Robotics and Autonomous Systems 164 (2023) 104385

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Efficient reinforcement learning with least-squares soft Bellman residual for robotic grasping☆

Yixing Lan[1], Junkai Ren *,[1], Tao Tang, Xin Xu *, Yifei Shi, Zixin Tang

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China

ARTICLE INFO

Article history:
Available online 23 February 2023

Keywords:
Reinforcement learning
Policy evaluation
Robotic grasping
Sparse kernel

ABSTRACT

Grasping control of intelligent robots has to deal with the difficulties of model uncertainties and nonlinearities. In this paper, we propose the Kernel-based Least-Squares Soft Bellman residual Actor–Critic (KLSAC) algorithm for robotic grasping. In the proposed approach, a novel linear temporal-difference learning algorithm using the least-squares soft Bellman residual (LS^2BR) method is designed for policy evaluation. In addition, KLSAC adopts a sparse-kernel feature representation method based on approximate linear dependency (ALD) analysis to construct features for continuous state–action space. Compared with typical deep reinforcement learning algorithms, KLSAC has two main advantages: firstly, the critic module has the capacity for rapid convergence by computing the fixed point of the linear soft Bellman equation via the least-squares optimization method. Secondly, the kernel-based features construction approach only requires predefining the basic kernel function and can improve the generalization ability of KLSAC. The simulation studies on robotic grasping control were conducted in the V-REP simulator. The results demonstrate that compared with other typical RL algorithms (e.g., SAC and BMPO), the proposed KLSAC algorithm can achieve better performance in terms of sample efficiency and asymptotic convergence property. Furthermore, experimental results on a real UR5 robot validated that KLSAC performed well in the real world.

1. Introduction

Robotic grasping control is an important research topic in the field of robotics, which has shown great potential in practical applications [1], such as express sorting [2], bin picking [3], human–robot interaction [4], cloth manipulation [5] and many sorts of dexterous manipulation [6–8]. The fast development of intelligent manufacturing has increasing demands on robots with precise manipulation and competent services. Although model-based control methods can solve robotic grasping tasks based on accurate models of robots and prior knowledge of motion patterns, it is still difficult to model complex and dynamic grasping tasks in practice. In addition, the performance of model-based methods will be unsatisfactory when the model is inaccurate or dynamically changing.

Reinforcement learning (RL), as an important class of machine learning approaches for solving sequential optimization decision-making issues, has played a critical role in improving the autonomous learning-control performance of robotic systems. RL agents learn an optimized control policy by interacting with the environment without accurate models, which significantly reduces the development cost and enables robots to be more adaptive to uncertain complex environments. Recently, RL has been applied to robotic grasping tasks to improve the grasping performance in various scenes [9–16]. For example, Kalashnikov et al. [10] proposed the QT-Opt algorithm to learn closed-loop vision-based manipulation skills with deep Q network. However, a large number of training samples are required to learn a near-optimal policy. Yuan et al. [13] applied DQN to object rearranging manipulation tasks with visual feedback and increased the operational success rate. Although these methods showed good grasping performance in simulation environments, additional efforts are required to enable the robots to adapt to the real world with different environmental background, system dynamics, and external noises. As discussed in [17], the training stability and generalization capability of current RL-based object grasping methods still need to be improved. As is well known, many existing deep reinforcement learning (DRL) algorithms require substantial interaction data from the environments to obtain good performance and still face challenges such as low sample efficiency, slow convergence speeds, and unstable training processes. Moreover, as collecting training data on physical robot systems is costly, time-consuming, and potentially unsafe, RL algorithms are usually trained in simulators and often may fail on

☆ This work was supported by the National Natural Science Foundation of China under Grants 61825305, 62002379.

* Corresponding authors.
E-mail addresses: jk.ren@nudt.edu.cn (J. Ren), xinxu@nudt.edu.cn (X. Xu).

[1] These authors have contributed equally to this work.

https://doi.org/10.1016/j.robot.2023.104385
0921-8890/

real robots due to the gap between the simulation environment and the real world [18].

In order to solve the above problems, we propose a novel actor–critic algorithm (called KLSAC) based on sparse-kernel representation and a least-squares soft Bellman residual method to learn optimized policies for robotic grasping. Firstly, a least-squares soft Bellman residual method (LS^2BR) is designed to evaluate grasping policies, effectively improving the learning efficiency and stability of the actor-critic algorithm. Secondly, to improve the transfer ability from simulation to real-world environments, KLSAC adopts automatic feature representation of continuous state-action pairs by performing an approximate linear dependency (ALD) analysis, which can provide sparse kernel representations and improve the generalization ability of its solutions. The experimental results illustrate that KLSAC can learn robust grasping control policies efficiently without an accurate model. The tests on the real UR5 robot verified that the proposed algorithm can generalize well from simulation to real-world applications. The contributions of this work are multi-fold and can be summarized as:

- A novel actor–critic algorithm based on least-squares soft Bellman residual is proposed for solving continuous learning-control problems. Some theoretical analysis on its performance is provided.
- A sparse-kernel representation method is designed for automatic feature selection, which improves the accuracy of policy evaluation and the generalization ability of KLSAC.
- Various simulations on robotic grasping tasks were conducted in V-REP simulator, which verified that the learning efficiency and asymptotic convergence of KLSAC are better than typical DRL algorithms. Extensive sim-to-real tests of KLSAC were performed on a 6-DOF robot (UR5) and further illustrate its generalization capability.

The remainder of this paper is organized as follows. Section 2 briefly provides some background information about the Markov decision process (MDP), and the related works on robotic grasping methods. In Section 3, the KLSAC algorithm is proposed and analyzed theoretically. In Section 4, a self-learning robotic grasping controller based on the KLSAC algorithm is designed, and then comprehensive experimental studies on robotic grasping problems were conducted, including the simulated (based on V-REP simulation platform) and the real-world UR5 grasping tasks were carried out. Section 5 draws the conclusions and discusses future work.

2. Background

2.1. Markov decision process

RL is a powerful method to solve sequential decision-making problems through interactions between the environment and the agent. In RL, the problem is usually formulated as an MDP, which is defined by a tuple of $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where S represents the state space and A represents the action space; $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the state transition probability that describes the dynamics of the Markovian environment; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the immediate reward, and $\gamma \in (0, 1]$ represents the discount factor. The policy π of the MDP is defined as a mapping from the state space to the action space: $S \to \Pr(\mathcal{A})$, where $\Pr(\mathcal{A})$ is the probability distribution of the action space. This paper aims to design an efficient RL algorithm that can discover a near-optimal policy, maximizing the expected discounted return of robotic grasping tasks.

2.2. RL for robotic grasping

With promising applications in multiple areas arising in recent years [19,20], RL has received a lot of attention. Compared with classic control methods and learning from demonstration methods [21–23], RL-based robotic grasping methods do not require an accurate model, and the performance do not rely on expert domain knowledge [17]. Generally, RL algorithms can be divided into two classes: on-policy and off-policy RL algorithms. The policy being learned is called the target policy, and the policy used to generate behavior is called the behavior policy. In the off-policy case, we say that learning is from data "off" the target policy. Since the historical data can be reused, the sample efficiency of off-policy RL algorithms (e.g., DDPG [24], TD3 [25], SQL [26], SAC [27]) is usually better than that of on-policy algorithms (e.g., A3C [28], TRPO [29], PPO [30]). And it should be noticed that most RL-based robotic grasping methods prefer to adopt off-policy RL algorithms [9,10,14,15]. However, off-policy training with non-linear function approximation will make TD-based RL algorithms face the deadly triad problem [31], resulting in the danger of instability and divergence. According to the training framework, current mainstream RL-based robotic grasping methods can be divided into the following three directions:

The first direction is directly training policies with original inputs in the real world [11,12,32,33]. Sergey et al. [11] proposed to use hand-eye coordination to constantly observe the gripper and perform grasping behavior on real robots. However, the training process took over two months on 14 robots. There is no sim-to-real problem in such methods, while the cost is too huge for universal research to afford. Gu et al. [32] applied deep Q function training method on 3D manipulation tasks via updating policy asynchronously on multiple robots. Although the agent learned the door-open skill from scratch in the real world, the learning speed and practical applicability are disappointing.

The second direction is training policies with inputs from simulators [9,10,13,15,34]. Dmitry et al. [10] provided a scalable self-supervised RL framework (QT-Opt) which utilized multiple vision sensors for grasping, and QT-Opt responded dynamically to disturbances and perturbations. However, they did not test QT-Opt on real robots. Yang et al. [34] devised a novel end-to-end deep Q-learning framework for collaborative pushing and grasping. Actually, because of the gap between the simulator and the real world, most of these methods are difficult to generalize to the real systems.

The third direction is training policies with pre-defined or extracted feature representations [14,35–37], but not in an end-to-end way. This approach uses object detection techniques to get the environment information, decoupling feature representation learning from the policy training, which can reduce the training cost and be profitable to transfer the policy from simulation environments to real-world robots. Chen et al. [14] proposed a grasping system which integrates SAC and YOLO-v3 [38] object detection technique to grasp moving objects. This paper mainly focuses on the third direction, which aims to train robotic grasping policies using RL with extracted feature representations.

3. Methodology

In this section, the KLSAC algorithm including the sparse-kernel feature representation method is presented first. Then, the structure of the linear soft Q function approximator based on the least-squares method is introduced in detail. After that, the properties of KLSAC are analyzed theoretically, including the convergence property of the policy evaluation part and the algorithm's generalization capability.

Algorithm 1: Feature vectors construction with kernel sparsification

Input: Data set Ω; Kernel function $k(\cdot)$; Kernel width σ; Threshold μ.
Output: Kernel dictionary *Dic*.

```
Initialize Dic = {} ;
for each (s_i, a_i, s'_i, r_i) ∈ Ω do
    Compute ψ_i via Equations (3) ∼ (5) ;
    if ψ_i > μ then
        Add sample (s_i, a_i) into Dic ;
    end
end
```

3.1. Feature vector construction with kernel sparsification

In the LS^{2}BR algorithm, constructing appropriate feature vectors is crucial to the soft Q function approximation and can influence the learning performance of the critic module. Meanwhile, deep neural networks are effective at extracting feature representations, especially for pixel inputs. However, nonlinear feature vector construction often fails when applied to online RL [39]. In contrast, kernel-based feature representation methods, which have been widely used in RL [40,41], are appropriate for the robotic grasping problem under state measurement conditions (describing the state information in coordinate space). In addition, the sparsity of feature vectors from kernel sparsification can reduce the computational cost.

