action space can be expressed as $A = \bigcup_{k \in A_d} \{(k, x_k) | x_k \in A_c\}$, described as a hybrid discrete-continuous action space.

In the case of hybrid discrete-continuous action spaces, traditional reinforcement learning algorithms such as DQN, DDPG, and PPO face challenges as they either discretize or relax the continuous actions without considering the specific characteristics of PAMDP during their initial design. As a result, achieving satisfactory results becomes difficult. P-DQN (Parametrized Deep Q-Network) [29] was proposed to address this issue by adapting the DQN algorithm to handle mixed action spaces. However, this approach involves inputting all action parameters into the action value network, which contradicts its theoretical derivation. To address this limitation, MP-DQN (Multi-Pass Deep Q-Network) [3] was introduced as an improvement over P-DQN. MP-DQN enhances the algorithm's performance by incorporating multiple inputs, thereby effectively handling hybrid discrete-continuous action spaces. By leveraging these multiple inputs, MP-DQN aims to overcome the limitations of previous approaches and achieve better results in scenarios involving mixed action spaces.

### 3.3. Sliding Primitive

This paper explores the use of deep reinforcement learning for end-to-end pre-grasp manipulations in cluttered scenes. When dealing with complex environments, agents typically require significant trial and error to acquire an appropriate policy. To expedite training, this study introduces a sliding primitive ($c_x$, $c_y$, $d$, $\theta$) that imposes constraints on the robot's action space. Here, $d$ represents the sliding distance, $\theta$ denotes the sliding direction, and ($c_x$, $c_y$) represents the central coordinate of the targeted object. Each object, identified by its central coordinate ($c_x$, $c_y$), is assigned a corresponding object number, and the robot employs a sliding operation ($d$, $\theta$) to move the object to a designated position. Alternatively, the sliding primitive can be expressed as ($k$, $d$, $\theta$), where $k$ represents the object number.

## 4. Method

### 4.1. Mask Images

The sliding policy requires gathering shape characteristics of both the object and the tabletop as observations to generate appropriate action instructions and subsequently execute the task. However, the direct use of the original image of the camera will be affected by noise, increasing the complexity, and considering the large number of samples required via reinforcement learning, it will further increase the difficulty of policy training. Therefore, the mask images are used as one of the state quantities in this paper. Given pre-processed mask images, the agent is capable of extracting shape features and relative positions of objects and tables. Subsequently, the agent can output sliding actions for each object to enable it graspable.

The mask images consist of three components: the object mask image $I_o$, the mask image of desktop $I_t$, and XOR image $I_x$, and $I_x$ represents objects located outside the desktop, which is expressed as:

$$I_x = AND(XOR(I_0, I_t), I_0) . \quad (1)$$

In the formula, AND denotes the logical and-operation, and XOR represents the exclusive xor operation. In the mask images $I_o$ and $I_t$, pixels corresponding to objects and desktops have a value of 1, while other areas have a pixel value of 0. In mask image $I_x$, the object is exposed to the outer part of the desktop with a pixel value of 1 and the rest of the area with a pixel value of 0. Figure 1 provides an illustration of two sets of distinct mask images ($I_o$, $I_t$, $I_x$). Notably, due to variations observed on the right side of the desktop, the desktop mask image $I_t$ solely displays the lower right area.

**Figure 1**
Binary mask images ($I_o$, $I_t$, $I_x$)

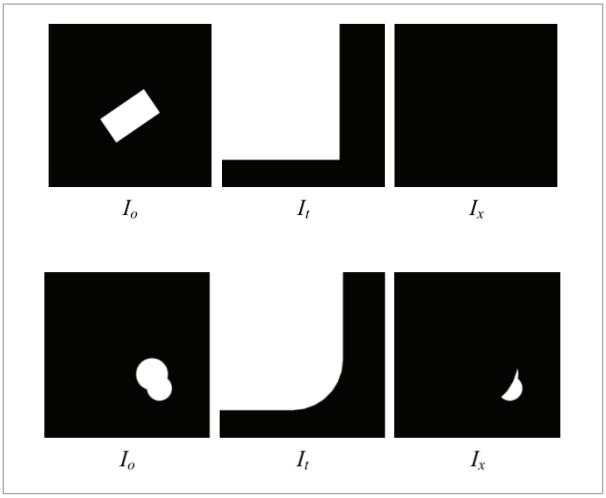
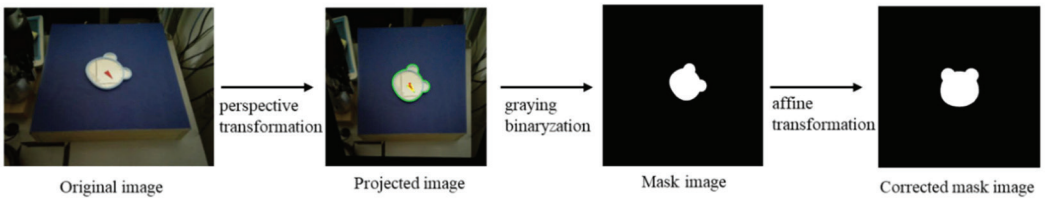


