Robotica (2026), **44**, pp. 540–564
doi:10.1017/S0263574726103191

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Robot autonomous learning system based on multi-source sensor information fusion

Ze Cui[1], Chenzhao Sun[1], Zenghao Chen[2], Jing'ao Huang[1] and Yang Zhang[1]

[1]College of Mechanical and Electrical Engineering and Automation, Shanghai University, China
[2]Shanghai Aerospace Control Technology Institute, Shanghai Engineering Research C, China
Corresponding author: Ze Cui; Email: cuize0421@126.com

Received: 3 April 2025; **Revised:** 11 December 2025; **Accepted:** 15 January 2026; **First published online:** 18 February 2026

Keywords: machine learning; deep reinforcement learning; demonstration learning; SAC algorithm

Abstract

With the rapid development of artificial intelligence technology, robotics, as its core branch, has attracted extensive attention from researchers. This paper designs and develops a robotic arm learning system based on multi-source sensor information fusion, which investigates the autonomous learning capability of robotic arms by closely integrating deep reinforcement learning (DRL) as the core framework for skill acquisition. By incorporating imitation learning as a source of expert prior and leveraging DRL's intrinsic ability for policy optimization through environmental exploration, the proposed system achieves both rapid learning and robust generalization. Specifically, we introduce the gradient penalty mechanism from Wasserstein generative adversarial networks (WGANs), a technique that improves the stability of adversarial training by penalizing gradients that deviate from a specified norm. This mechanism is incorporated into the soft actor-critic (SAC) algorithm, a widely used off-policy DRL method known for its sample efficiency and robust performance in continuous control tasks. The resulting SAC-GP (SAC-gradient penalty) algorithm benefits from both SAC's stable policy learning and WGAN's improved training regularization, leading to superior convergence speed and system stability. Furthermore, this paper proposes a hybrid learning framework by combining generative adversarial imitation learning (GAIL) with SAC-GP, enabling the agent to benefit from both demonstration-based policy initialization and continuous self-improvement via reinforcement learning. Finally, a door-opening experiment is designed to verify the learning and execution capabilities of the system in both virtual and real environments. Experimental results demonstrate that the proposed learning system possesses excellent learning and motion execution abilities in practical applications. This achievement not only provides new insights for research in robot learning but also lays a solid foundation for the future development of robotic technology.

1. Introduction

With the rapid development of technology and the increasing trend of population aging, robots are playing an increasingly important role in social life. In the manufacturing industry, robots have become a core component of automated production lines. Collaborative robots and new logistics robots are driving the development of flexible manufacturing [1] and intelligent warehousing. They can perform high-precision and high-efficiency tasks, significantly improving production efficiency and product quality. In the healthcare field, robot technology is also changing the form and quality of medical services. Surgical robots [2] can assist doctors in performing precise, minimally invasive surgeries, improving surgical success rates. Rehabilitation robots help patients with rehabilitation training, accelerating the recovery process. In the agricultural sector, agricultural robots [3] are promoting the development of smart agriculture. They can precisely perform tasks such as planting, fertilizing, and spraying pesticides, improving agricultural production efficiency. As technological innovation continues and the scope of applications expands, robots with improved self-learning and operational capabilities are becoming

© The Author(s), 2026. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

increasingly prevalent and influential across diverse areas. These advancements not only hold substantial theoretical value but also present profound societal significance and practical application prospects.

Robot learning [4] enables robots to acquire the ability to autonomously learn and execute tasks. Robot learning refers to the process where robots continuously improve their skills and strategies through self-exploration and learning. The core of this method lies in utilizing artificial intelligence technologies such as machine learning to allow robots to automatically adjust and optimize their behavior through interactions with the environment, thus achieving self-learning [5]. Robot learning requires robots to have the ability to perceive the surrounding environment in real time [6] and interact with it [6]. With traditional robot control methods, robots cannot pre-acquire environmental parameters and task models for complex and variable environments, resulting in ineffective interactions with the surrounding environment and an inability to complete tasks. Reinforcement learning (RL) is considered as the well-performing machine learning algorithm, which guarantees to obtain actions by the agents in an environment to maximize the rewards [7]. RL enables robotic agents to continually improve their decision-making and control strategies through direct interaction with the environment, receiving environmental feedback in the form of rewards and penalties. This trial-and-error mechanism allows for the discovery of optimal or near-optimal solutions in complex, uncertain, and dynamic scenarios without explicit supervision. With advances in deep learning techniques, deep reinforcement learning (DRL) further enhances the scalability and representational power of RL, making it applicable to high-dimensional sensory inputs and intricate manipulation tasks.

Using RL for robot control [8] does not require pre-established control strategies but instead allows robots to autonomously learn methods and strategies to solve tasks through interactions with the environment. In this process, the robot's policy network is continuously iterated and updated until the policy network converges and the robot learns an optimal task-solving strategy. With the continuous development of artificial intelligence technologies, robot learning methods will continue to be improved and perfected, providing broader prospects for the development and application of robots.

Building upon these advancements, this study develops a robotic arm learning system that synergistically combines DRL with imitation learning under the multi-sensor information fusion paradigm to achieve autonomous and robust door-opening maneuvers.

The main contributions of this work are as follows: (1) we improve the soft actor-critic (SAC) algorithm by incorporating Wasserstein generative adversarial network (WGAN)'s gradient penalty, resulting in SAC-gradient penalty (SAC-GP), which enhances learning stability and efficiency. By further integrating generative adversarial imitation learning (GAIL)-based imitation learning, the algorithm achieves higher performance and faster convergence. Experiments – including block lifting and door-opening tasks – demonstrate superior performance and efficiency compared to baseline methods. (2) We construct both simulation and real-world experiment environments. The simulation is built on the MuJoCo physics engine with the robosuite framework. The real-world setup employs a Sawyer robot with an AG-95 gripper as actuators, combined with a NOKOV optical motion capture system and a six-axis Srisensor force sensor for perception. (3) We deploy agents trained in simulation to the real Sawyer platform for tasks such as grasping, rotating, pulling, and opening a door handle. Experimental results show that the proposed self-learning system exhibits high functional completeness in both hardware and algorithm, strong learning and execution capabilities for complex tasks, and exhibits stable and smooth operation in real-world conditions.

2. Related work

RL is a method that solves decision optimization problems through interactive learning [9]. Its core idea is that an agent attempts different actions and observes the results through interactions with the environment, learning how to maximize the cumulative reward from the environment and ultimately finding the optimal strategy to achieve the goal. Due to their excellent ability to approximate nonlinear functions and extract relevant features from raw inputs, deep neural networks have been introduced into RL, leading to the emergence of DRL [10]. As a bridge connecting deep learning and RL, DRL combines

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

the perceptual capabilities of deep learning with the decision-making capabilities of RL, enabling the autonomous decision-making of the agent through end-to-end learning [11].

Tuomas [12, 13] proposed the SAC algorithm, which is based on the maximum entropy RL framework and aims to improve exploration, stability, and sample efficiency by maximizing the expected return and entropy of the policy. This method encourages the agent to perform actions as randomly as possible when executing tasks, enhancing exploration and robustness. To demonstrate the effectiveness of the SAC algorithm, the authors validated it on a three-finger, dexterous robotic hand platform. After 20 h of training, the robotic hand learned the strategy of rotating valves through image input and successfully completed the task.