As is well known, a pivotal problem for applying kernel methods in learning algorithms is that the number of adjustable parameters is proportional to the number of samples. Consequently, when the sample size is huge, the computational cost will be explosive, and the generalization performance will also decrease. To solve the problem, various approaches for sparse kernel machines were studied [40,42]. In our method, we first generate a sparse sample set by employing the approximately linear dependency (ALD) analysis [40] and choosing appropriate kernel functions. And then the feature vectors are constructed from the selected sample set automatically. Different from previous works [40], ALD analysis is employed to construct the feature vectors of continuous state–action pairs.

Specifically, given a set of data $\Omega = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_L\}$, in which each element $\boldsymbol{c}_i = (\boldsymbol{s}_i, \boldsymbol{a}_i)$ is the concat of normalized state and action. $\phi(\boldsymbol{c})$ is defined as the mapping from state–action space $\mathcal{C}$ to reproducing kernel Hilbert spaces $\mathcal{H}$, and $k(\cdot,\cdot)$ is the kernel function mapping from $\mathcal{C}\times\mathcal{C}$ to $\mathbb{R}$. According to the property of Mercer function, we have $k\left(\boldsymbol{c}_i, \boldsymbol{c}_j\right) = \left\langle \phi\left(\boldsymbol{c}_i\right), \phi\left(\boldsymbol{c}_j\right)\right\rangle$ where $\langle\cdot,\cdot\rangle$ represents the inner product in $\mathcal{H}$. To choose linearly independent samples, the following criterion function is set up:

$$\psi_\upsilon = \min_{z} \left\| \sum_{\boldsymbol{c}_j \in \mathcal{D}_\upsilon} z_j \phi\left(\boldsymbol{c}_j\right) - \phi\left(\boldsymbol{c}_\upsilon\right) \right\|^2, \tag{1}$$

where $\mathcal{D}_\upsilon$ is the kernel dictionary with $d_{\upsilon-1}$ elements, $\mathbf{z} = \left[z_1, z_2, \ldots, z_{d_{\upsilon-1}}\right]^T \in \mathbb{R}^{d_{\upsilon-1}}$ represents the coefficient vector, and ψ_υ represents the evaluation value for sample $\boldsymbol{c}_\upsilon \in \Omega$. Therefore, we have:

$$\psi_\upsilon = \min_{\mathbf{z}} \left\{ \mathbf{z}^T \mathbb{K}_{\upsilon-1} \mathbf{z} - 2\mathbf{z}^T \mathbf{k}_{\upsilon-1}\left(\boldsymbol{c}_\upsilon\right) + \mathbf{k}_{\upsilon\upsilon} \right\}, \tag{2}$$

where $\mathbf{k}_{\upsilon\upsilon} = k\left(\boldsymbol{c}_\upsilon, \boldsymbol{c}_\upsilon\right)$, and $\mathbb{K}_{\upsilon-1}$ is a matrix composed by elements $k\left(\boldsymbol{c}_i, \boldsymbol{c}_j\right)$. The item $\mathbf{k}_{\upsilon-1}\left(\boldsymbol{c}_\upsilon\right)$ is computed as:

$$\mathbf{k}_{\upsilon-1}\left(\boldsymbol{c}_\upsilon\right) = \left[k\left(\boldsymbol{c}_1, \boldsymbol{c}_\upsilon\right), k\left(\boldsymbol{c}_2, \boldsymbol{c}_\upsilon\right), \ldots, k\left(\boldsymbol{c}_{d_{\upsilon-1}}, \boldsymbol{c}_\upsilon\right)\right]^T. \tag{3}$$

Then we get the optimum solution of Eq. (1):

$$\psi_\upsilon = \mathbf{k}_{\upsilon\upsilon} - \mathbf{k}_{\upsilon-1}^T\left(\boldsymbol{c}_\upsilon\right)\mathbf{z}_\upsilon, \tag{4}$$

where $\mathbf{z}_\upsilon$ can be computed as follows:

$$\mathbf{z}_\upsilon = \mathbb{K}_{\upsilon-1}^{-1}\mathbf{k}_{\upsilon-1}\left(\boldsymbol{c}_\upsilon\right). \tag{5}$$

If ψ_υ is greater than the predefined threshold μ, the sample c_υ is added to the $\mathcal{D}_\upsilon$, otherwise it will be ignored. After all samples are analyzed, we can get the subset $\Omega_{sub} = \left\{\boldsymbol{c}_{l_1}, \boldsymbol{c}_{l_2}, \ldots, \boldsymbol{c}_{l_n}\right\}$ to serve as the kernel dictionary. Applying the Gaussian kernel function to all elements in Ω_{sub}, the feature vector of a sample $\boldsymbol{c}$ is obtained by:

$$\phi(\boldsymbol{c}) = \left[k\left(\boldsymbol{c}, \boldsymbol{c}_{l_1}\right), k\left(\boldsymbol{c}, \boldsymbol{c}_{l_2}\right), \ldots, k\left(\boldsymbol{c}, \boldsymbol{c}_{l_n}\right)\right]^T, \tag{6}$$

$$k(\boldsymbol{c}, \boldsymbol{c}_{l_i}) = \exp\left(-(\boldsymbol{c} - \boldsymbol{c}_{l_i})^2/2\sigma^2\right), \tag{7}$$

where l_n is the dimension of the feature vector and σ is the kernel width of Gaussian kernel functions.

Based on the previous discussion, feature vector construction with kernel sparsification for the proposed LS^{2}BR algorithm is presented in Algorithm 1.

3.2. KLSAC algorithm

Soft Actor-Critic (SAC) [27] is one of the most popular DRL algorithms to be compared in many application scenarios. SAC integrates the entropy maximization mechanism into the standard optimization goal to encourage exploring. However, the use of multi-layer nonlinear neural networks in the critic module slows down the training speed and weakens the convergence guarantee, resulting in the demand for a large number of samples and the risk of falling into local convergence.

In order to solve these problems, we propose the Least-Squares Soft Bellman Residual method (LS^{2}BR) for approximating the soft Q function. Specifically, LS^{2}BR has made improvements in the following two aspects: (i) The nonlinear approximation structure constructed by multi-layer neural networks in critic is replaced by the linear approximation structure based on the construction of basis functions; (ii) The least-squares regression method for minimizing soft Bellman residual is designed to improve the sample efficiency and asymptotic performance.

In the critic part of SAC, the soft Q function network $Q_{\mathbf{w}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ is parameterized by $\mathbf{w}$ and updated through minimizing the soft Bellman residual with the data from historical datasets$\mathcal{D}$:

$$J_Q(\mathbf{w}) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_{\mathbf{w}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) - \mathcal{T}^\pi Q_{\mathbf{w}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)^2\right], \tag{8}$$

in which $\mathcal{T}^\pi$ represents the Bellman backup operator and can be defined as:

$$\mathcal{T}^\pi Q_{\mathbf{w}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) \triangleq r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) + \gamma \mathbb{E}_{\boldsymbol{s}_{t+1}\sim p}\left[V_{\bar{\mathbf{w}}}\left(\boldsymbol{s}_{t+1}\right)\right], \tag{9}$$

where the target soft Q function parameter $\bar{\mathbf{w}}$ is added to stabilize training. $V_{\bar{\mathbf{w}}}\left(\boldsymbol{s}_{t+1}\right)$ is the soft state value function derived from $Q_{\bar{\theta}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$:

$$V_{\bar{\mathbf{w}}}\left(\boldsymbol{s}_t\right) = \mathbb{E}_{\boldsymbol{a}_t\sim\pi}\left[Q_{\bar{\mathbf{w}}}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) - \alpha \log \pi\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)\right], \tag{10}$$

where α is the temperature parameter used to adjust the relative importance of the entropy item, thus controlling the intensity of exploration. To get over the drawbacks of the multi-layer nonlinear neural network of the critic module in SAC, we replace it with the linear structure and represented the approximated soft Q function as:

$$\hat{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t, \mathbf{w}) \doteq \mathbf{w}^\top \phi(\boldsymbol{c}_t) \doteq \sum_{i=1}^{d} w_i \phi_i(\boldsymbol{c}_t), \tag{11}$$

$\boldsymbol{c}_t = concat(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the state–action pair at step t, $\phi(\boldsymbol{c}_t) = (\phi_1(\boldsymbol{c}_t), \phi_2(\boldsymbol{c}_t), \ldots, \phi_d(\boldsymbol{c}_t))$ is the corresponding feature vector constructed by basis functions, $\mathbf{w}$ is the network weights, and d is the dimension of basis functions.

Based on the new soft Q function in Eq. (11), we can rewrite the soft Bellman residual as follows:

$$J_{\hat{Q}}(\mathbf{w}) = \mathbb{E}_{(\boldsymbol{s_t},\boldsymbol{a_t})\sim\mathcal{D}}\left[\frac{1}{2}\left(\sum_{i=1}^{d} w_i\phi_i(\boldsymbol{c}_t) - \mathcal{T}^{\pi}\sum_{i=1}^{d} w_i\phi_i(\boldsymbol{c}_t)\right)^2\right], \tag{12}$$

where $\mathcal{T}^{\pi}$ is the Bellman operator. Thus the soft Q function and the soft value function can be redefined as:

$$\mathcal{T}^{\pi}\hat{Q}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \mathbf{w}\right) \triangleq r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) + \gamma\mathbb{E}_{\boldsymbol{s}_{t+1}\sim p}\left[V\left(\boldsymbol{s}_{t+1}, \bar{\mathbf{w}}\right)\right], \tag{13}$$

$$V\left(\boldsymbol{s}_t, \bar{\mathbf{w}}\right) = \mathbb{E}_{\boldsymbol{a_t}\sim\pi}\left[\hat{Q}\left(\boldsymbol{s}_t, \boldsymbol{a}_t, \bar{\mathbf{w}}\right) - \alpha\log\pi\left(\boldsymbol{a_t} \mid \boldsymbol{s}_t\right)\right]. \tag{14}$$