**Figure 2**
The generation of mask images in real world




In real world, considering that the robot will maintain contact with the object during the sliding operation, the robot may block the object. In this case, we cannot get the full object mask image directly from the current frame $I_p$. To circumvent this issue, before the beginning of each round, the RGB images of objects and desktops without occlusion are collected in advance and converted into mask images $I_0^-$ and $I_t^-$ with OpenCV library. The specific generation process is shown in Figure 2. Since the camera is positioned in the upper right region of the table, the camera's shooting angle is not perpendicular to the desktop, so the original image captured by the camera needs to be first transformed by perspective and projected on a plane parallel to the desktop. After that, we carry out grayscale, binarization and morphological processing of the projected image to extract the contour information and draw mask image of the object. Finally, the mask image of the object is adjusted to the center of the image by two-dimensional affine transformation, and the mask image of the object is obtained $I_0^-$. The desktop mask image $I_t^-$ is generated in a similar way.

## 4.2. Sliding Policy

At the initial phase of this research, we first carry out the design of policy network and value network for the case of only one object in environment. The status of the policy network $s$ includes three mask images $(I_o, I_t, I_x)$, the height of the object and the desktop, the pose and speed of the end-effector. Action $a$ is the linear velocity of the end effector along the *x/y/z* axis. Meanwhile, in order to speed up the convergence of the network, the policy network outputs not only action $a$, but also seven other variables as auxiliary output $y_{aux}$. The seven auxiliary output variables are the coordinates of the object and the desktop in the direction of *x/y*, their relative positions, and a signal (1/-1) about whether an object can be graspable. The loss function of the policy network $\pi_\varphi$ is:

$$L_\pi = -Q_\omega(s, \pi_\phi(s)) + (y_{aux} - \hat{y}_{aux})^2, \quad (1)$$

where $\hat{y}_{aux}$ is the true value of the auxiliary output variable $y_{aux}$.

The value network $Q_\omega$ bears resemblance to the policy network $\pi_\varphi$; however, it solely entails the action value $Q$ and lacks any auxiliary output. Simultaneously, the value network $Q_\omega$ possesses a more comprehensive range of input states compared to the policy network. Besides the input of the policy network, it also inputs the object's pose and the desktop and the signal of whether the object can be grasped (1/-1). These additional input states can assist the value network better evaluate actions $a$ and guide the policy network $\pi_\varphi$ to update the weights. The loss function of the value network $Q_\omega$ is:

$$L_Q = (r + \gamma \min_{i=1,2} Q_{\omega_i'}(s', \pi_\phi(s')) - Q_\omega(s, a))^2, \quad (2)$$

where, the two action values $Q$ in the TD3 algorithm are distinguished by the footmark $i$, and $s'$ is the state of the next moment.

However, when confronted with a more intricate cluttered scene involving multiple targets and a higher number of objects, training a policy network for each individual object becomes challenging. Convergence becomes difficult for each network, resulting in a lower task success rate. For all objects, given the similarity of operational parameters, denoted as $(d, \theta)$, utilizing a weighted sharing policy network $\pi_\varphi(s)$ can significantly enhance the efficiency of data utilization. To facilitate differentiation between objects, the system incorporates the observation $o^k$ as one of the states, and $k$ is a specific object. Consequently, the pol-

icy $\pi_\varphi(s,o^k)$ is employed to predict the action parameters for the execution of object $k$. An additional value network $Q_\omega(s,o^k,x^k)$ is employed to perform the object selection process: $k^* \leftarrow \arg\max Q(s,o^k,\pi(s,o^k))$. The action space based on sliding primitives comprises discrete object numbers denoted as $k$ and continuous action parameters $x_k:(d,\theta)$, resulting in a mixed action space $(k,x_k)$ that combines both discrete and continuous elements. In this paper, how to learn the sliding policy in cluttered scenarios is translated into a PAMDP problem.

Currently, the Bellman equation $Q(s,a)=\underset{r,s'}{E}[r+\gamma\max_{a'}Q(s',a')|s,a]$ can be rewritten as:

$$Q(s,o^k,x^k)=\underset{r,s'}{E}[r+\gamma\max_{k'}Q(s',o^{k'},\pi(s',o^{k'}))|s,o^k,x^k], \tag{3}$$

where $k'$ designates the object that is eligible for sliding in the subsequent state, $s'$ represents the state of the next time step, and $o^{k'}$ corresponds to the observation of object $k'$. In the equation, the maximum target variable is related to $k'$, and the observation $o^k$ is incorporated as an input variable to discern between distinct objects.

To determine the optimal network parameter $\varphi$ is the goal of the policy network $\pi_\varphi(s)$, which maximizes the action value $Q$, while keeping the parameter $\omega$ of the value network $Q_\omega$ constant.

$$\varphi \leftarrow \arg\max Q(s,o^k,\pi_\varphi(s,o^k))|\omega). \tag{4}$$

Therefore, the loss function for the policy network $\pi_\varphi(s)$ can be articulated as follows:

$$L_\pi=-Q_\omega(s,o^k,\pi_\varphi(s,o^k)). \tag{5}$$

The parameter $\omega$ is optimized by gradient descent, and the value function $Q_\omega$ loss function is represented by minimum mean square error:

$$\omega \leftarrow \arg\min_{\omega}(y_t-Q_\omega(s,o^k,x^k))^2, \tag{6}$$

where $y_t=r+\gamma\max_{k'}Q_{\omega'}(s',o^{k'},\pi_{\varphi'}(s',o^{k'}))$. $y_t$ can be used for estimation by only one step, or alternatively, N steps can be utilized for estimation to further enhance the model's performance.