Guo [14] proposed a RL algorithm based on SAC, which achieves real-time optimal feedback control by using a composite auxiliary reward structure and sum-tree priority experience replay (SAC-SP). This method can achieve real-time navigation of automated guided vehicles (AGVs) in various obstacle scenarios and high-accuracy prediction of optimal trajectories under unknown conditions.

Cao [10] proposed combining path planning (PP) control and the SAC algorithm, where PP control is responsible for steering while the SAC algorithm handles speed control, establishing a complementary relationship between the two. This combination allows PP control to compensate for the deficiencies of the SAC algorithm through adaptive speed adjustment, thereby improving path convergence. Their simulation and experimental results demonstrate that the algorithm is easy to train and has advantages in tracking paths with different curvatures.

In specific applications, DRL often requires more training time and computational resources and may be affected by sparse rewards and difficulties in exploration in some cases, greatly increasing training costs and reducing training efficiency [15]. Demonstration learning is a paradigm in which an agent learns to perform a task by imitating the behavior of an expert shown in demonstrations [16]. Therefore, people have considered combining demonstration learning and RL to address the issue of low sample utilization efficiency and improve training efficiency [17].

Huang [18] proposed an efficient RL algorithm named demonstration-guided exploration (DEX), which combines RL with demonstration learning to overcome exploration issues in surgical robots. This method effectively utilizes expert data to improve exploration efficiency. It uses nonparametric regression to estimate higher values for behaviors similar to those of experts, promoting productive interactions. Extensive experiments on 10 surgical manipulation tasks on the surgical simulation platform SurRoL showed that the designed method significantly improves exploration efficiency and task success rates.

Zhu [19] proposed a method combining RL and demonstration learning, using teaching data to reduce the difficulty of exploration in continuous domains and utilizing this data to accelerate and stabilize the learning of visual–motor policies in multi-stage tasks. This method enables end-to-end learning, learning directly from pixel inputs and making actions through a visual–motor controller. The authors evaluated their method on six different manipulation tasks, including stacking blocks and pouring water, and validated it on real robots.

Wu [20] proposed the DayDreamer algorithm, which addresses the learning problem for robots executing tasks in complex environments. This algorithm combines actor-critic networks from RL and reduces the number of trial-and-error attempts in the real environment by planning within the learned world model. The authors applied the DayDreamer algorithm to four different robots, demonstrating its ability to learn in the real world. These robots included a quadruped robot, two different robotic arms, and a wheeled robot. The tasks covered various challenges, including continuous and discrete actions, dense and sparse rewards, proprioception and camera inputs, and sensor fusion with multiple input modes.

Furthermore,recent advances have paid considerable attention to skill transfer and generalization in human–robot interaction. For example, Gawali and Gawali [7, 21] proposed a "Transfer Expert Reinforcement Learning (TERL)" framework, which enables skill knowledge transfer between humans and robots. Their approach integrates modified RL with an improved Coyote Optimization Algorithm (BWF-COA) to tune the action features of robotic arms. By employing a deep neural network to

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

process input kinematic movements, their method facilitates the effective transfer of movement strategies. Experimental results demonstrate that such optimized skill transfer models can significantly reduce prediction errors and enhance policy rewards compared to traditional approaches, highlighting the importance of optimization-driven learning in autonomous robot control.

3. Preliminaries

3.1. Soft actor-critic

SAC [12, 13] is a modern off-policy DRL algorithm that combines policy-based and value-based methods for efficient policy learning and optimization in continuous action spaces. Its core idea is to maximize expected returns while maintaining the randomness of the policy, which helps the agent improve the efficiency of exploring the environment and find better policies. "Soft" in the algorithm refers to the introduction of the concept of entropy during training the policy, making the policy have a certain degree of randomness. This randomness helps the algorithm achieve a balance between exploration and exploitation of existing policies, enabling it to discover and utilize better policies. In addition, the SAC algorithm also employs entropy optimization methods, where the entropy term of the policy is added to the objective function to encourage exploratory behavior, effectively preventing the policy from prematurely falling into local optima.

The maximum entropy objective extends the standard objective by simultaneously maximizing the target and the entropy, as shown in Eq. (1). The SAC algorithm encourages policy exploration through maximum entropy, enabling the model to learn more near-optimal policies and ensuring the robustness of the model:

$$\pi^* = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(s_v, a_i)\rho_\pi} \left[r\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right) \right] \tag{1}$$

Among them, π^* denotes the optimal policy, π is a candidate policy, t is the time step, s_t is the state at time t, a_t is the action at time t, ρ_π represents the trajectory distribution under policy π, $r\left(s_t, a_t\right)$ is the reward function, $\mathcal{H}(\pi(\cdot \mid s)) = -\mathbb{E}_a \log \pi(a \mid s)$ represents the entropy term, and α is the temperature coefficient, which determines the importance of the entropy term in the return function; thus, it controls the degree of randomness in the optimal policy.

In the SAC algorithm, α is an important hyperparameter, which controls the degree of randomness of the strategy. The higher the α, the more inclined the strategy is to choose random actions; The lower the α, the more inclined the strategy is to choose deterministic actions. The SAC algorithm adaptively adjusts the parameters to achieve a balance between exploration and utilization.

Compared to other algorithms, SAC's advantage lies in its ability to maximize expected returns while maintaining policy randomness, which improves the algorithm's exploration efficiency in complex environments. SAC further enhances its performance and stability through techniques such as entropy optimization and adaptive temperature parameters.

3.2. Generative adversarial imitation learning

Demonstration learning is a process of learning and improving decision-making abilities by observing the behavior of demonstrators. Learning from expert demonstrations allows for a more natural and easier generalization of the expert's strategy to the current observed situation [22]. GAIL is a method of demonstration learning. Its goal is to enable agents to acquire the ability to perform tasks through expert demonstrations. The core idea of GAIL is to compare and learn the robot's own behavior and policy with the behavior and policy demonstrated by experts through a generative adversarial network (GAN), thus achieving demonstration learning.

In GAIL, the generator acts as the policy network, responsible for generating the agent's behavior, while the discriminator plays the role of a binary classifier, dedicated to distinguishing whether the input behavior data originates from demonstration data or forged data produced by the generator. The

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

generator aims to create forged data, while the discriminator strives to accurately distinguish between demonstration data and behavior data generated by the generator. Through this adversarial training process, the generator ultimately learns and generates strategies that are highly similar to the demonstration behavior.

In GAIL, the input to the generator is the state of the environment that the agent is currently in, and the output is the action that the agent should perform in the current state. The input to the discriminator is a state–action pair, that is, the behavior of an agent in a particular state. The output of the discriminator is a probability value that represents the likelihood of the input's state–action pair coming from the expert data. For real expert data, this probability value should be accepted close to 1, and for data generated by the generator, this probability value should be close to 0.