To find the optimal weight $\mathbf{w}^*$, we can update $\mathbf{w}$ through stochastic gradient descent as following:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \beta_c\hat{\nabla}_{\mathbf{w}}J_Q(\mathbf{w}) \\ &= \mathbf{w}_t + \beta_c\left[r_{t+1}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) - \gamma\alpha\log\left(\pi_{\boldsymbol{\theta}}\left(\boldsymbol{a}_{t+1} \mid \boldsymbol{s}_{t+1}\right)\right)\right. \\ &\quad \left.+\mathbf{w}_t^{\top}\left(\gamma\phi\left(\boldsymbol{c}_{t+1}\right) - \phi\left(\boldsymbol{c}_t\right)\right)\phi\left(\boldsymbol{c}_t\right)\right],\end{aligned} \tag{15}$$

where β_c is the learning rate of the critic. To simplify the above update equation, the expectation of updated weight at each time step can be written as:

$$\mathbb{E}_t\left[\mathbf{w}_{t+1} \mid \mathbf{w}_t\right] = \mathbf{w}_t + \beta_c\left(\mathbf{b} - \mathbf{A}\mathbf{w}_t\right), \tag{16}$$

where

$$\mathbf{A} \doteq \mathbb{E}_t\left[\phi\left(\boldsymbol{c}_t\right)\left(\phi\left(\boldsymbol{c}_t\right) - \gamma\phi\left(\boldsymbol{c}_{t+1}\right)\right)^{\top}\right] \in \mathbb{R}^d \times \mathbb{R}^d, \tag{17}$$

and

$$\mathbf{b} \doteq \mathbb{E}_t\left[\left(r_{t+1} - \alpha\gamma\log\left(\pi_{\boldsymbol{\theta}}\left(\boldsymbol{a}_{t+1} \mid \boldsymbol{s}_{t+1}\right)\right)\right)\phi_t\right] \in \mathbb{R}^d. \tag{18}$$

When the solution converges to the optimal fixed point, we have $\mathbf{b} - \mathbf{A}\mathbf{w}^* = \mathbf{0}$, in which $\mathbf{w}^*$ represents the best solution for minimizing the soft Bellman residual. Then, based on Eqs. (17) and (18), the least-squares solution can be defined as:

$$\mathbf{w}^* \doteq \mathbf{A}^{-1}\mathbf{b}. \tag{19}$$

In order to stabilize the training process, the parameter $\bar{\mathbf{w}}$ of the soft target Q-network is synchronized by $\mathbf{w}^*$ periodically. Compared with the original method for evaluating the soft Q function in the SAC algorithm, LS^{2}BR holds the following three main advantages: (i) Historical samples from the replay buffer can be used simultaneously, which makes the learning process more efficient; (ii) There is no need to adjust the learning rate in LS^{2}BR; (iii) LS^{2}BR is unaffected by the initialization of the parameter $\boldsymbol{\theta}$.

Based on LS^{2}BR and sparse kernel feature representation, the Kernel-based Least-square Soft Bellman residual Actor–Critic (KLSAC) algorithm is introduced in this section. The critic of KLSAC is constructed and updated as described in Section 3.2. The feature vector is constructed as algorithm 1. As for the actor, KLSAC adopts the similar network architecture and update procedure as SAC, which performs the policy update by obeying the Boltzmann distribution of the soft Q function as much as possible. In other words, the policy is updated by minimizing the Kullback–Leibler divergence between the policy distribution and the Boltzmann distribution of the soft Q function with a reparameterization trick. The objective function is defined as:

$$\begin{aligned}J_{\pi}(\xi) = \mathbb{E}_{\boldsymbol{s}_t\sim D, \boldsymbol{\epsilon_t}\sim\mathcal{N}}\left[\alpha\log\pi_{\xi}\left(f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right) \mid \boldsymbol{s}_t\right)\right. \\ \left.-Q_{\boldsymbol{\theta}}\left(\boldsymbol{s}_t, f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right)\right)\right],\end{aligned} \tag{20}$$

where D is the historical dataset. $\boldsymbol{\epsilon_t}$ is an input noise vector for reparameterization, sampled from Gaussian distribution $\mathcal{N}$. $f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right)$ represents the reparameterized policy network. Stochastic gradient descent is used to optimize this objective function. The gradient of the objective function with respect to $\boldsymbol{\xi}$ is estimated as:

$$\begin{aligned}\hat{\nabla}_{\xi}J_{\pi}(\xi) =& \nabla_{\xi}\alpha\log\left(\pi_{\xi}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)\right) + \left(\nabla_{\boldsymbol{a}_t}\alpha\log\left(\pi_{\xi}\left(\boldsymbol{a}_t \mid \boldsymbol{s}_t\right)\right)\right. \\ &\left.-\nabla_{a_t}Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)\nabla_{\xi}f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right),\end{aligned} \tag{21}$$

Then the parameter $\boldsymbol{\xi}$ is updated in the following way:

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \beta_{\pi}\hat{\nabla}_{\xi}J_{\pi}(\boldsymbol{\xi}), \tag{22}$$

where β_{π} represents the learning rate.

The reparameterized network $f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right)$ is used to simulate the sampling process of actions, and it is defined as:

$$\boldsymbol{a}_t = f_{\xi}\left(\boldsymbol{\epsilon}_t; \boldsymbol{s}_t\right) = \mu_{\boldsymbol{a}} + \boldsymbol{\epsilon}_t \cdot \sigma_{\boldsymbol{a}}, \tag{23}$$

where $\mu_{\boldsymbol{a}}$ and $\sigma_{\boldsymbol{a}}$ represent the mean and variance of a Gaussian distribution, respectively. This Gaussian distribution is used to parameterize the policy network outputs as a continuous distribution, hence the policy is called the Gaussian policy [43].

The pseudocode of the KLSAC algorithm is presented in Algorithm 2. Fig. 1 illustrates the architecture of the proposed KLSAC algorithm, where the grasping simulation environment is constructed in V-REP simulator. Specifically, the actor module of the KLSAC agent interacts with the environment, collecting training data and improving the behavior policy iteratively; the critic module is used to evaluate the value of state–action pairs, guiding the optimizing direction of the policy training process in the actor module. The critic includes two soft Q-networks and two soft target Q-networks. The soft Q-networks provide the learned soft Q function to update the actor, and the soft target Q-networks are used to generate the soft value function to update the soft Q-networks according to Eq. (12)–Eq. (14). The weights of soft target Q-networks are obtained as an exponentially moving average of the two soft Q-networks weights, respectively. After training the policy from the simulator, the learned policy can be generalized to the real-world system, carrying out practical grasping tasks.

3.3. Performance analysis

The proposed KLSAC algorithm inherits the exploration capability of SAC. This is because that the actor in KLSAC is designed to approximate the Boltzmann distribution of the soft Q function rather than the unimodal distribution, which encourages the exploration between the max state–action function and other potential peaks, ignoring the low-value parts. This mechanism has been demonstrated to considerably increase exploration efficiency [27].

Moreover, compared with the original SAC algorithm, the innovations of KLSAC are reflected in two parts. The first is that the LS^{2}BR algorithm is proposed for the critic, leading to a better convergent property when doing policy evaluation. The second is the proposal of the feature vector construction approach via kernel sparsification, which improves the generalization capability of the learned policy, particularly for policy generalization from simulation to the real-world robot. Both improvements have significantly contributed to the improvement in learning performance for robotic grasping control. The detailed analysis of the two properties is as follows.

Tuomas et al. [27] provided the proof of soft policy evaluation convergence, however, it only applies to the situation when $|\mathcal{A}| < \infty$. Therefore, there is no guarantee that the soft policy evaluation of original SAC is convergent in continuous action

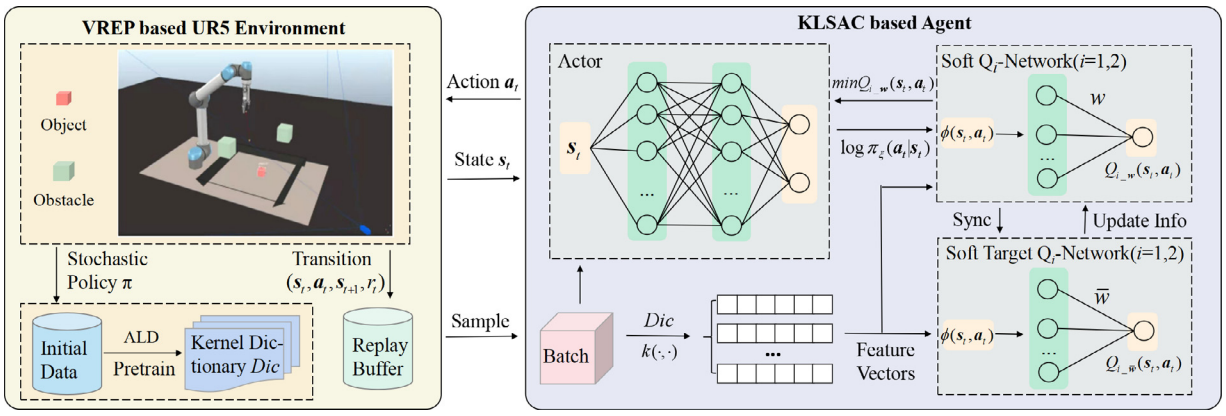


Fig. 1. Architecture of the KLSAC Algorithm. The agent interacts with the grasping environment at each time step and stores the experience in the replay buffer. Then a batch of transitions is sampled based on the pre-trained ALD feature constructor and utilized to train the actor and critic according to the proposed KLSAC algorithm. After that, the learned robotic grasping policy can be transferred straightly to the real-world UR5 robotic arm without retraining.

Algorithm 2: KLSAC

Input: Stochastic policy π_ξ; Critic parameter $\mathbf{w}$; Replay buffer $\mathbf{R}$; Batch size N; *episode* $= 0$; Learning rate β_π and β_c; Update delay d.
Output: A near-optimal policy.