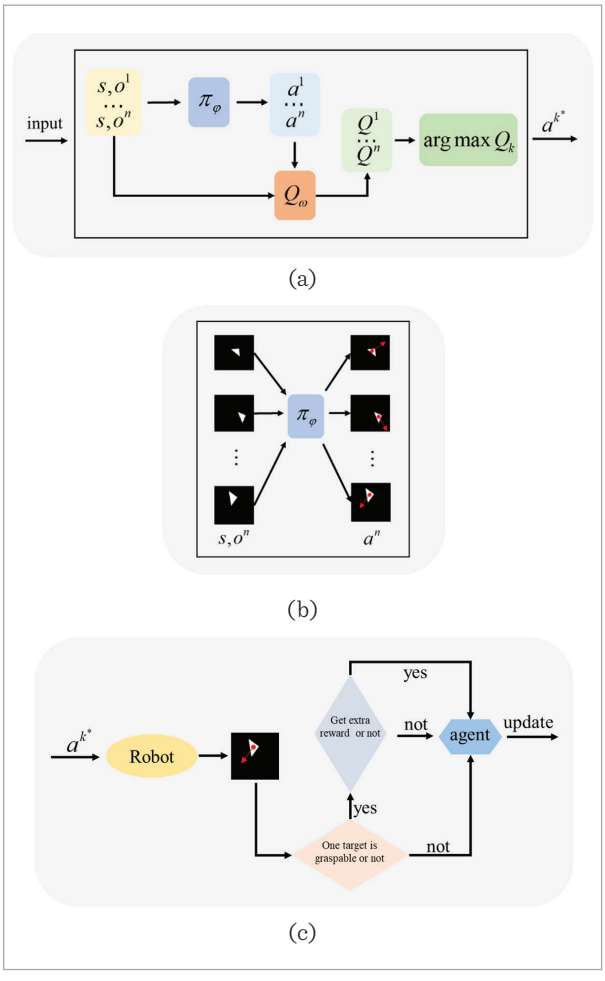
Subsequently, the loss function of the value network $Q_\omega$ can be formulated as follows:

$$L_Q=(y_t-Q_\omega(s,o^k,x^k))^2. \tag{7}$$

To obtain the sliding parameter $x^k$ and its associated action value $Q_K$, it is necessary to perform a forward operation on each object. Subsequently, the object $k^*$ is selected from the set of $n$ objects to execute the action $x_{k^*}$, which possesses highest action value. As a result, in an environment containing $n$ objects, the prediction of sliding parameters $x^k$ and action values $Q_K$ requires $n$ forward operations. An overview of ours is depicted in Figure 3.

**Figure 3**

The overview of our method. (a) The basic operating logic of policy network $\pi_\varphi$ and Q-network $Q_\omega$ in the system. (b) The policy network $\pi_\varphi$ outputs corresponding actions for each object, and the input-output logic of value network $Q_\omega$ is similar. (c) Two types of reward assist the agent in updating



For the reward function, we use a combination of -1/0 sparse reward and extra reward. The setting of sparse reward is expressed as:

$$r = \begin{cases} 0, & situation1 \\ -1, & others \end{cases}. \quad (8)$$

Situation 1 pertains to a scenario where the target object satisfies the following two conditions: (1) The target object possesses ample surface area to be stably exposed off the table; (2) The target object lies within the reachable workspace of the robot. Meanwhile, all objects are positioned on the table.

### 4.3. Extra Reward

In a clutter scene with multiple flat targets, it is not enough to just shift one target to render it graspable. Moreover, after each action is executed, the pose of each object may change due to touch, so ensuring the overall efficiency of the actions is the key for policy to complete the task. This paper adds extra reward, the main purpose of which is to enable the policy to learn associated actions (for example, sliding target A can move target B to the table together), thus optimizing task efficiency.

The system records the positions of targets at the beginning of episode $p_i$ ( $i \in$ number of targets).When the robot slides a target $T_1$ to the table's edge, $T_1$ is graspable and reach the goal position ($T_1$ reach goal, abbreviating $T_{1g}$). The $d_1$ is the distance between $p_i$ and $T_{1g}$. The $d_2$ is the distance between $p_i'$ (when $T_1$ reach goal) and $T_{1g}$. If $d_2 < d_1$, obtain an extra reward $r_e$ :

$$r_e = \frac{n_t}{n_0}, \quad (9)$$

where, $n_0$ is the quantity of other targets (excluding $T_1$), and $n_t$ represents the quantity of targets that satisfy the requirement $d_2 < d_1$. The agent will learn beneficial associated actions and improve the performance of system via extra reward.