Ishaan [23] proposed gradient penalty for WGANs. GANs use $D(.;\omega)$ to represent the discriminator and use $G(.;\theta)$ to represent the generator, learning through the use of the discriminator's loss function's stochastic gradient $\tilde{g}(\theta,\omega)$ and the generator's loss function's stochastic gradient $\tilde{h}(\theta,\omega)$. When the true gradients are denoted as $g(\theta,\omega)=\nabla_{\omega}\mathcal{L}_D$ and $h(\theta,\omega)=\nabla_{\theta}\mathcal{L}_G$, these stochastic gradients can be defined as:

$$\begin{cases}\tilde{g}(\theta,\omega)=\nabla_{\omega}\mathcal{L}_D+M^{(\omega)}\\ \tilde{h}(\theta,\omega)=\nabla_{\theta}\mathcal{L}_G+M^{(\theta)}\end{cases} \tag{2}$$

In this formula, $\mathcal{L}_D$ is the loss function of the discriminator, $\mathcal{L}_G$ is the loss function of the generator, and $M^{(\omega)}$ and $M^{(\theta)}$ are random variables representing the noise introduced by stochastic gradient estimation, while $\tilde{g}(\theta,\omega)$ and $\tilde{h}(\theta,\omega)$ are stochastic approximations of the true gradients.

The goal of gradient penalty is to introduce a gradient penalty term to improve the training process of WGANs, thereby avoiding issues caused by weight clipping and enhancing the performance of the GAN. This method introduces a gradient penalty term into the loss function, making the gradient norm of the discriminator closer to 1, thus satisfying the 1-Lipschitz constraint, which enhances the stability of training and the quality of the generated samples. Specifically, the form of the gradient penalty is given as:

$$L_{\text{gp}}=\lambda\mathbb{E}\hat{x}\sim P\hat{x}\left[\left(||\nabla_{\hat{x}}f(\hat{x})||_2-1\right)^2\right] \tag{3}$$

where L_{gp} is the gradient penalty term added to the discriminator's loss function, λ is a hyperparameter controlling the strength of the gradient penalty, $P\hat{x}$ represents the linear interpolation distribution between real data and generated data, $f\left(\hat{x}\right)$ is the discriminator's score for sample $\hat{x}$, and $\nabla_{\hat{x}}f\left(\hat{x}\right)$ is the gradient of the discriminator with respect to sample $\hat{x}$.

4. Method

4.1. Algorithm improvements

In this work, we formulate the control policy for robotic manipulation as a Markov decision process (MDP) [15], where the agent observes the system state and selects actions according to a parameterized policy. The RL algorithm is employed to optimize the policy by maximizing the expected cumulative reward. This data-driven policy serves as the control law for the robot, mapping observations to actions in a closed-loop manner, functionally equivalent to a classical feedback controller [24, 25]. As in conventional control design, we view the RL-trained policy as a closed-loop feedback controller. Our policy maps the robot's state to control actions at every timestep, following the structure and operational logic of a feedback control system.

Similar to WGANs, this paper introduces a gradient penalty term into the SAC loss function. This term calculates the gradient norm of the policy network (actor) in the state space and penalizes it accordingly. By doing so, the policy network tends to maintain a certain degree of continuity during parameter updates, avoiding overly drastic changes. Meanwhile, SAC must maintain a balance between exploration and exploitation during the policy optimization process. The gradient penalty term in WGAN-GP has

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

been shown to help maintain diversity in the generated samples by encouraging smooth and continuous changes in the output [23]. Inspired by this, introducing a gradient penalty into the SAC framework can likewise encourage diversity in the action space and improve training stability. Therefore, introducing the concept of the gradient penalty into SAC can encourage the policy network to generate actions that maintain diversity in the action space, preventing premature convergence to local optimal solutions.

In this study, the gradient penalty is applied to the loss calculation for the policy network. The update rule for the policy network (actor) is given as:

$$\theta_{n+1} = \theta_n + a\,(n)\,(s_n, a_n, \theta_n)\,d\,(s_n, s_{n+1}, \omega_n) \tag{4}$$

In this formula, θ_n is the parameter vector of the actor network at iteration n, $a\,(n)$ is the learning rate at iteration n, $d\,(s_n, s_{n+1}, \omega_n)$ is the temporal-difference (TD) error or advantage function, capturing the update direction, e_n is an eligibility trace or scaling factor associated with the sample at iteration, s_n is the observed state at iteration n, a_n is the chosen action at iteration n, and the likelihood ratio derivative $\Psi \in \mathbb{R}^m\theta$ is defined as:

$$\Psi(s, a, \theta) = \frac{\nabla_\theta \pi(a \mid s)}{\pi(a \mid s)} \tag{5}$$

In this formula, $\pi(a \mid s)$ is the policy's probability of selecting action a in state s, parameterized by θ, and $\nabla_\theta \pi(a \mid s)$ is the gradient of the policy with respect to its parameters θ.

In the SAC algorithm, the original loss function for the policy network typically consists of a policy gradient loss term and an entropy regularization term. Now, a gradient penalty term is added to obtain the new loss function. Assuming the policy network π has parameters denoted by θ, the gradient of the policy network's output actions with respect to the state can be calculated as:

$$\nabla_a \log \pi(a \mid s; \theta) \tag{6}$$

where $\pi(a \mid s; \theta)$ is the policy network (actor), which outputs the probability of taking action a given state s, parameterized by θ.

Then, the *L2* norm of this gradient is calculated, and a target norm r is defined. The gradient penalty term can be expressed as:

$$\lambda \mathbb{E}_{s\sim p, a\sim\pi}\left[(|\nabla a \log \pi(a \mid s; \theta)|_2 - r)^2\right] \tag{7}$$

In this equation, λ is the gradient penalty coefficient, a hyperparameter that controls the strength of the penalty term; $E_{s\sim p, a\sim\pi}$ is expectation over states s sampled from the state distribution p and actions a sampled from the policy π; and $\nabla a \log \pi(a \mid s; \theta)$ is the gradient of the log-probability of action a with respect to a, given state s and parameters θ.

Finally, by adding the gradient penalty term to the original loss function of the actor network, we obtain the new loss function:

$$L_{\text{actor}}^{\text{with GP}} = L_{\text{actor}} + \lambda \mathbb{E}_{s\sim \rho, a\sim\pi}\left[(|\nabla_a \log \pi(a \mid s; \theta)|_2 - r)^2\right] \tag{8}$$

This new loss function combines the policy gradient loss term, entropy regularization term, and gradient penalty term from the SAC algorithm. By introducing the gradient penalty term, the following advantages are achieved:

1. Improved policy stability: Gradient penalty encourages the actions generated by the policy network to maintain diversity in the action space, preventing the policy from falling into local optima. This allows the agent to better adapt to environmental changes and improve task success rates.
2. Enhanced exploration capabilities: The SAC algorithm inherently encourages exploration, and the addition of the gradient penalty term further enhances this ability. Through the gradient