Collect interaction data using an initialized policy ;
Obtain the dictionary *Dic* using Algorithm 1;
while *policy not converge* **do**
  episode $\leftarrow$ *episode* $+ 1$;
  Initialize the system and observe state $\boldsymbol{s}_0$;
  while *not at the terminate state* **do**
    Obtain action $\boldsymbol{a}_t$ based on π_ξ under state $\boldsymbol{s}_t$;
    Execute $\boldsymbol{a}_t$, receive state $\boldsymbol{s}_{t+1}$ and reward r_t;
    Store $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ into replay buffer $\mathbf{R}$;
    if *buffer is full* **then**
      Sample training batch $\mathcal{B}$;
      Construct feature vectors with *Dic* , $\mathcal{B}$;
      Update parameter $\mathbf{w}$ via Equation (19); **if** *episode mod d* $==$ 0 **then**
        Update parameter $\boldsymbol{\xi}$ via Equation (22) and (21);
      end
    end
    $\boldsymbol{s}_t \leftarrow \boldsymbol{s}_{t+1}$;
  end
end

tasks, e.g., robotic grasping control. The proposed LS^{2}BR algorithm employs the linear approximation structure and directly computes the best solution of soft Bellman residual. In fact, the linear approximation is guaranteed to converge to a fixed point with probability 1, which is the best solution for soft Bellman residual. Here we give the Soft Bellman residual fixed point theorem in Theorem 3.1.

Theorem 3.1 (*Soft Bellman Residual Fixed Point Theorem*)**.** *Given any MDP stationary distribution under policy π, and any group of feature vectors $\{\phi(\boldsymbol{c}_i) \mid i \in N\}$ is linearly independent. If the learning rate β_c in linear soft Bellman residual update equation meets the following conditions:*

$$0 \leq \beta_t \leq 1, \sum_{n=t}^{\infty} \beta_t = \infty, \sum_{t=1}^{\infty} \beta_t^2 < \infty, \tag{24}$$

as $t \to \infty$, the weights $\mathbf{w}$ of soft Q function converges to the soft Bellman residual fixed point $\mathbf{w}^*$ with probability 1.

We provide the detail proof of Theorem 3.1 in Appendix A. Theorem 3.1 shows that the linear structure for evaluating policies can converge to the soft Bellman residual fixed point, without any assumption about the action space. Therefore, the LS^{2}BR algorithm computes the soft Bellman residual fixed point directly for the policy evaluation in KLSAC, which theoretically guarantees a considerable improvement in learning efficiency, particularly for problems with continuous action space, such as robotic grasping.

Theorem 3.2. *Assume the soft Q function weight $\mathbf{w}^*$ is obtained using the LS^{2}BR algorithm with soft value iteration. We have*

$$\left\| V^\star - V^{\pi_{Q_{\mathbf{w}^*}}} \right\|_\infty \leq \frac{2 \left\| Q_{\mathbf{w}^*} - Q^\star \right\|_\infty}{1-\gamma}, \tag{25}$$

where $V^\star$ is the optimal soft value-function, and $Q^\star$ is the optimal soft action value-function. $\pi_{Q_{\mathbf{w}^*}}$ is the learned policy by approximating the Boltzmann distribution of soft Q function $Q_{\mathbf{w}^*}$.

Theorem 3.2 indicates that the suboptimality of the learning policy $\pi_{Q_{\mathbf{w}^*}}$ can be bounded by $\frac{2\|Q_{\mathbf{w}^*} - Q^\star\|_\infty}{1-\gamma}$. Therefore, by computing the soft Bellman residual fixed point $\mathbf{w}^*$ directly, we can obtain the near-optimal policy efficiently with the approximated soft Q function. The proof of Theorem 3.2 is similar to that of Corollary 2 in [44].

Furthermore, many DRL algorithms couple representation learning with value function approximation, increasing the complexity and structural risk. In KLSAC, we construct the feature vectors by the kernel sparsification method, which can decouple the representation learning from value function approximation. The generalization capability of KLSAC will be illustrated in the next section. In addition, according to previous theoretical analysis of ALD [45], the proposed feature vector construction method by kernel sparsification makes the assumption about feature vectors in Theorem 3.1 easily hold.

4. Case studies

In this section, in order to verify the effectiveness of the proposed KLSAC algorithm, robotic grasping experiments in simulation and real world were performed. We first introduced the construction of the simulation environment and the task settings. Then the experimental results from the simulation environment were analyzed. After that, the generalization property of KLSAC was investigated by doing a sim-to-real test in a realistic scene.

Table 1
Actor network structure in KLSAC.

Layers	Nodes	Activation function
First layer	128	Relu
Hidden layer 1	256	Relu
Hidden layer 2	256	Relu
Output layer	6	Tanh

Table 2
Hyperparameter settings.

Parameter name	Value
Replay buffer size R	1e6
Actor learning rate β_π	1e−4
Discount factor γ	0.99
Entropy regularization coefficient α	−6
Gradient clip coefficient ϵ	0.2
Actor update delay d	10
Batch size M	256
Generalized advantage estimation coefficient λ	0.95
Reward scale of target object e_1	10
Reward scale of target goal e_2	10
Reward scale of obstacle e_3	1000

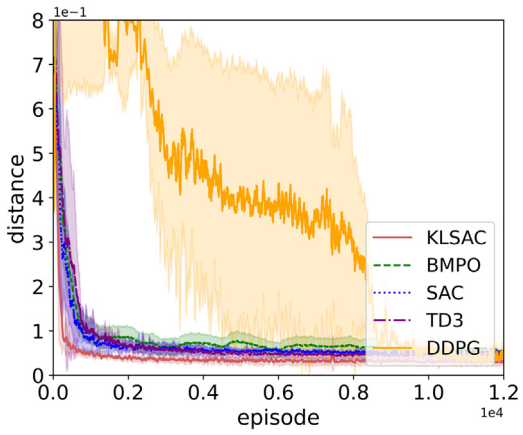


Fig. 2. The average distance between the gripper and the object of the five algorithms in the grasping without obstacle avoidance task. The horizontal axis represents the training episode, and the vertical axis represents the average distance between the gripper at the end of the robotic arm and the target object at the end of each episode.

4.1. Simulation platform and task setup

DRL algorithms usually require a large amount of interactive data to train the policies, making it unrealistic to train agents by DRL algorithms directly in the real world. The standard procedure is to build a simulation environment, train the agent in it, and then transfer the trained agent to the real robot. This not only improves the efficiency of training but also avoids adverse factors such as the wear and collision caused by realistic training. However, the problem that the policy cannot be transferred straightly caused by the gap between the simulation and the real environment must be solved. The difficulties mainly come from three aspects: the first is the attribute difference between the simulated and the real-world object; the second is the feature difference between the simulator and the real-world environment; the third is the physical model difference between the simulated and the real-world system.

In order to ensure that the policy learned in the simulation can be directly applied to the real world, we chose the V-REP (Virtual Robot Experiment Platform) simulator to build the robot grasping task scene. V-REP has many functional modules such as collision detection, kinematic model, path planning, various sensors such as cameras, radars, and depth cameras, which can support constructing simulation scenarios that are highly similar to real tasks. In addition, V-REP contains a variety of built-in robot model files including UR5 robot. By adding the UR5 model to the scene and setting approximate parameters, we can control it through commands and receive the corresponding state information such as the joint turning angle. A complete grasping scene was built after adding an extra table and cubic blocks. For the object to be grasped was set as a dynamic object, so that the robot could move it after grasping. For obstacles, the collision detection attribute was added, in order to judge whether the robot collided with them.

We set up two different grasping tasks, including the object grasping tasks with and without obstacle avoidance. The specific task design and the MDP modeling are described in Section 4.2.1. The goal of both tasks is to grasp objects with the shortest path in the smallest number of steps. The maximum number of steps in each episode is set as 20. And if the robot collides with any obstacles or the table, the episode will terminate immediately and start the next training episode. The KLSAC algorithm is based on an actor–critic architecture, in which the critic adopts a linear approximation structure as described in Section 3.2. Table 1 lists the structure of the actor network of KLSAC. Table 2 summarizes the hyperparameters of KLSAC involved in the simulation training.

4.2. Simulation results

The proposed KLSAC algorithm was compared with three typical DRL algorithms, namely Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic policy gradient (TD3), Soft Critic Actor (SAC) and Bidirectional Model-based Policy Optimization (BMPO) [46]. To make a fair comparison of different algorithms, we set five different random training seeds for each task and took the average results. The shaded area in the related graphs represents the corresponding standard deviation. DDPG, TD3 and SAC used the same hyperparameter settings as the implementation in stable-baselines 3 [47].

4.2.1. Grasping task without obstacle avoidance

The industrial robot UR5 has 6 degrees of freedom, which determines the position and posture of the robot. In the grasping task, the initial pose of the robot arm was fixed, while the location of the grasping target and the placement site was arbitrarily set in a 3D workspace. The state of this MDP model includes four parts: the joint angles $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$, the center coordinates of the gripper $\boldsymbol{g} = [x_g, y_g, z_g]$, the object position $\boldsymbol{b} = [x_o, y_o, z_o]$, and the target position $\boldsymbol{y} = [x_t, y_t, z_t]$. Hence, the state space $\mathcal{S} = [\boldsymbol{\theta}, \boldsymbol{g}, \boldsymbol{b}, \boldsymbol{y}]$ is continuous, and the total dimension is 15. The continuous action space is $\mathcal{A} = [\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6]$ where δ_i refers to the angle increment of ith joint and is normalized into $[-1, 1]$. The reward function is essential for the final performance, which is set as:

$$r(\boldsymbol{s}) = \frac{e_1}{d\left(\boldsymbol{b}_s, \boldsymbol{g}_s\right)} + \frac{e_2}{d\left(\boldsymbol{b}_s, \boldsymbol{y}_s\right)}, \tag{26}$$

where $d(\cdot)$ represents the euclidean distance between two points in three-dimensional space, and $\boldsymbol{b}_s$, $\boldsymbol{g}_s$, $\boldsymbol{y}_s$ represents the coordinates of object, gripper and target position respectively. e_1 and e_2 are constant used to adjust the reward scale.

As shown in Eq. (26), the reward function is designed based on the distance in different stages. In the approaching stage, the distance item $d\left(\boldsymbol{b}_s, \boldsymbol{g}_s\right)$ reduces to zero which results in gradually increasing positive reward and motivates the agent to grasp the object, while the another item $d\left(\boldsymbol{b}_s, \boldsymbol{y}_s\right)$ holds at certain constant. In the placing stage, the distance item $d\left(\boldsymbol{b}_s, \boldsymbol{y}_s\right)$ reduces to zero

Table 3
Performance statistics of KLSAC and the compared algorithms in the task of grasping without obstacle avoidance.