### 4.4. Adaptive Policy Scaling

The scarcity of rewards poses a challenge for reinforcement learning in obtaining reward signals and acquiring an appropriate policy. To address this, the initial state is initialized to the state observed during the demonstration track, which expedites the training process and facilitates the acquisition of an improved policy. However, over-reliance on the presentation trajectory can lead to inadequate exploration of the entire state space. Therefore, this paper not only initializes the initial state of the environment as a demonstration trajectory, but also randomly initializes it to ensure the agent's ability to explore other state spaces. Specifically, the initialization is based on two conditions: In situations where the agent lacks knowledge on how to accomplish the task, the environment's initial state is initialized to the state observed during the demonstration trajectory. This process directly sets up the environment in a state that is near the completion of the task, and the policy learns these relatively simple subtasks first, thereby facilitating the learning of the overall task. After the agent is capable of partially completing the task, the initial state is randomly initialized directly, and the agent begins to learn the overall task and explore the state space not covered by the presentation trajectory.

In a multi-target environment with additional objects, if the policy network generates all actions simultaneously and the number of output actions varies, the policy model becomes unsuitable for continued use. The inconsistency in the number of output actions can lead to compatibility issues, rendering the policy model ineffective for handling the dynamic nature of the environment. In the proposed method, the policy network also takes the observation $o^k$ as an input, while $k$ is a specific object, and only the sliding parameter $x_k$ of a single object $k$ is output each time. It means that this method is not constrained by the number of objects.

Meanwhile, we draw upon the concept of course learning [8] and further extends the proposed method to the chaotic scene containing more objects, so as to incrementally raise the quantity of objects in the environment and realize the gradual learning of sliding operations, rather than letting the agent learn in the scene of many objects at the beginning, which can better help the agent to overcome exploratory problem. At the end of the course, even in a complex scene, the agent can separate the target objects and make them graspable orderly. The algorithm is shown as follow.

**Algorithm1.** Adaptive Policy Scaling

1: Initialize the policy network $\pi_\varphi$ and value network $Q_\omega$

2: Initialize parameters $\mu, \sigma, \alpha, T$

3: $\mu \leftarrow 2, \sigma \leftarrow \alpha \times (n-1), T \in \mu$

4: Initialize the train buffer $B_t$ and demo buffer $B_d$, generate $n$ reference tracks and save them in $B_d$

5: **for** epi=1, $M$ **do**

6: The environment is initialized to a cluttered scene of $n_i$ objects

7: if $n_i$<$M_0$ and $p$<$p_d$ then

8: State initialization of demonstrates trajectory

9: end if

10: if $n_i$<$M_0$ and $p \geq p_d$ then

11: Random initialization

12: end if

13: Rollout and save data to the buffer $B_t$

14: Update the success rate $\zeta$ with $\mu$ objects

15: Update parameters $\mu, T$ after $m$ episodes

16: $\mu' \leftarrow \mu + 1(\zeta > \beta_h) - 1(\zeta < \beta_l)$

17: if $\mu' > \mu$ then

18: $T \leftarrow T + 1$

19: else

20: $T \leftarrow T - 1$

21: end if

22: $\mu \leftarrow \mu'$

23: Update the network parameters $\varphi, \omega$ by Equationstab (4) and (6)

24: if $\mu \geq n$ then

25: Update $\delta, \varepsilon$, $\delta \leftarrow \delta \times \xi, \varepsilon \leftarrow \varepsilon \times \xi$

26: end if

27: **end for**

where, $p_d = clip(1 - \frac{\zeta}{\alpha}, \varepsilon_1, \varepsilon_2)$ represents the probability of initialization to a state on the demonstration trajectory, and $p_r = 1 - p_d$ represents the probability of random initialization, the sum of which is 1. $\zeta$ is the current task success rate, and $\varepsilon_1$ and $\varepsilon_2$ are the minimum and maximum values of $p_d$. $\alpha \in (0,1]$ is a hyperparameter adjusting the size of $p_d$. When the success rate $\varsigma \geq \alpha$, the environment is initialized to the state of the presentation trajectory with only a small probability $\varepsilon_1$; otherwise, it is initialized randomly. Initial stage of training is accelerated in this way, where $M_0$ is significantly less than $M$.

To facilitate better transfer of the sliding policy trained in an environment with a limited number of objects to a scenario with a higher number of objects, this paper adopts truncated normal distribution sampling $N_\tau(\mu, \sigma, l, h)$ for ascertaining the quantity of objects in the current environment $n_i$, where $l$ and h are truncated boundaries. $\alpha$ is a hyperparameter that adjusts the size of the standard deviation $\sigma$, and the integer down function adjusts the number of objects $n_i$ to an integer.

At the beginning of the training, there are only 2 objects in the environment, including 1 target object and 1 obstacle object. When the success rate of the agent in this simple subtask exceeds $\beta_h$, we increase the object mean $\mu$ to improve the task difficulty. Conversely, when the agent's success rate in the ongoing subtask drops below a certain threshold $\beta_l$, the mean of objects initialized by the environment $\mu$ is reduced. This measure is taken to decrease the task difficulty and enhance the agent's chances of achieving successful outcomes in subsequent attempts. Meanwhile, the number of target objects $T$ also changes accordingly. The process is repeated until the object mean $\mu$ reaches the preset maximum number of objects. In line 16 of the algorithm, 1(g) is the symbolic function.