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

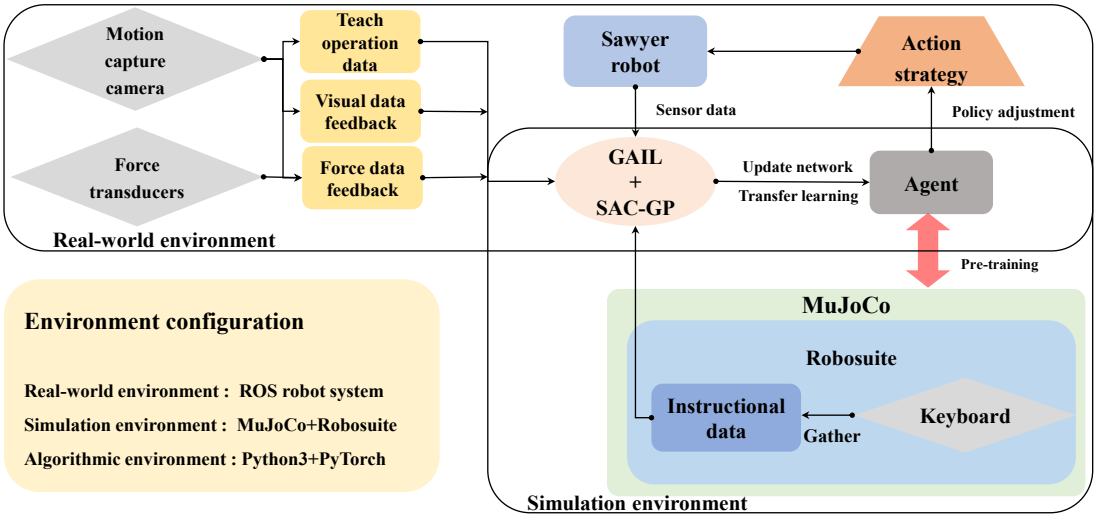


Figure 1. *The composition of a robotic arm self-learning system.*

penalty term, the algorithm can better balance exploration and exploitation strategies, thereby finding the optimal policy more quickly.

3. Increased training efficiency: The inclusion of the gradient penalty term helps reduce instability and fluctuations during the training process, further improving training efficiency.
4. Improved task completion quality: By optimizing the loss function, errors during task completion can be reduced, thereby enhancing the efficiency and quality of task completion.

This study integrates the GAIL algorithm into the SAC-GP algorithm. In GAIL, the generator attempts to optimize its policy to deceive the discriminator, whereas in SAC-GP, the policy network optimizes by maximizing the expected return and entropy. This research combines the adversarial loss from GAIL with the policy optimization loss from SAC-GP, which consists of maximizing the expected return and entropy regularization, to form a hybrid loss function. This loss function balances the two objectives, enabling the robotic arm learning system to perform imitation learning while maintaining sufficient exploration.

4.2. Construction of a self-learning system for robotic arms

The robotic arm learning system and its components built in this study are shown in Figure 1.

The real-world experimental platform is built on the robot operating system (ROS), which integrates the control of the Sawyer robotic arm, the AG-95 gripper, and data acquisition from a NOKOV optical motion capture system and a six-axis Srisensor force/torque sensor. All algorithms are implemented in Python 3 with the PyTorch library. To thoroughly validate the effectiveness of the proposed GAIL + SAC-GP algorithm, both a virtual simulation environment and a real-world physical platform are constructed. The simulation environment is developed using the MuJoCo physics engine [26] and the RoboSuite framework [27]. MuJoCo is widely used for simulating complex robot–environment interactions, while RoboSuite provides various robot tasks and user-friendly interfaces for robot learning research. In our workflow, the agent is first trained in the virtual environment, where it optimizes its policy through iterative trial-and-error learning. Once the policy converges, we apply Sim2Real (Simulation to Reality) transfer [28–32] to directly deploy the trained agent on the real hardware without further real-world training, thereby enabling immediate evaluation and rapid deployment. Ensuring high consistency between simulation and reality is critical in this process. The following sections further elaborate on the construction of both environments and the corresponding training/evaluation processes.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

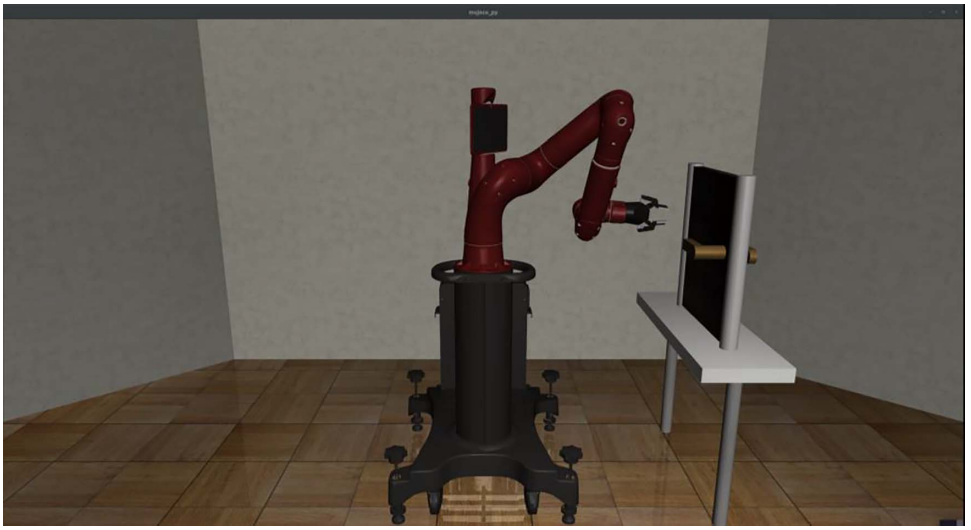


Figure 2. *Construction of a virtual environment for door handle task.*

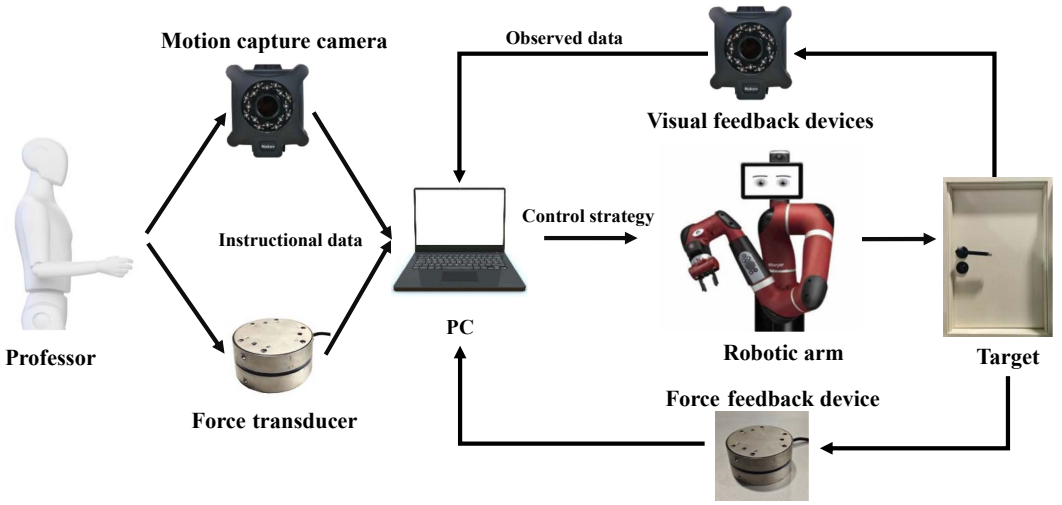


Figure 3. *The basic composition of the learning system platform for a robot.*

4.3. Environment setup

The construction of the virtual environment is based on the RoboSuite robot simulation framework. The setup task scenario is illustrated in Figure 2.

The hardware platform of the robotic arm learning system mainly involves a computing platform, a robotic arm, an electric gripper, visual feedback equipment, and force feedback equipment. To realize task scenarios in a real environment, this study has built a learning system platform for the robotic arm, as shown in Figure 3.