Metric	Episode	KLSAC	SAC	TD3	DDPG	BMPO
Distance	2000	**0.062±0.03**	0.084 ± 0.06	0.07 ± 0.07	0.813 ± 0.24	0.097 ± 0.09
	12000	**0.040±0.04**	0.060 ± 0.05	0.054 ± 0.06	0.103 ± 0.32	0.059 ± 0.07
Reward	2000	**5993±312**	5732 ± 408	5659 ± 613	87 ± 12	5973 ± 387
	12000	**6853±431**	6035 ± 397	6231 ± 793	5633 ± 211	6104 ± 413

Table 4
Performance statistics of KLSAC and the compared algorithms in the task of grasping with obstacle avoidance.

Metric	Episode	KLSAC	SAC	TD3	DDPG	BMPO
Distance	2000	**0.098±0.07**	0.187 ± 0.24	0.152 ± 0.18	0.48 ± 0.37	0.194 ± 0.30
	12000	**0.048±0.05**	0.056 ± 0.05	0.064 ± 0.07	0.121 ± 0.46	0.062 ± 0.08
Reward	2000	**4963±832**	4133 ± 1463	4265 ± 1213	76 ± 23	4469 ± 1072
	12000	**6531±181**	5635 ± 754	5191 ± 1043	4633 ± 2324	5498 ± 361

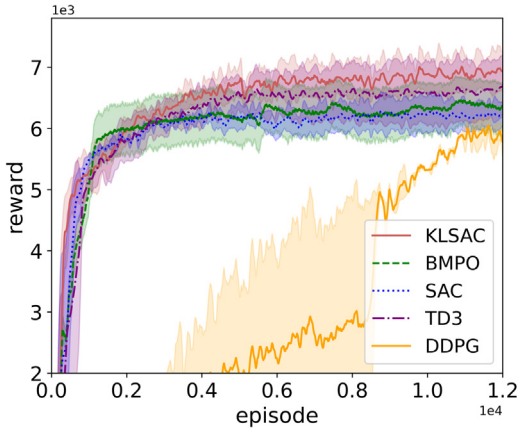


Fig. 3. The training curves of the five algorithms in the grasping without obstacle avoidance task. The horizontal axis represents the episode, and the vertical axis represents the accumulated reward of each episode.

and $d(\boldsymbol{b}_s, \boldsymbol{g}_s)$ stays unchanged. To prevent the reward from becoming infinite when the distance is close to zero, we truncated the distance to a reasonable limit. Compared with other reward function designs [34,48], we found empirically that this setting could make RL algorithms more efficient, primarily because the reward is more sensitive to distance.

Fig. 2 shows the average distance between the gripper and the object at the end of each episode in the grasping without obstacle avoidance task. The distance between the gripper and the object reflects the quality of grasping. It shows that the proposed KLSAC could learn to approach the object accurately, which helped the agent learn to grasp the object efficiently. Fig. 3 makes comparisons of the learning curve of the four algorithms, which proved that the proposed KLSAC algorithm held higher sample efficiency, more stable training capability, faster convergence speed, and better asymptotic performance. Other algorithms, such as TD3 and SAC, learned rather slowly and suffered from more training-process instability. The model-based RL algorithm, BMPO, converged faster than other RL algorithms on this task, however, the asymptotic performance of BMPO is not as good as KLSAC. The DDPG algorithm performed the worst, of which the learning process oscillated obviously, and took more than 10000 episodes to converge. The comparison results showed that KLSAC could learn the robot grasping policy more effectively.

In order to quantitatively compare the performance between different algorithms, we made statistics on the training process, which is shown in Table 3. The statistical results refer to the average statistics, including the distance and reward of 20 episodes before and after the specific episode of training. We compared the results at episode = 2000 and episode = 12000 respectively. The proposed KLSAC algorithm showed better performance and lower variance on both distance and reward metrics.

4.2.2. Grasping task with obstacle avoidance

Compared with the first task, this task required the robot to avoid two obstacles while grasping and placing the object from the initial position to the target position, which increased the difficulty significantly. Naturally, the state space is extended to $\boldsymbol{s} = [\boldsymbol{\theta}, \boldsymbol{g}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{o}_1, \boldsymbol{o}_2]$ where $\boldsymbol{o}_1 = [x_{o1}, y_{o1}, z_{o1}]$ and $\boldsymbol{o}_2 = [x_{o2}, y_{o2}, z_{o2}]$ represents the coordinates of two obstacles respectively. The action space is the same as the previous task. The reward function is set as following:

$$r(\boldsymbol{s}) = \frac{e_1}{d(\boldsymbol{b}_s, \boldsymbol{g}_s)} + \frac{e_2}{d(\boldsymbol{b}_s, \boldsymbol{y}_s)} - e_3, \tag{27}$$

where $d(\cdot)$ denotes the euclidean distance, e_3 is a penalty term when the robot collides with obstacles.

The training curves of the five algorithms were plotted, as shown in Fig. 4, KLSAC could learn to approach the object with better efficiency and precision. The advantage of KLSAC is more obvious than that in the first experiment, which proves that the proposed KLSAC algorithm can adapt to more difficult tasks. Similarly, in Fig. 5, We can see that the proposed KLSAC algorithm has obvious advantages in terms of learning efficiency, stability, and asymptotic performance than other algorithms. The training statistics of KLSAC and the three compared algorithms in this task are shown in Table 4. The proposed KLSAC algorithm surpassed TD3, SAC, BMPO and DDPG, proving that KLSAC was effective on simple tasks and maintained relative advantages on complex tasks.

The above simulation results demonstrate the effectiveness and superiority of the KLSAC algorithm in learning the robotic grasping control policy. The reason is mainly due to the following three aspects: the first is that the LS^{2}BR method is designed for the critic of KLSAC, which makes full use of the samples and accelerates the convergence speed of policy evaluation. Thus, the critic can provide more accurate information for the actor in each update step. The second is the feature construction based on the sparse kernel method, which effectively extracts useful information from the state and improves the accuracy of the policy update. Compared with the original SAC algorithm, KLSAC can achieve better asymptotic performance. The third is the application of the maximum entropy framework, which balances the relationship between the exploration and the exploitation problems in RL.

Fig. 6 shows that the robot successfully completed grasping with obstacle avoidance task in the simulation environment. The sub-images from 1 to 12 represent the steps from the starting position to grabbing the object and placing it in the target position.

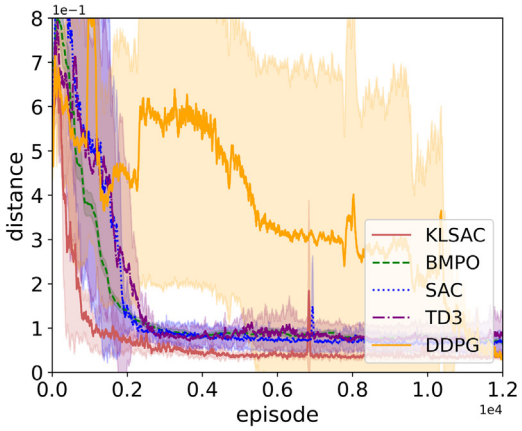


Fig. 4. The average distance between the gripper and the object of the four algorithms in the grasping with obstacle avoidance task. The horizontal axis represents the episode, and the vertical axis represents the average distance between the gripper at the end of the robotic arm and the target object at the end of each episode.

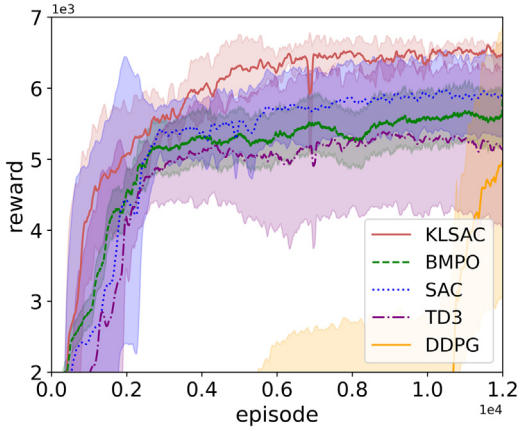


Fig. 5. The training curves of the four algorithms in the grasping with obstacle avoidance task. The horizontal axis represents the episode, and the vertical axis represents the accumulated reward of each episode.

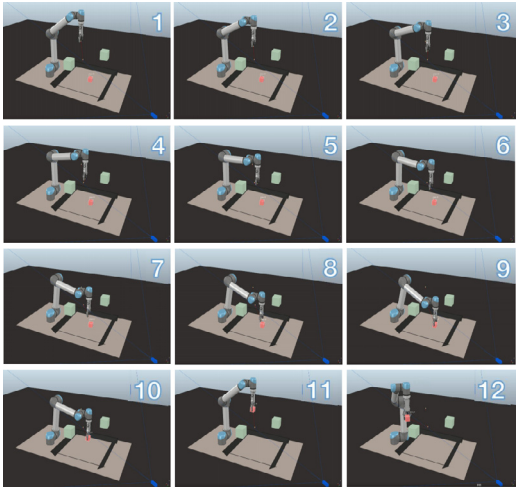


Fig. 6. The process of grasping task with obstacle avoidance. Sub-picture 1 represents the starting state; sub-picture 9 represents that the object was successfully picked up; sub-picture 12 represents that the object was accurately placed at the target position.

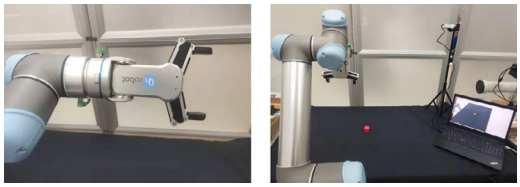


Fig. 7. The left picture shows the RG2 gripper, and the right picture shows the constructed physical system.

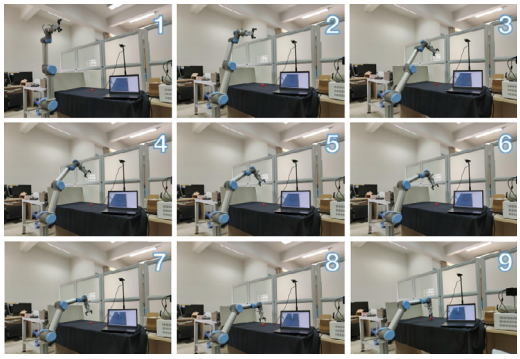


Fig. 8. The process of the grasping task without obstacle avoidance in real world. The sub-images from 1 to 9 represent the time step from the starting position to grabbing the object and placing it in the target position; sub-picture 1 represents the starting position; sub-picture 9 represents that the object was successfully picked.