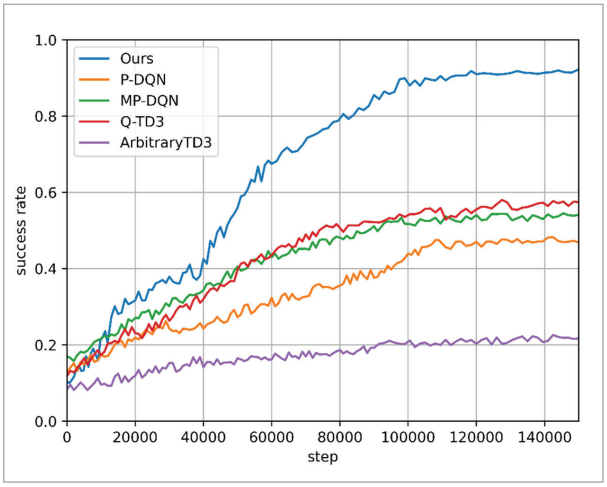
For avoiding the policy falling into local optimality, noise $\delta$ is added to the action of the agent, and the $\varepsilon$-greedy policy is utilized to choose the object $k$ to be acted on. In addition, the exploration noise $\delta$ and $\varepsilon$ are limited by decreasing the attenuation rate $\xi$ to assist the policy network convergence.

## 5. Experiments

### 5.1. Experimental Setting

In this paper, the simulation is conducted using the Mujoco software [26]. The main body of the experiment is a six-axis UR3 robot equipped with Robotiq85 gripper. The task assigned to the agent involves sliding the objects within the cluttered scene while ensuring that none of the objects fall. Additionally, the agent must guide each target towards the table edge,

**Figure 4**
The learning curve of each method is obtained in a training environment consisting of 2 targets and 4 non-targets




ultimately achieving a graspable state for all the targets. In the simulation environment, $n$ objects, which includes the target objects (purple triangle), are randomly dispersed on the desk as depicted in Figure 5. For evaluation, we employ the average quantity of actions and task success rate in 50 times as the metrics.

## 5.2. Simulated Results

In order to illustrate the effectiveness of our approach, we conduct a comparative analysis against various baseline approaches.

**P-DQN:** The method presented in reference [29] generates all actions in a simultaneous manner, employing a Q-network to handle a mixed action space comprising both discrete and continuous elements.

**MP-DQN:** In order to mitigate false gradients caused by Q-values dependence, this method [3] takes the action base vector as the input of the Q network. This variant of P-DQN helps to address the issue effectively.

**Q-TD3:** To yield the precise policy gradient, the $Q_\omega$ network exclusively computes the loss function of $\pi_\phi$ network. Additionally, an auxiliary $Q_\psi$ network is utilized to choose the object to be acted, further contributing to the overall functionality of the method.

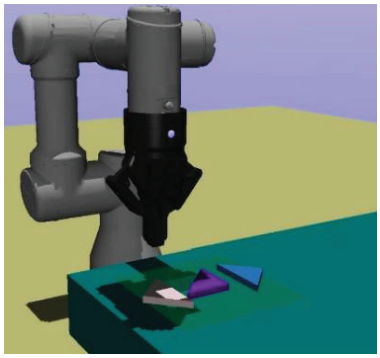
**ArbitraryTD3:** The workspace ($c_x$, $c_y$, $d$, $\theta$) is consistent with our approach, but can be slid from arbitrary point. A method following the standard TD3 framework.

The success rate of training is shown in Figure 4. The training environment is a clutter scene with 2 target objects (Tob) and 4 non-target objects (NTob). In addition, both random and challenging scenes are utilized to evaluate the methods, considering various numbers of objects, and the results are summarized in Table 1. Superior performance reflected in our method, achieving a higher success rate and requiring fewer actions compared to other methods, making it the optimal choice for the given tasks.

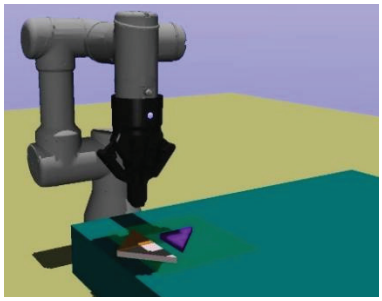
The network directly obtains the object observation $o^k$ in our method, and the network exclusively estimates $Q^k$ without computing all of the Q-values. This focused estimation allows both $\pi_\varphi$ network and the Q-network to acquire more precise Q-values, contributing to the overall precision and efficacy of ours. Indeed, our method differs from P-DQN and MP-DQN. MP-DQN introduces a multi-channel method that weakens the Q-values error estimation, while in ours, we focus on estimating $Q^k$ directly from the object observation, which results in a distinct and more accurate Q-values estimation process. Furthermore, our proposed method utilizes the Q-network $Q_\omega$ for action selection, allowing it to provide a policy gradient for the policy network $\pi_\varphi$. This close integration between action parameters and target selection leads to a more cohesive and effective way in handling the task. The incorporation of the extra reward further improves the sensitivity of policy to changes in the state of each target during training. This improvement allows the system to more effectively adapt to varying conditions and optimizes the overall system's execution efficiency. The large action space of ArbitraryTD3 exacerbates the challenge of exploration, making it difficult to learn a proper policy. In contrast, our method tackles this issue by rearranging the non-target objects to create separation between each target and make them graspable orderly through sliding.