In the hardware composition of this robotic system, the visual feedback equipment uses the NOKOV optical motion capture system, the force feedback equipment employs the SRI six-axis force sensor, the computing platform is a desktop PC, the robotic arm adopts the Sawyer collaborative robot from Rethink Robotics, and the gripper uses the DAHUAN AG-95 series gripper. Among them, the computing platform's CPU is an AMD 2970WX with 24 cores and 48 threads and a base frequency of 3.0 GHz. The computer's memory is 128 GB, and the graphics card model is the NVIDIA RTX 3,090.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table I. *The IP address of the hardware device.*

Category	Equipment	IP
PC	Switch	192.168.1.100
	NOKOV motion capture camera	192.168.3.6
	Actuators	192.168.0.200
NOKOV motion capture camera	Camera	10.1.1.201 10.1.1.215
	PC	10.1.1.198/192.168.3.6
Actuators	Dahuan AG-95 gripper	192.168.1.29
	Sawyer robot	169.192.2.60
Six-dimensional force transducers	SRI	192.168.0.100

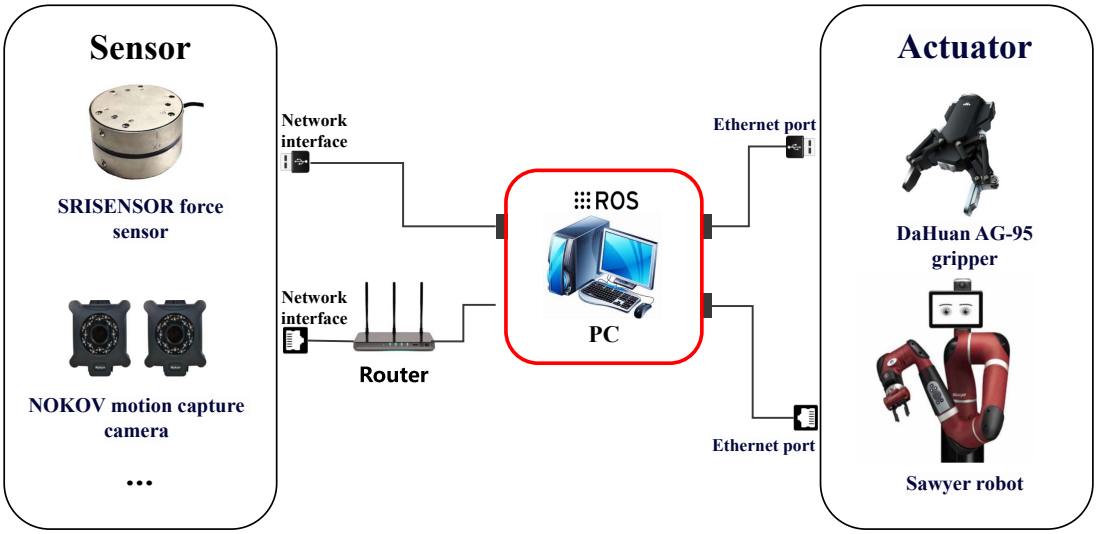


Figure 4. *Control system hardware composition.*

This article chooses to build the software platform under the Ubuntu 18.04 operating system, using ROS Melodic to construct the control system and implement data communication functions. The hardware composition of this control system is illustrated in Figure 4. The IP addresses of each device are detailed in Table I.

This study first trains agents in a virtual simulation environment and then deploys the trained agents to the real environment through Sim2Real. In Sim2Real, it is necessary to ensure a high degree of consistency between the simulation environment and the real environment. Therefore, the real environment interface is strictly written according to the interface format in RoboSuite, making the interfaces of the virtual simulation environment and the real environment consistent. In addition, this paper also unifies the spatial coordinate system and spatial units in the virtual and real environments. Ultimately, the environmental state information dimension and action space dimension in the real environment are kept consistent with those in the simulation environment to satisfy the requirements of Sim2Real.

4.4. Multi-source sensor information fusion

The mechanical arm learning method studied in this paper utilizes two types of sensors: NOKOV motion capture cameras and SRI six-dimensional force sensors. To fully leverage the advantages of these two sensors, it is necessary to effectively fuse the information provided by them, thus providing more

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

comprehensive and accurate information. Each sensor has its own unique measurement capabilities and limitations, and through fusion, errors or biases that may exist in a single sensor can be eliminated, improving the reliability of the overall data. Based on the importance of the two modal information, this paper actively adjusts their advantages in the fused features and performs a weighted fusion of spatial position and posture features and force perception features to obtain the final multi-modal fused features. Through the fusion of multi-source sensor information, the fused features can better represent the environmental state of the mechanical arm, providing richer and more accurate information for subsequent learning and decision-making. The following introduces how to fuse these two modal sensor information in detail, combined with the mechanical arm learning algorithm.

This paper adopts a method combining DRL and imitation learning for training. In RL, through the setting of a reasonable reward function, the agent can be guided to effectively learn and optimize its behavior. This study decomposes the complex task of the mechanical arm opening a door into three subtasks: grasping the door handle, rotating the door handle, and pulling the door handle. For these three subtasks, this paper respectively introduces how the multi-source sensor information is fused in the setting of the reward function and guides the mechanical arm to optimize its behavior.

4.4.1. The design of the reward function of the robotic arm grabbing the door

For this task, special attention is paid to the position and orientation of the mechanical arm's gripper end and the door handle. By considering the positional distance and orientation between the mechanical arm's end gripper and the door handle, the mechanical arm is guided to approach the door handle for grasping. The reward function is defined as:

$$reward = 1.5 * (1 - np.\tanh(10.0 * dist)) \tag{9}$$

In this equation, *dist* represents the Euclidean distance between the center of the gripper's end and the door handle.

4.4.2. The design of the reward function of the robotic arm revolving door handle

For this task, the focus is on the position and orientation of the mechanical arm's gripper end as well as the force applied to it, taking into account both the positional distance between the gripper and the door handle and the force exerted by the gripper's end. This information guides the mechanical arm to rotate the door handle and subsequently unlock the door. The reward function for position and orientation is defined as:

$$pos_reward = 0.25 * np.abs(handle_qpos/(0.5 * np.pi)) \tag{10}$$

In this equation, *handle_qpos* represents the rotation angle of the door handle.

Force sense reward is defined as:

$$tor_reward = 1.0 * ((fz - base_f)/target_f) * *exponent \tag{11}$$

In this equation, fz is the force on the z-axis captured by the force sensor, $base_f$ is the force in the initial state of the z-axis, $target_f$ is the force in the target state of the z-axis, and *exponent* is the exponential value.

Force sense rewards use an exponential reward function so that the agent's reward increases faster as it approaches the target. The reward function for this task is as follows:

$$reward = pos_reward + 0.5 * tor_reward \tag{12}$$

4.4.3. The design of the reward function of the robotic arm pulling the door handle

For this task, the focus is on the position and orientation of the mechanical arm's gripper end, the position of the door handle, and the force exerted by the gripper's end. By comparing the force applied to the door and the gripper's end, as well as their positional relationship, the mechanical arm is guided to pull the

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table II. *Reward terms for pretraining.*

Stage	Rewards and penalty	Weight
Universal rewards and penalties	DoF velocity limits	−5
	DoF position limits	−10
	Torque limits	−5
	DoF velocity limits	−2
	Termination	−50
	DoF postion	0.5
Grab handle	Distance	2
Revolve handle	Handle pos	1
	When closed DoF torque	0.5
Pull handle	Handle pos	1
	When closedtorque	0.5

door handle, thereby opening the door. The reward function for position and orientation is defined as:

$$pos_reward = 1.5 * (1 - np.\tanh(10.0 * dist)) \tag{13}$$

In this equation, *dist* represents the distance of the door's movement on the *x*-axis.