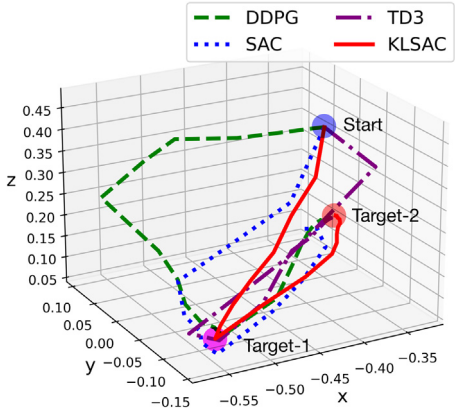


Fig. 9. The grasper moving trajectories of four algorithms in the real-world grasping task. The blue circle indicates the start position of the robot grasper. Target-1 indicates the position where the object should be grasped. Target-2 indicates the position where the robot should place the object to.

It can be seen that the agent trained by KLSAC could learn the grasping policy well, and select the shortest route to complete the grasping task, which is also shown in Fig. 9 for the real-world scene.

4.3. Sim-to-real test

In order to verify the effectiveness of KLSAC on the realistic problem, we built a real grasping system to conduct the grasping without obstacle avoidance task. The system includes a UR5 robot, an Inter real sense d435i depth camera, an RG2 gripper, and other necessary equipment, as shown in Fig. 7. We used the ArUco QR code to calibrate the depth camera. Then the d435i depth

camera was utilized to identify the target object on the desktop, obtaining the 3D coordinates of the object. We tested the policies trained the proposed KLSAC algorithm and other baselines in the simulator, which were saved after episode = 15000. Then the policies were directly transferred to the real world without any retraining or fine-tuning. Due to the generalization ability of the proposed KLSAC algorithm and the consistency between the simulation and the physical platform, the grasp success rate of KLSAC in the real UR5 grasping task without obstacles is over 97%, which is superior to DDPG (86%), TD3 (92%) and SAC (93%) and BMPO (91%). In the real UR5 grasping task with obstacles, the grasp success rate of KLSAC is 90%, which has obvious advantages compared with DDPG (72%), TD3 (76%) and SAC (83%) and BMPO (80%).

Fig. 8 shows the grasping process in the real world. It can be seen that the policy trained in the simulation environment can effectively complete the grasping task, without any performance decrease. This also shows that the proposed KLSAC algorithm has significant advantages in generalization performance, which is beneficial to practical applications. In addition, as shown in Fig. 9, we inversely deduced the position of the end of UR5 from the angle of each joint. In this task, the robot needs to grasp the object in Target-1 first, and then move it to Target-2. As can be seen, the trajectory of our proposed KLSAC algorithm (red line) is shorter and smoother than the other compared algorithms, which means that KLSAC learned a better grasping policy in this task. Meanwhile, in Fig. 10, we show the angle variation curves with the time step of three representative joints of UR5. The curves of the KLSAC algorithm are relatively stable, which demonstrated that the proposed KLSAC algorithm can perform the grasping task more efficiently and robustly than the other three algorithms.

5. Conclusion

In this paper, we proposed a novel DRL algorithm based on the least-squares soft Bellman residual under the actor–critic framework. Efforts are devoted to improving sample efficiency and learning stability for robotic grasping. On the basis of constructing features with kernel sparsification, a linear structure is used to approximate the soft Q function. The fixed point of minimizing soft Bellman residual is calculated by the least-squares method. Comprehensive experiment studies on robotic grasping learning control problems were carried out, which are considered as more challenging RL problems with high dimensions. The performance of KLSAC and typical DRL algorithms were compared analyzed. Based on our analysis and experiment results, the KLSAC algorithm was shown to have some significant merits when compared with previous DRL approaches. One advantage is that it is simple for practical implementation since there are fewer parameters that need to be tuned in the kernel-based feature construction. Fast feature construction can be easily implemented using ALD analysis because features can be automatically generated. Furthermore, KLSAC has obvious advantages compared with the other three DRL algorithms in the field of training stability and sample efficiency because of the linear function approximation structure. The sim-to-real results also demonstrate the practicability of the proposed KLSAC method. The policy trained in the simulation can be directly transferred to the real-world task.

There are still some problems that need further investigation such as theoretically analyzing the sample efficiency of the KLSAC algorithm and performing case studies on more challenging scenarios. Furthermore, how to deal with high-dimensional inputs rather than state measurements is a direction worthy of further study. It will be desirable to extend the application range of the KLSAC approach by using advanced vision representation technologies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Proofs

A.1. Soft Bellman residual fixed point theorem

Proof. Assume the parameter space is $\mathbb{W}^d$, and the vector norm is $\|\cdot\|_2$. Let the normed linear space is $(\mathbb{W}^d, \|\cdot\|_2)$, and the square matrix norm induced by vector norm for a square matrix $\mathbf{D} \in \mathbb{R}^d \times \mathbb{R}^d$ is defined as:

$$\| \mathbf{D} \|_2 \doteq \rho(\mathbf{D}^H \mathbf{D})^{\frac{1}{2}}, \tag{28}$$

where $\rho(\cdot)$ is the spectral radius of the matrix $\mathbf{D}$, which refers to the maximum value of the magnitude of $\mathbf{D}$'s eigenvalues.

According to the expectation update of weight $\mathbf{w}_t$ in Eq. (15), we can define a map $\mathbf{B} : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mathbf{B}(\mathbf{w}_t) = \mathbf{w}_t + \beta\,(\mathbf{b} - \mathbf{A}\mathbf{w}_t)\,. \tag{29}$$

To prove the map $\mathbf{B}$ is a contraction map, we need to prove that there exists $0 \le \alpha < 1$ satisfies $d(\mathbf{B}x, \mathbf{B}y) \le \alpha d(x, y)$. Suppose $\mathbf{w}_1$ and $\mathbf{w}_2$ are two points in the parameter space, then we can get:

$$\begin{aligned} d(\mathbf{B}\mathbf{w}_1, \mathbf{B}\mathbf{w}_2) &= \| \mathbf{B}(\mathbf{w}_1) - \mathbf{B}(\mathbf{w}_2) \|_2 \\ &= \| (\mathbf{w}_1 + \beta(\mathbf{b} - \mathbf{A}\mathbf{w}_1)) - (\mathbf{w}_2 + \beta(\mathbf{b} - \mathbf{A}\mathbf{w}_2)) \|_2 \\ &= \| (\mathbf{w}_1 - \mathbf{w}_2)(\mathbf{I} - \beta\mathbf{A}) \|_2 \\ &\le \| (\mathbf{w}_1 - \mathbf{w}_2) \|_2 \| (\mathbf{I} - \beta\mathbf{A}) \|_2\,. \end{aligned} \tag{30}$$

To guarantee $\| (\mathbf{I} - \beta\mathbf{A}) \|_2 \le 1$, we have:

$$d(\mathbf{B}\mathbf{w}_1, \mathbf{B}\mathbf{w}_2) \le \alpha \| \mathbf{w}_1 - \mathbf{w}_2 \|_2 = \alpha d(\mathbf{w}_1, \mathbf{w}_2), \tag{31}$$

which means map $\mathbf{B}$ is a contraction map. When the metric space $(\mathbb{W}^d, \|\cdot\|_2)$ is complete, there exists a unique fixed point $\mathbf{w}^* \in \mathbf{W}$ which satisfies $\mathbf{B}\mathbf{w}^* = \mathbf{w}^*$.

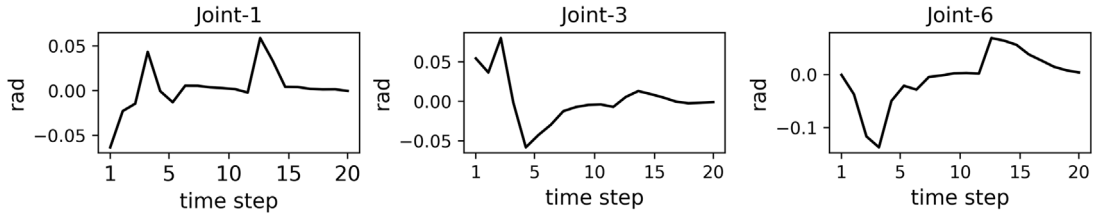
Therefore, in order to obtain such a fixed point, there are two conditions that need to meet: the first is the completeness of metric space $(\mathbb{W}^d, \|\cdot\|_2)$, and the second is the $\| (\mathbf{I} - \beta\mathbf{A}) \|_2 \le 1$. Considering the former is easy to prove, we will focus on the proof of the latter.

We consider the positive definiteness of the matrix $\mathbf{A}$, and the expected next weight vector $\mathbf{w}_{t+1}$ can be written as:

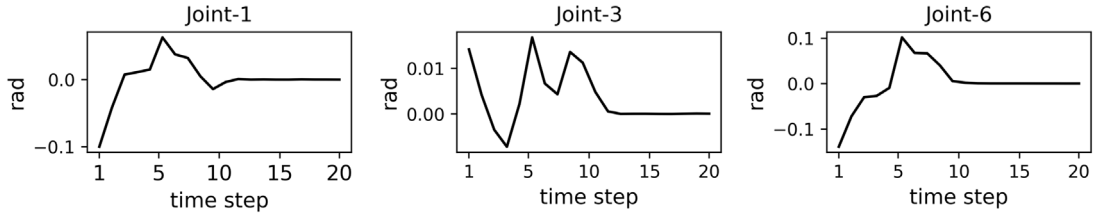
$$\begin{aligned} \mathbb{E}\,[\mathbf{w}_{t+1} \mid \mathbf{w}_t] &= \mathbf{w}_t + \beta\,(\mathbf{b} - \mathbf{A}\mathbf{w}_t) \\ &= (\mathbf{I} - \beta\mathbf{A})\mathbf{w}_t + \beta\mathbf{b}, \end{aligned} \tag{32}$$

where just the matrix $\mathbf{A}$ is multiplied by $\mathbf{w}_t$, so only $\mathbf{A}$ is related to the convergence.