To analyze the difference between the proposed method and the baselines, we conduct training in 3 randomly arranged scenarios, then testing in random ones and other 3 challenging scenarios. The objects in each type of scenario are 1 Tob and 2 NTob, 2 Tob and 4 NTob, and 3 Tob and 6 NTob, respectively, as shown in Figure 5, purple triangle(s) representing the target(s). The results are presented in Table 1, where r represents randomly distributed scene, c represents challenging scene, and 3, 6, 9 represents the number of objects. The success rate tends to decrease and the

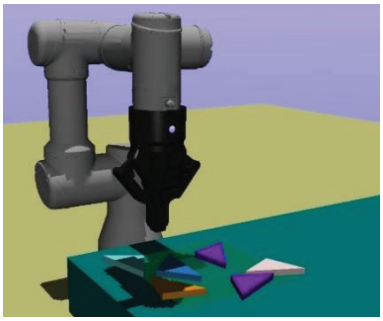
**Figure 5**
Random clutter scene (a, c, e) and challenging scene (b, d, f) under simulation



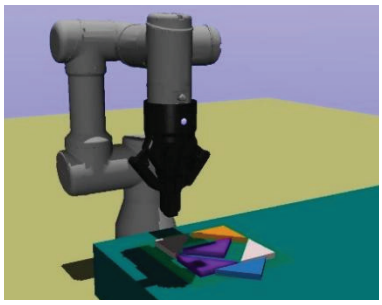
(a)



(b)



(c)



(d)

**Table 1**
The test results of proposed method are compared with other baselines in different clutter scenes

| Metric | Success Rate (%) | | | | | |
|---|---|---|---|---|---|---|
| Arrangement | Random | | | Challenging | | |
| | r3 | r6 | r9 | c3 | c6 | c9 |
| P-DQN | 91.2 | 46.8 | 21.3 | 89.7 | 39.5 | 16.5 |
| MP-DQN | 92.4 | 53.1 | 24.5 | 90.6 | 47.2 | 19.7 |
| Q-TD3 | 94.3 | 56.3 | 26.1 | 88.5 | 49.6 | 20.4 |
| ArbitraryTD3 | 68.9 | 20.6 | 5.3 | 57.3 | 11.2 | 3.9 |
| **Ours** | **98.8** | **91.2** | **80.4** | **94.8** | **88.3** | **77.8** |

(a)

| Metric | Number of Action | | | | | |
|---|---|---|---|---|---|---|
| Arrangement | Random | | | Challenging | | |
| | r3 | r6 | r9 | c3 | c6 | c9 |
| P-DQN | 4.2 | 7.8 | 12.4 | 4.6 | 8.7 | 13.2 |
| MP-DQN | 3.6 | 7.2 | 11.6 | 3.9 | 8.1 | 12.8 |
| Q-TD3 | 2.9 | 6.7 | 10.8 | 3.8 | 7.8 | 11.9 |
| ArbitraryTD3 | 4.7 | 8.9 | 13.5 | 5.4 | 9.7 | 15.1 |
| **Ours** | **2.1** | **4.3** | **6.8** | **2.5** | **4.9** | **7.3** |

(b)

average number of actions required increases, with the increasement in the quantity of objects. This trend indicates that handling a larger number of objects introduces greater complexity and challenges for the methods, resulting in decreased success rates and increased action requirements across the board. However, our method manages to maintain a decent performance, even in challenging scenarios with 9 objects (3 Tob in them) and achieve 77.8% success rate, and other metrics are optimal. Although other methods have a significant performance while the quantity of objects is few, the success rate of the task decreases visibly with the increase of the number of objects, which is due to the inaccurate estimation of Q-values and insufficient exploration of action space.

We conducted ablation experiments on the proposed algorithm 1 and the extra reward mechanism to verify their effectiveness for the whole system. There are 9 objects (including 3 Tob) in the simulation environment, and the results are shown in Table 2, w/o representing without and w/ representing with. When al-

**Table 2**
Ablation experiment results of algorithm1 and extra reward

| Metric | Success Rate (%) | | Number of Action | |
|---|---|---|---|---|
| Arrangement | r | c | r | c |
| Ours w/o alg1 w/o ER | 38.5 | 32.3 | 10.2 | 11.3 |
| Ours w/o alg1 w/ ER | 42.1 | 36.8 | 9.5 | 10.4 |
| Ours w/ alg1 w/o ER | 47.8 | 42.6 | 8.9 | 9.7 |
| **Ours w/ alg1 w/ ER** | **80.4** | **77.8** | **6.8** | **7.3** |

gorithm 1 and extra reward are not used neither, each data is the lowest, and when both are used, it is the best. In addition, the data also show that the experimental result of algorithm 1 utilized only is better than extra reward only. Comprehensive analysis shows that with the assistance of algorithm 1, the generalization ability of the whole system is improved, and the addition of extra reward can effectively compensate for the exploratory problems and enhance the performance of policy.

## 5.3. Results of Real System

In the real system, we use UR3 robot and Robotiq85 gripper for sliding manipulation, as shown in Figure 6, and adopt Kinect2.0 camera on the top to locate objects and generate image mask via Apriltag [3] and Section 4.1. Using mask image can not only accelerate the training of policy in simulation, but also better eliminate the error and noise caused by the original image in the policy transfer application of real system. To facilitate the execution of the sliding action, the rubber block is fixed at the end of the gripper, which also makes the contact process cushioned. We conducted a comparative evaluation between the proposed method and four other baselines, running a total of 50 times in both random and challenging scenes with 9 objects (including 3 randomly assigned targets). To evaluate the performance of the proposed method under real-world conditions, the policy trained in the simulation is directly transferred to the real system without any fine-tuning. The obtained results are shown in Table 3. The video is available at https://www.youtube.com/watch?v= LA05I6POuzA.