Force sense reward is defined as:

$$tor_reward = 1.0 * ((fx - base_f)/target_f) * *exponent \tag{14}$$

In this equation, *fx* is the force captured by the force sensor on the *x*-axis, *base_f* is the force in the initial state on the *x*-axis, *target_f* is the target force on the *x*-axis, and *exponent* is the exponential value.

The reward function for this task is defined as:

$$reward = pos_reward + 0.5 * tor_reward \tag{15}$$

The above describes the special reward function design of the manipulator under different tasks. To guide the robotic agent towards safe and efficient task execution, we designed a set of universal and task-specific rewards and penalties, as summarized in Table II. Universal penalties are imposed for exceeding velocity, position, and torque limits of the robot joints (DoF: degree of freedom). For instance, exceeding the DoF position limit incurs a penalty of −10 per occurrence, while exceeding velocity or torque limits results in penalties of −5 and −2, respectively. These terms ensure the manipulator operates within safe mechanical bounds. Termination penalty: a significant negative reward of −50 is assigned for episode termination due to one of the following events: (1) task timeout (the agent does not complete the task within the allowed step limit), (2) collision with the environment (e.g., door, frame), and (3) manual emergency stop triggered for safety. Minimal positive reward (e.g., +0.5) is given for simply maintaining joint positions within safe ranges, which encourages stable control even when not making dramatic progress. The magnitude of each reward/penalty term has been empirically tuned to balance task completion, motion safety, and learning efficiency. The combination of these terms ensures the policy learns to maximize task progress while strictly avoiding unsafe behaviors.

5. Experiments

5.1. Simulation experiment of robotic arm lifting objects

In the virtual simulation experiments, we adopted three main performance parameters to evaluate and compare the proposed SAC-GP algorithm with the baseline SAC algorithm in the block grasping task: (1) convergence speed: defined as the number of training episodes required for the model's reward curve to stabilize and reach a plateau; (2) average episode reward: the mean final reward achieved per episode, reflecting the policy's effectiveness in completing the task and adhering to safety or efficiency

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table III. *Hyperparameters of the "Lift" simulation experiment algorithm.*

Target steps	Temperature coefficient(γ)	Network width	Learning rate
1,000	0.97	2^7	2^{-13}

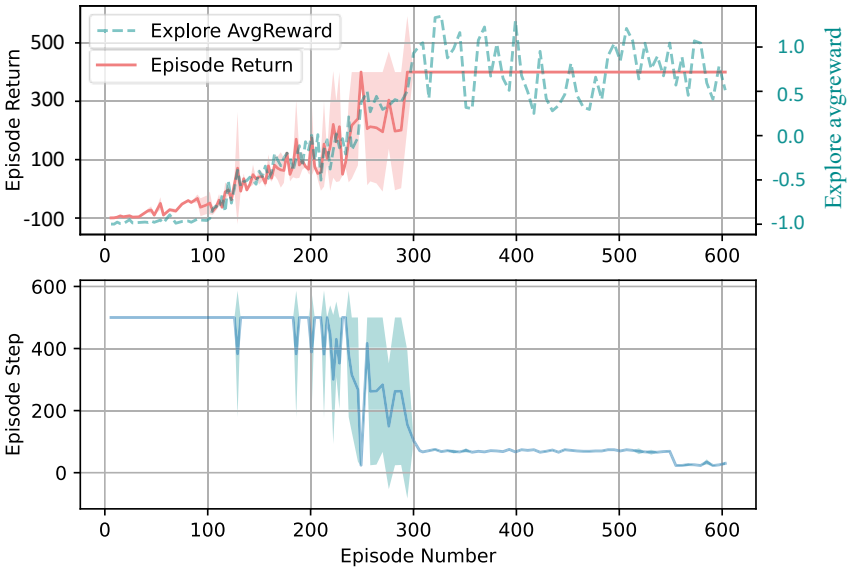


Figure 5. *The training results of the block lifting experiment of the SAC-GP algorithm.*

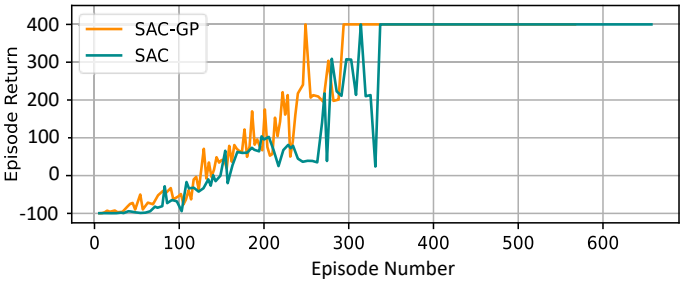


Figure 6. *Comparison of the SAC-GP algorithm and the SAC algorithm in the "Lift" experiment.*

objectives; and (3) robustness (reward fluctuation): quantified by the amplitude and frequency of reward fluctuations throughout training, indicative of the algorithm's stability.

First, in the virtual environment, the block grasping experiment of the robotic arm was carried out to compare the SAC algorithm and the SAC-GP algorithm. The hyperparameters of the algorithms are shown in Table III.

Figure 5 shows the training reward curve using the SAC-GP algorithm, and Figure 6 presents a direct comparison between SAC-GP and SAC. As can be seen, the SAC-GP algorithm achieves faster convergence, with the model stabilizing in about 300 episodes, whereas SAC requires approximately 330 episodes to converge. Additionally, the average episode reward attained by SAC-GP is consistently higher than that of SAC. Regarding robustness, the reward curve of SAC-GP exhibits notably fewer and smaller fluctuations compared to that of SAC, indicating greater training stability and robustness of the learned policy.

These results demonstrate that the proposed SAC-GP algorithm not only achieves higher learning efficiency but also exhibits superior effectiveness and robustness compared to the baseline SAC algorithm.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table IV. *Hyperparameters of the "Door" simulation experiment algorithm.*

Target steps	Temperature coefficient(γ)	Network width	Learning rate
1,000	0.95	2^7	2^{-13}

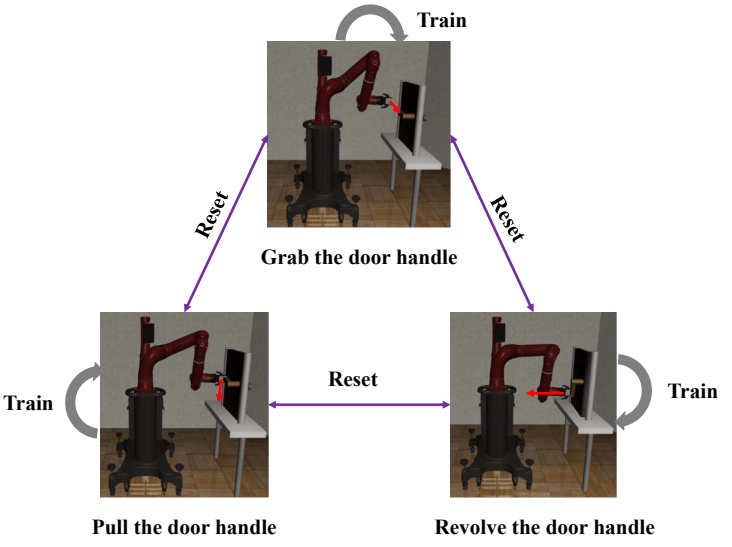


Figure 7. *Subtask coupling relationship.*

5.2. Simulation experiment of robotic arm opening the door

The experiment introduces a hierarchical RL approach [33–36] to slice the complex task of robotic door-opening. Specifically, the door-opening task is divided into three subtasks: the first subtask is for the robotic arm to grasp the door handle, the second subtask is for the robotic arm to rotate the door handle, and the third task is for the robotic arm to pull the door handle. The adjusted hyperparameters for this experiment are shown in Table IV.