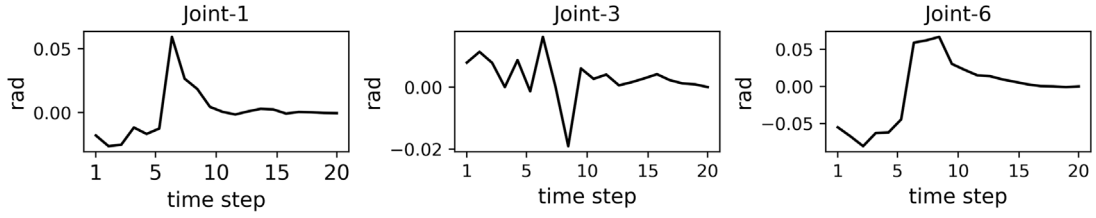
Consider the special case that $\mathbf{A}$ is a diagonal matrix. If any diagonal element of the matrix $\mathbf{A}$ is negative, the corresponding element in $(\mathbf{I} - \beta\mathbf{A})$ is greater than 1, and the element in $\mathbf{w}_t$ will be amplified, resulting in divergence. In contrast, if all diagonal elements of the matrix $\mathbf{A}$ are positive, the diagonal element of $(\mathbf{I} - \beta\mathbf{A})$ can be introduced between 0 and 1 by choosing a suitable β, which is smaller than the reciprocal of the largest element in $\mathbf{A}$. In this case, all components of $\mathbf{w}_t$ will be reduced when $(\mathbf{I} - \beta\mathbf{A})$ multiplying $\mathbf{w}_t$, so that the convergence is guaranteed. Therefore,



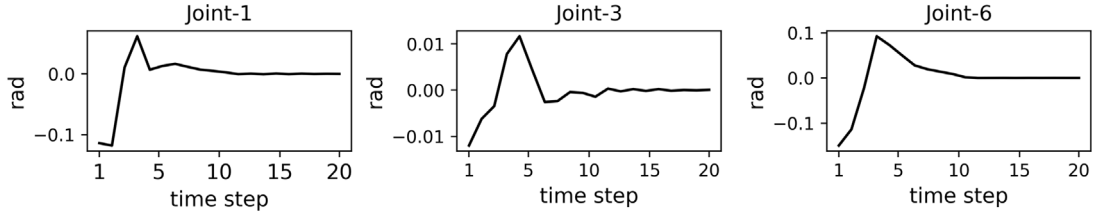
(a) DDPG



(b) SAC



(c) TD3



(d) KLSAC

Fig. 10. The curve of angle variation with time step. In this figure, joint-1, joint-3 and joint-6 of UR5 were chosen to compare the real-world performance between our proposed KLSAC algorithm and the other three algorithms.

the problem of convergence is transformed into the problem of **A**'s positive definiteness. The positive definite matrix **A** means that for any vector $\mathbf{y} \neq 0$, $\mathbf{y}^\top \mathbf{A}\mathbf{y} > 0$ is satisfied. Positive definite also ensures the existence of $\mathbf{A}^{-1}$.

For the linear soft Bellman residual, recalling Eq. (17), the matrix **A** can be written as:

$$\begin{aligned}\mathbf{A} &= \sum_{\boldsymbol{s}} \mu(\boldsymbol{s}) \sum_{\boldsymbol{a}} \pi(\boldsymbol{a} \mid \boldsymbol{s}) \sum_{r,\boldsymbol{s}'} p\left(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}\right) \phi(\boldsymbol{c}) \\ &\quad \left(\phi(\boldsymbol{c}) - \gamma \phi\left(\boldsymbol{c}'\right)\right)^\top \\ &= \sum_{\boldsymbol{c}} \mu(\boldsymbol{c}) \sum_{\boldsymbol{c}'} p\left(\boldsymbol{c}' \mid \boldsymbol{c}\right) \phi(\boldsymbol{c}) \left(\phi(\boldsymbol{c}) - \gamma \phi\left(\boldsymbol{c}'\right)\right)^\top \\ &= \sum_{\boldsymbol{c}} \mu(\boldsymbol{c}) \phi(\boldsymbol{c}) \left(\phi(\boldsymbol{c}) - \gamma \sum_{\boldsymbol{c}'} p\left(\boldsymbol{c}' \mid \boldsymbol{c}\right) \phi\left(\boldsymbol{c}'\right)\right)^\top \\ &= \boldsymbol{\Phi}^\top \mathbf{D}(\mathbf{I} - \gamma \mathbf{P}) \boldsymbol{\Phi},\end{aligned} \tag{33}$$

where $\mu(\boldsymbol{s})$ and $\mu(\boldsymbol{c})$ satisfy the stationary distribution of policy π, $p(\boldsymbol{c}'|\boldsymbol{c})$ is the transition probability from $\boldsymbol{c}$ to $\boldsymbol{c}'$, and **P** is the $(|S| + |A|) \times (|S| + |A|)$ matrix corresponding to this probability, **D** is the $(|S| + |A|) \times (|S| + |A|)$ diagonal matrix whose diagonal is $\mu(\boldsymbol{c})$, and $\boldsymbol{\Phi}$ is the $(|S| + |A|) \times d$ matrix whose row is $\phi(\boldsymbol{c})$. It can be found that the intermediate matrix $\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})$ in Eq. (33) is the key to the positive definite judgment of **A**.

According to the Verga matrix positive definite theorem [49]: any symmetric real matrix is positive definite if all of its diagonal entries are positive and greater than the sum of the corresponding off-diagonal entries. Since **P** is the probability matrix where each element $p_{ij} \in [0, 1]$ and **D** is the diagonal matrix of stationary distribution, hence the diagonal elements of the key matrix $\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})$ are positive while the off-diagonal elements are negative. Then what we have to show is that each row sum plus the corresponding column sum is positive. It is convenient to verify the row sum is positive because $p_{ij} \in [0, 1]$ and $\gamma < 1$, the last step is to show that the column sums are non-negative. Let **1**

denotes the column vector with all components equal to 1, and μ denotes the $(|S|+|A|)$-vector of stationary $\mu(\boldsymbol{c})$, then the column sums of the key matrix can be written as:

$$\begin{aligned}\mathbf{1}^\top \mathbf{D}(\mathbf{I}-\gamma\mathbf{P}) &= \boldsymbol{\mu}^\top(\mathbf{I}-\gamma\mathbf{P})\\ &= \boldsymbol{\mu}^\top - \gamma\boldsymbol{\mu}^\top\mathbf{P}\\ &= \boldsymbol{\mu}^\top - \gamma\boldsymbol{\mu}^\top\\ &= (1-\gamma)\boldsymbol{\mu}^\top,\end{aligned} \tag{34}$$

where all the components of $(1-\gamma)\boldsymbol{\mu}^\top$ are positive, thus the key matrix $\mathbf{D}(\mathbf{I}-\gamma\mathbf{P})$ and matrix $\mathbf{A}$ are both positive definite, which proves the convergence of the soft Q-value function parameter $\mathbf{w}$ in Eq. (16).

By now we have proved that the matrix $\mathbf{A}$ is positive definite, while our purpose is to prove the $\| (\mathbf{I}-\beta\mathbf{A}) \|_2 \leq 1$. Because matrix A is positive definite, all its eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_d)$ are positive. Then the eigenvalues of the matrix $(\mathbf{I}-\beta\mathbf{A})$ are $(1-\beta\lambda_1, 1-\beta\lambda_2, \ldots, 1-\beta\lambda_d)$. Suppose λ_{min} is the smallest eigenvalue of matrix $\mathbf{A}$, according to the definition in Eq. (28) the norm of matrix $(\mathbf{I}-\beta\mathbf{A})$ can be computed as:

$$\| (\mathbf{I}-\beta\mathbf{A}) \|_2 = |1-\beta\lambda_{min}|, \tag{35}$$

which means that we can always choose an appropriate β such that $|1-\beta\lambda_{min}| \leq 1$ and then $\| (\mathbf{I}-\beta\mathbf{A}) \|_m \leq 1$. Therefore, the map $\mathbf{B}$ is a contraction map and $\mathbf{B}\mathbf{w}^* = \mathbf{w}^*$ exists. That is:

$$\mathbf{B}(\mathbf{w}^*) = \mathbf{w}^* + \beta\left(\mathbf{b} - \mathbf{A}\mathbf{w}^*\right) = \mathbf{w}^*, \tag{36}$$

which derives $\mathbf{b} - \mathbf{A}\mathbf{w}^* = 0$, and then $\mathbf{w}^* = \mathbf{A}^{-1}\mathbf{b}$. Therefore, as $t \to \infty$, the weights $\mathbf{w}$ of soft Q function converges to the soft Bellman residual fixed point $\mathbf{w}^*$ with probability 1.

Appendix B. Additional experimental results

For clearness, we show the curve of angle variation of joint-1, joint-3 and joint-6 in Fig. 10. The detailed description of analysis can be found in the last paragraph in Section 4.3.

Appendix C. Experimental videos

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104385.These experimental videos are provided to show the performance of the proposed KLSAC algorithm on a real robotic platform.

References

[1] Oliver Kroemer, Scott Niekum, George Dimitri Konidaris, A review of robot learning for manipulation: Challenges, representations, and algorithms, J. Mach. Learn. Res. 22 (1) (2021) 1395–1476.

[2] Bipan Zou, René De Koster, Yeming Gong, Xianhao Xu, Guwen Shen, Robotic sorting systems: Performance estimation and operating policies analysis, Transp. Sci. 55 (6) (2021) 1430–1455.

[3] Tierui He, Shoaib Aslam, Zhekai Tong, Jungwon Seo, Scooping manipulation via motion control with a two-fingered gripper and its application to bin picking, IEEE Robot. Autom. Lett. 6 (4) (2021) 6394–6401.

[4] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, Jeannette Bohg, Concept2Robot: Learning manipulation concepts from instructions and human demonstrations, Int. J. Robot. Res. 40 (12–14) (2021) 1419–1434.

[5] Yoshihisa Tsurumine, Yunduan Cui, Eiji Uchibe, Takamitsu Matsubara, Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation, Robot. Auton. Syst. 112 (2019) 72–83.

[6] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, Wojciech Zaremba, Learning dexterous in-hand manipulation, Int. J. Robot. Res. 39 (1) (2020) 3–20.