**Figure 6**
Real system (a) and experimental objects (b)

(a) (b)

**Table 3**
Experimental results in real system

| Metric | Success Rate (%) | Number of Action |
|---|---|---|
| P-DQN | 23.5 | 11.9 |
| MP-DQN | 28.6 | 10.8 |
| Q-TD3 | 29.7 | 9.7 |
| Ours | 82.3 | 5.6 |
| Ours (novel objects) | 76.5 | 6.3 |

The proposed method achieves the best results for sliding task in success rate and number of actions. Specifically, 82.3% is the result of success rate, and the number of actions required in the random scene is 5.6. It should be noted that there may be relative motion between the object and the gripper when it is in contact under simulation, but this relative motion can be ignored in the actual situation. In addition, we also conduct experiments on novel real objects, as shown in Figure 7, where seven objects are new

**Figure 7**
Novel 3D printed objects and living objects for experiments

(a) (b)

and the other two are existing triangular objects. The results demonstrate that even when encountering shapes that it has not been exposed to during training, the policy model can achieve 76.5% success rate in completing the sliding task. This also verifies the robustness and generalization of the proposed method.

## 6. Conclusion

In this paper, we transform the flat multi-target sliding manipulation task in clutter scenes into a parameterized action Markov decision process. Subsequently, we propose a method to address the issue, which is based on deep reinforcement learning. In this method, the mask images are taken as one of the states at the input side to avoid the noise effect of the original image. To improve data utilization, the parameters of objects' sliding primitives are predicted by the policy network, while the policy is weight-sharing, and then the Q-network selects the optimal execution object. Meanwhile, adding extra reward makes the policy better able to cope with multi-targets situation. In addition, an adaptive policy scaling algorithm is proposed to boost the speed and adaptability of policy training. In simulation and real system, the proposed method achieves the flat multi-targets sliding manipulation task with preferable performance, which verifies the effectiveness of ours.

This paper centers its attention on the pursuit of attainable graspability for individual targets within cluttered environments. It does not delve into the subsequent grasping, which is also our future research work. Two viable ideas emerge for consideration: the first entails employing hierarchical reinforcement learning to train a sub-policy for grasping, while the second is to use parallel training to train both sliding and grasping policies.

### Acknowledgements

This work was supported in part by the National Key Research and Development Plan of China under Grant 2020AAA0108902; in part by the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32050100; and in part by the Dongguan Core Technology Research Frontier Project, China, under Grant 2019622101001.

## References

1. Babin, V., Gosselin, C. Picking, Grasping, or Scooping Small Objects Lying on Flat Surfaces: A Design Approach. The International Journal of Robotics Research, 2018, 37(12), 1484-1499. https://doi.org/10.1177/0278364918802346
2. Berscheid, L., Meißner, P., Kröger, T. Robot Learning of Shifting Objects for Grasping in Cluttered Environments. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, 612-618. https://doi.org/10.1109/IROS40897.2019.8968042
3. Bester, C. J., James, S. D., Konidaris, G. D. Multi-Pass Q-Networks for Deep Reinforcement Learning with Parameterized Action Spaces. arXiv Preprint arXiv:1905.04388, 2019. https://doi.org/10.48550/arXiv.1905.04388
4. Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., Levine, S., Vanhoucke, V. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, 4243-4250. https://doi.org/10.1109/ICRA.2018.8460875
5. Chang, L. Y., Srinivasa, S. S., Pollard, N. S. Planning Pre-Grasp Manipulation for Transport Tasks. 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, 2697-2704. https://doi.org/10.1109/ROBOT.2010.5509651
6. Danielczuk, M., Angelova, A., Vanhoucke, V., Goldberg, K. X-ray: Mechanical Search for an Occluded Object by Minimizing Support of Learned Occupancy Distributions. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, 9577-9584. https://doi.org/10.1109/IROS45743.2020.9340984
7. Dogar, M. R., Srinivasa, S. S. Push-Grasping with Dexterous Hands: Mechanics and a Method. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, 2123-2130. https://doi.org/10.1109/IROS.2010.5652970
8. Gupta, J. K., Egorov, M., Kochenderfer, M. Cooperative Multi-Agent Control Using Deep Reinforcement Learning. Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Pau-

lo, Brazil, May 8-12, 2017, Revised Selected Papers 16. Springer International Publishing, 2017, 66-83. https://doi.org/10.1007/978-3-319-71682-4_5