Furthermore, this study also employs training acceleration techniques based on hierarchical RL [37], achieving fully automatic scene initialization without manual intervention, thus speeding up the training process. To address the coupling relationship between subtasks, this paper defines the scene transition relationship for subtask training, as illustrated in Figure 7. The training starts with grasping the door handle. If the door handle is not successfully grasped, the training will continue with grasping the door handle; if the door handle is successfully grasped, the training will proceed to rotating the door handle; if the door handle is successfully rotated open, the training will then move on to pulling the door open. This process continues until the robotic arm successfully completes three subtasks, which are grabbing the door handle, rotating the door handle, and opening the door.

To quantitatively evaluate the performance of the proposed system, the following performance parameters are used: success rate: the proportion of episodes in which the robot successfully completes three subtasks and opens the door; average completion time: the average number of time steps or real time required to complete the door-opening task from start to finish; average reward: the mean cumulative reward accumulated by the agent per episode, reflecting both task efficiency and behavior quality; convergence episodes: the number of training episodes required for the policy to converge to a stable or near-optimal success rate; and robustness: the consistency of task completion under environmental perturbations or variations, such as noise, different initial positions, or changing door properties. These metrics provide a comprehensive assessment of the effectiveness, efficiency, and reliability of the proposed hierarchical RL approach for robotic door-opening.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

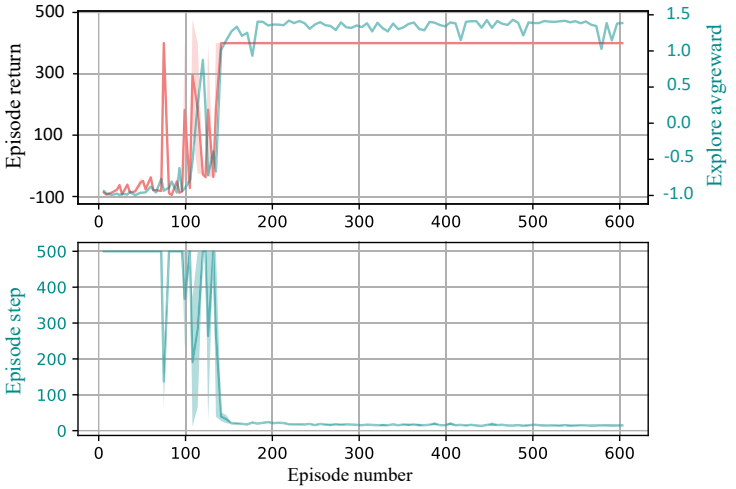


Figure 8. *The reward curve obtained by the SAC-GP algorithm in the simulation experiment of the robotic arm grabbing the door handle.*

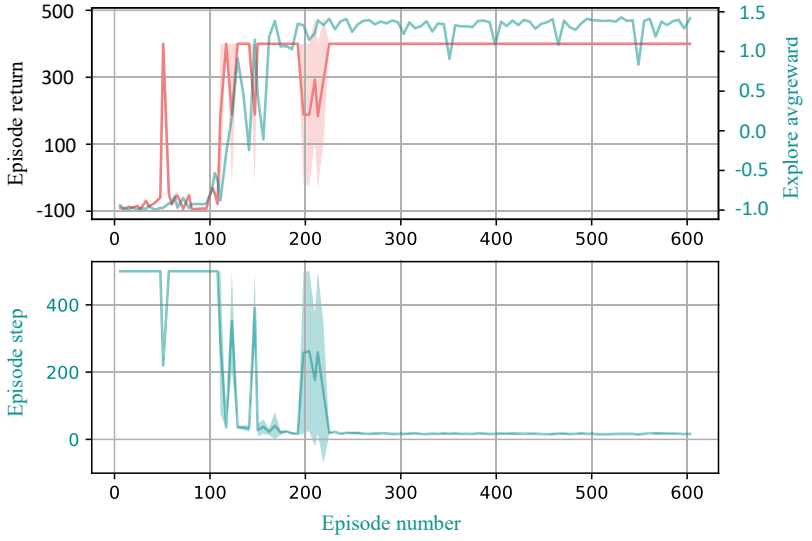


Figure 9. *The reward curve obtained by the SAC algorithm in the simulation experiment of the robotic arm grabbing the door handle.*

5.2.1. Simulation experiment of robotic arm grabbing the door handle

Using the hyperparameters listed in Table IV and the SAC-GP algorithm, we conducted training on the door-grasping task. The change in reward and the number of steps per episode are shown in Figure 8. Similarly, training results with the original SAC algorithm are shown in Figure 9.

Comparison results are summarized as follows: Convergence speed: The SAC-GP algorithm enables the agent to converge to an effective policy in approximately 130 episodes, while the original SAC algorithm requires around 210 episodes to achieve a similar level of performance, indicating faster learning with SAC-GP. Stability: In the stable phase, the reward curve of the SAC-GP agent exhibits 10 significant fluctuations, whereas the SAC agent's reward curve has 12. This demonstrates improved stability with SAC-GP. Overall performance: The SAC-GP agent attains a higher average reward per episode and a higher final success rate, showing improved effectiveness for the door-opening task. These performance

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

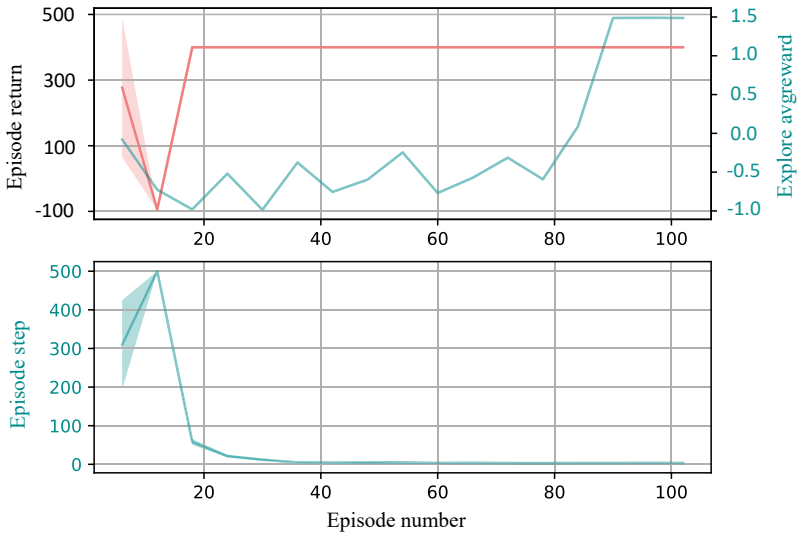


Figure 10. *The reward curve obtained by using the SAC-GP algorithm in the simulation experiment of the robotic arm revolving the door handle.*