[7] Heecheol Kim, Yoshiyuki Ohmura, Yasuo Kuniyoshi, Gaze-based dual resolution deep imitation learning for high-precision dexterous robot manipulation, IEEE Robot. Autom. Lett. 6 (2) (2021) 1630–1637.

[8] Henry Charlesworth, Giovanni Montana, Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning, in: Proceedings of the 38th International Conference on Machine Learning, ICML, 2021, pp. 1496–1506.

[9] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, James Bergstra, Benchmarking reinforcement learning algorithms on real-world robots, in: Proceedings of the 2nd Annual Conference on Robot Learning, Vol. 87, CoRL, 2018, pp. 561–591.

[10] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, Sergey Levine, Scalable deep reinforcement learning for vision-based robotic manipulation, in: Proceedings of the 2nd Annual Conference on Robot Learning, Vol. 87, CoRL, 2018, pp. 651–673.

[11] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, Deirdre Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, Int. J. Robot. Res. 37 (4–5) (2018) 421–436.

[12] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav S. Sukhatme, Stefan Schaal, Sergey Levine, Combining model-based and model-free updates for trajectory-centric reinforcement learning, in: Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 703–711.

[13] Weihao Yuan, Johannes A. Stork, Danica Kragic, Michael Yu Wang, Kaiyu Hang, Rearrangement with nonprehensile manipulation using deep reinforcement learning, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 270–277.

[14] Pengzhan Chen, Weiqing Lu, Deep reinforcement learning based moving object grasping, Inform. Sci. 565 (2021) 62–76.

[15] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, Sergey Levine, Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 6284–6291.

[16] Bohan Wu, Iretiayo Akinola, Abhi Gupta, Feng Xu, Jacob Varley, David Watkins-Valls, Peter K. Allen, Generative attention learning: a "GenerAL" framework for high-performance multi-fingered grasping in clutter, Auton. Robots 44 (6) (2020) 971–990.

[17] Marwan Qaid Mohammed, Kwek Lee Chung, Chua Shing Chyi, Review of deep reinforcement learning-based object grasping: techniques, open challenges, and recommendations, IEEE Access 8 (2020) 178450–178481.

[18] Han Hu, Kaicheng Zhang, Aaron Hao Tan, Michael Ruan, Christopher Agia, Goldie Nejat, A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain, IEEE Robot. Autom. Lett. 6 (4) (2021) 6569–6576.

[19] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, Demis Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.

[20] Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, Ziyu Wang, Autonomous navigation of stratospheric balloons using reinforcement learning, Nature 588 (7836) (2020) 77–82.

[21] Pengyuan Wang, Fabian Manhardt, Luca Minciullo, Lorenzo Garattoni, Sven Meier, Nassir Navab, Benjamin Busam, DemoGrasp: Few-shot learning for robotic grasping with human demonstration, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2021, pp. 5733–5740.

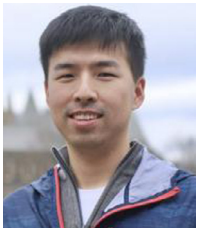
[22] Daichi Kawakami, Ryoichi Ishikawa, Menandro Roxas, Yoshihiro Sato, Takeshi Oishi, Learning 6DoF grasping using reward-consistent demonstration, 2021, arXiv preprint arXiv:2103.12321.

[23] Shuran Song, Andy Zeng, Johnny Lee, Thomas A. Funkhouser, Grasping in the wild: learning 6DoF closed-loop grasping from low-cost demonstrations, IEEE Robot. Autom. Lett. 5 (3) (2020) 4978–4985.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[25] Scott Fujimoto, Herke van Hoof, David Meger, Addressing function approximation error in actor-critic methods, in: Proceedings of the 35th International Conference on Machine Learning, ICML, 2018, pp. 1582–1591.
[26] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, Sergey Levine, Reinforcement learning with deep energy-based policies, in: Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, pp. 1352–1361.
[27] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Piet er Abbeel, Sergey Levine, Soft actor-critic algorithms and applications, 2018, arXiv preprint arXiv:1812.05905.
[28] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: Proceedings of the 33nd International Conference on Machine Learning, ICML, 2016, pp. 1928–1937.
[29] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, Philipp Moritz, Trust region policy optimization, in: Proceedings of the 32nd International Conference on Machine Learning, ICML, 2015, pp. 1889–1897.
[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[31] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[32] Shixiang Gu, Ethan Holly, Timothy Lillicrap, Sergey Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2017, pp. 3389–3396.
[33] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, Sergey Levine, Composable deep reinforcement learning for robotic manipulation, in: Proceedings of IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 6244–6251.
[34] Yuxiang Yang, Zhihao Ni, Mingyu Gao, Jing Zhang, Dacheng Tao, Collaborative pushing and grasping of tightly stacked objects via deep reinforcement learning, IEEE/CAA J. Autom. Sin. 9 (1) (2022) 135–145.
[35] Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, Jeannette Bohg, Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks, in: Proceedings of International Conference on Robotics and Automation, ICRA, 2019, pp. 8943–8950.
[36] Yifei Shi, Xin Xu, Junhua Xi, Xiaochang Hu, Dewen Hu, Kai Xu, Learning to detect 3d symmetry from single-view rgb-d images with weak supervision, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (4) (2023) 4882–4896.
[37] Yifei Shi, Zixin Tang, Xiangting Cai, Hongjia Zhang, Dewen Hu, Xin Xu, Symmetrygrasp: symmetry-aware antipodal grasp detection from single-view rgb-d images, IEEE Robotics and Automation Letters 7 (4) (2022) 12235–12242.
[38] Joseph Redmon, Ali Farhadi, YOLOv3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.
[39] Vincent Liu, Sparse Representation Neural Networks for Online Reinforcement Learning (Ph.D. thesis), University of Alberta, 2019.
[40] Xin Xu, Dewen Hu, Xicheng Lu, Kernel-based least squares policy iteration for reinforcement learning, IEEE Trans. Neural Netw. 18 (4) (2007) 973–992.
[41] Guang Yang, Xingguo Chen, Shangdong Yang, Huihui Wang, Shaokang Dong, Yang Gao, Online attentive kernel-based temporal difference learning, 2022, arXiv preprint arXiv:2201.09065.
[42] Jyrki Kivinen, Alexander J. Smola, Robert C. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (8) (2004) 2165–2176.
[43] Ofir Nachum, Mohammad Norouzi, George Tucker, Dale Schuurmans, Smoothed action value functions for learning gaussian policies, in: Proceedings of the 35th International Conference on Machine Learning, ICML, 2018, pp. 3692–3700.
[44] Satinder P. Singh, Richard C. Yee, An upper bound on the loss from approximate optimal-value functions, Mach. Learn. 16 (3) (1994) 227–233.
[45] Yaakov Engel, Shie Mannor, Ron Meir, The kernel recursive least-squares algorithm, IEEE Trans. Signal Process. 52 (8) (2004) 2275–2285.
[46] Hang Lai, Jian Shen, Weinan Zhang, Yong Yu, Bidirectional model-based policy optimization, in: Proceedings of the 37th International Conference on Machine Learning, ICML, 2020, pp. 5618–5627.
[47] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, Noah Dormann, Stable-Baselines3: Reliable reinforcement learning implementations, J. Mach. Learn. Res. 22 (268) (2021) 1–8.
[48] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, Sergey Levine, Scalable deep reinforcement learning for vision-based robotic manipulation, in: Proceedings of the 2nd Annual Conference on Robot Learning, CoRL, 2018, pp. 651–673.
[49] Richard S. Varga, Iterative Analysis, Springer, 1962.

Yixing Lan received the B.S. degree from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2016, and the M.S. degree from National University of Defense Technology, Changsha, China, in 2018. He is currently pursuing the Ph.D. degree. His current research interests include reinforcement learning, transfer learning and robotics.



Junkai Ren is currently an Assistant Professor at National University of Defense Technology (NUDT), Changsha, China. He received his Ph.D. degree in control science and engineering from NUDT in 2021. During 2018–2019, he was a visiting student at the University of Alberta, advised by Prof. Richard Sutton. His current research interests include reinforcement learning, robot manipulation and autonomous vehicles.

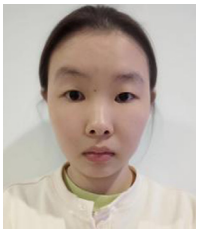
Tao Tang received his Bachelor Degree from National University of Defense Technology, Changsha, China, in 2020. He is currently pursuing the Doctor Degree in the same university in the direction of machine learning. His current research interests include reinforcement learning, transfer learning and meta-learning.

Xin Xu received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, where he received the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 2002.

He has been a visiting professor in Hong Kong Polytechnic University, University of Alberta, University of Guelph, and the University of Strathclyde, U.K., respectively. He is currently a Professor with the College of Intelligence Science and Technology, NUDT, China. He has co-authored more than 160 papers in international journals and conferences, and co-authored four books. His research interests include intelligent control, reinforcement learning, approximate dynamic programming, machine learning, robotics, and autonomous vehicles.

He received the Fork Ying Tong Youth Teacher Fund of China in 2008 and the 2nd class National Natural Science Award of China, in 2012. He serves as the co-Editor-in-Chief of Journal of Intelligent Learning Systems and Applications and the Associate Editor-in-Chief of CAAI Transactions on Intelligent Technology (Elsevier). He is an Associate Editor of Information Sciences, Intelligent Automation and Soft Computing, and Acta Automatica Sinica. He was a Guest Editor of the International Journal of Adaptive Control and Signal Processing and Mathematical Problems in Engineering. He is a Senior Member of IEEE and a member of the IEEE CIS Technical Committee on Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the IEEE RAS Technical Committee on Robot Learning.

Yifei Shi is an Assistant Professor at National University of Defense Technology (NUDT). He received his Ph.D. degree in computer science from NUDT in 2019. During 2017–2018, he was a visiting student research collaborator at Princeton University, advised by Thomas Funkhouser. His research interests mainly include computer vision, 3D reconstruction and object/scene understanding. He has published 20+ papers in top tier conferences and journals, including CVPR, ECCV, ICCV, SIGGRAPH Asia and ACM Transactions on Graphics.



Zixin Tang received her Bachelor Degree from Sichuan University, Chengdu, China, in 2020. She is currently pursuing the Master Degree in National University of Defense Technology, Changsha, China. Her current research interests include computer vision, deep reinforcement learning, and robot learning.