9. Hang, K., Morgan, A. S., Dollar, A. M. Pre-Grasp Sliding Manipulation of Thin Objects Using Soft, Compliant, or Underactuated Hands. IEEE Robotics and Automation Letters, 2019, 4(2), 662-669. https://doi.org/10.1109/LRA.2019.2892591
10. Huang, B., Han, S. D., Yu, J., Boularias, A. Visual Foresight Trees for Object Retrieval from Clutter with Nonprehensile Rearrangement. IEEE Robotics and Automation Letters, 2021, 7(1), 231-238. https://doi.org/10.1109/LRA.2021.3123373
11. Huang, H., Dominguez-Kuhne, M., Ichnowski, J., Satish, V., Danielczuk, M., Sanders, K., Lee, A., Angelova, A., Vanhoucke, V., Goldberg, K. Mechanical Search on Shelves Using Lateral Access X-ray. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, 2045-2052. https://doi.org/10.1109/IROS51168.2021.9636629
12. Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., Levine, S. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. Conference on Robot Learning, PMLR, 2018, 651-673.
13. Kappler, D., Chang, L. Y., Pollard, N. S., Asfour, T., Dillmann, R. Templates for Pre-Grasp Sliding Interactions. Robotics and Autonomous Systems, 2012, 60(3), 411-423. https://doi.org/10.1016/j.robot.2011.07.015
14. King, J., Klingensmith, M., Dellin, C., Dogar, M., Velagapudi, P., Pollard, N., Srinivasa, S. Pregrasp Manipulation as Trajectory Optimization. Robotics: Science and Systems, 2013. https://doi.org/10.15607/RSS.2013.IX.015
15. 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. The International Journal of Robotics Research, 2018, 37(4-5), 421-436. https://doi.org/10.1177/0278364917710318
16. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. Continuous Control with Deep Reinforcement Learning. arXiv Preprint arXiv:1509.02971, 2015. https://doi.org/10.48550/arXiv.1509.02971
17. Masson, W., Ranchod, P., Konidaris, G. Reinforcement Learning with Parameterized Actions. Proceedings of the AAAI Conference on Artificial Intelligence, 2016, 30(1). https://doi.org/10.1609/aaai.v30i1.10226
18. Moll, M., Kavraki, L., Rosell, J. Randomized Physics-Based Motion Planning for Grasping in Cluttered and Uncertain Environments. IEEE Robotics and Automation Letters, 2017, 3(2), 712-719. https://doi.org/10.1109/LRA.2017.2783445
19. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. International Conference on Machine Learning, PMLR, 2016, 1928-1937.
20. Puhlmann, S., Harris, J., Brock, O. RBO Hand 3: A Platform for Soft Dexterous Manipulation. IEEE Transactions on Robotics, 2022, 38(6), 3434-3449. https://doi.org/10.1109/TRO.2022.3156806
21. Sarantopoulos, I., Kiatos, M., Doulgeri, Z., Malassiotis, S. Total Singulation with Modular Reinforcement Learning. IEEE Robotics and Automation Letters, 2021, 6(2), 4117-4124. https://doi.org/10.1109/LRA.2021.3062295
22. Sarantopoulos, I., Koveos, Y., Doulgeri, Z. Grasping Flat Objects by Exploiting Non-Convexity of the Object and Support Surface. 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, 5606-5611. https://doi.org/10.1109/ICRA.2018.8461192
23. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. Proximal Policy Optimization Algorithms. arXiv Preprint arXiv:1707.06347, 2017. https://doi.org/10.48550/arXiv.1707.06347
24. Sun, Z., Yuan, K., Hu, W., Yang, C., Li, Z. Learning Pregrasp Manipulation of Objects from Ungraspable Poses. 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, 9917-9923. https://doi.org/10.1109/ICRA40945.2020.9196982
25. Tang, B., Corsaro, M., Konidaris, G., Nikolaidis, S., Tellex, S. Learning Collaborative Pushing and Grasping Policies in Dense Clutter. 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, 6177-6184. https://doi.org/10.1109/ICRA48506.2021.9561828
26. Todorov, E., Erez, T., Tassa, Y. MuJoCo: A Physics Engine for Model-Based Control. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, 5026-5033. https://doi.org/10.1109/IROS.2012.6386109
27. Tong, Z., He, T., Kim, C. H., Ng, Y. H., Xu, Q., Seo, J. Picking Thin Objects by Tilt-and-Pivot Manipulation and Its Application to Bin Picking. 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, 9932-9938. https://doi.org/10.1109/ICRA40945.2020.9197493

28. Wu, J., Zhong, S., Li, Y. Learning Pre-Grasp Pushing Manipulation of Wide and Flat Objects Using Binary Masks. Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8-12, 2021, Proceedings, Part IV 28. Springer International Publishing, 2021, 366-377. https://doi.org/10.1007/978-3-030-92273-3_30

29. Xiong, J., Wang, Q., Yang, Z., Sun, P., Han, L., Zheng, Y., Fu, H., Zhang, T., Liu, J., Liu, H. Parametrized Deep Q-Networks Learning: Reinforcement Learning with Discrete-Continuous Hybrid Action Space. arXiv Preprint arXiv:1810.06394, 2018. https://doi.org/10.48550/arXiv.1810.06394

30. Xu, K., Yu, H., Lai, Q., Wang, Y., Xiong, R. Efficient Learning of Goal-Oriented Push-Grasping Synergy in Clutter. IEEE Robotics and Automation Letters, 2021, 6(4), 6337-6344. https://doi.org/10.1109/LRA.2021.3092640

31. Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., Funkhouser, T. Learning Synergies Between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, 4238-4245. https://doi.org/10.1109/IROS.2018.8593986