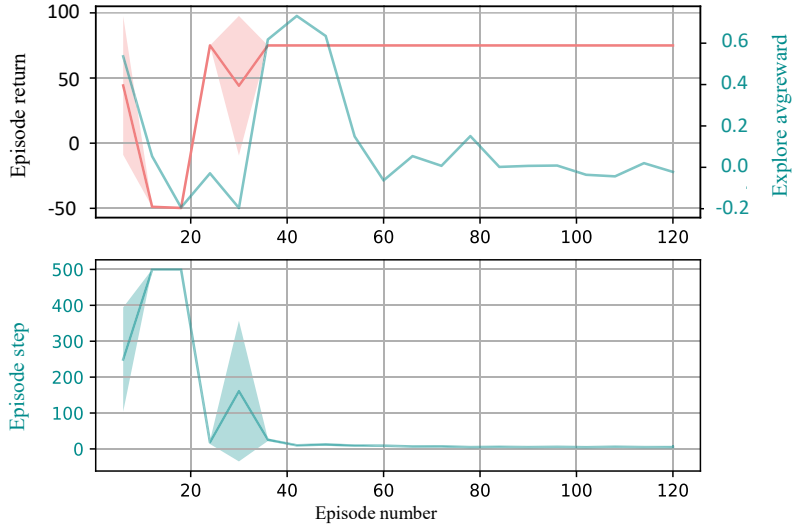


Figure 11. *The reward curve obtained by using the SAC algorithm in the simulation experiment of the robotic arm rotating the doorknob.*

parameters comprehensively demonstrate that the SAC-GP algorithm provides superior efficiency and stability compared to the classical SAC algorithm in robotic door-opening tasks.

5.2.2. Simulation experiment of robotic arm revolving the door handle

Later, the revolving door handle was trained, and the change in reward value and round steps was shown in Figure 10. When training with the SAC algorithm, the changes in reward values and episode steps are illustrated in Figure 11.

Based on the predefined performance parameters, a comparative analysis between the SAC-GP and SAC algorithms yields the following insights: The SAC-GP algorithm achieves policy convergence in approximately 20 episodes, whereas the SAC algorithm requires about 40 episodes. This highlights the superior learning efficiency of SAC-GP. Stability: The reward curve of SAC-GP shows fewer

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

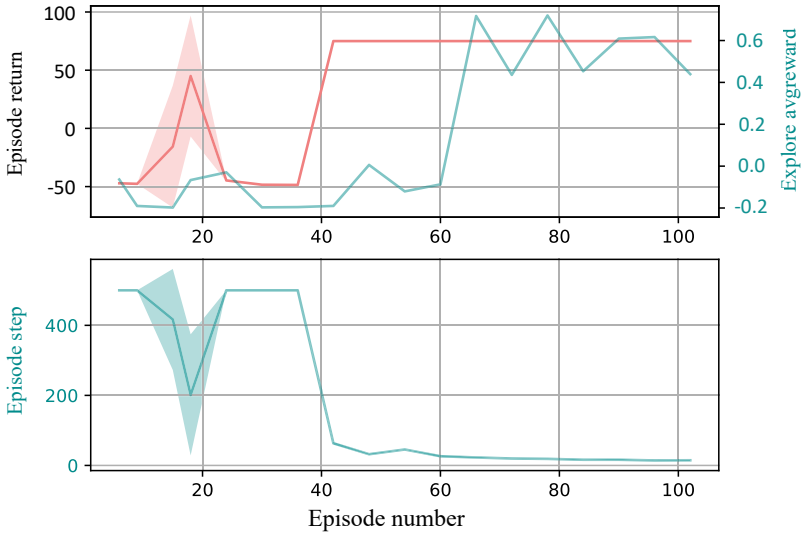


Figure 12. *The reward curve obtained by using the SAC-GP algorithm in the simulation experiment of the robotic arm pulling the doorknob.*

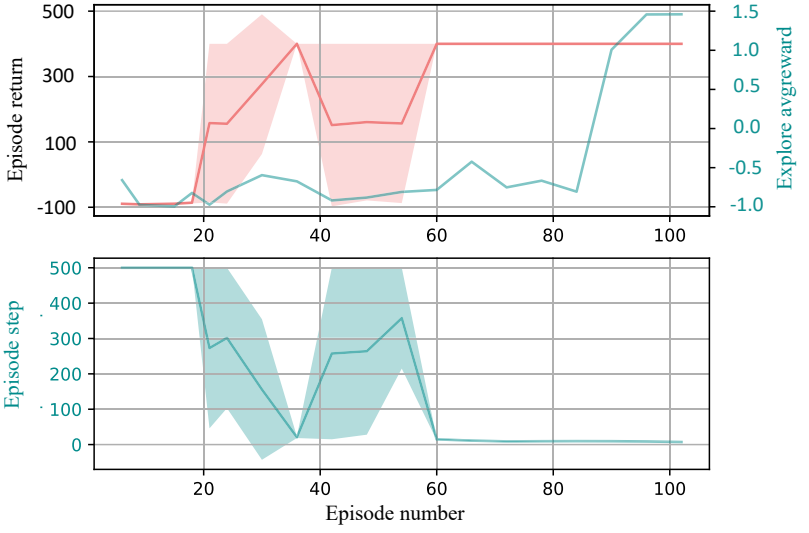


Figure 13. *The reward curve obtained by using the SAC algorithm in the simulation experiment of the robotic arm pulling the door handle.*

significant fluctuations compared to that of SAC, indicating improved training stability and more consistent policy performance. Average reward: After convergence, the average cumulative reward per episode achieved by the SAC-GP algorithm is higher than that of the SAC algorithm, reflecting better overall task performance. In summary, with respect to the defined performance parameters – convergence speed, stability, and average reward – the SAC-GP algorithm demonstrates clear advantages over the original SAC algorithm for the robotic door-opening task.

5.2.3. Simulation experiment of robotic arm pulling the door handle

Finally, the change in reward values and episode steps during the door handle pulling training is shown in Figure 12. For comparison, the training results using the SAC algorithm are illustrated in Figure 13.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

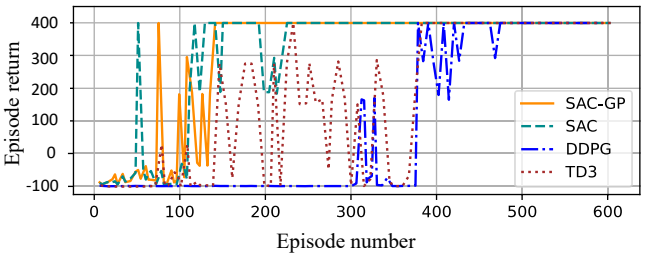


Figure 14. *Comparison of the reward curves of RL algorithm in the experiment of grabbing doorknobs.*

Based on the predefined performance parameters, a comparative analysis between the SAC-GP and SAC algorithms yields the following conclusions: the SAC-GP algorithm enables the agent to converge to an effective policy in approximately 40 episodes, whereas the SAC algorithm requires about 60 episodes to achieve a similar level of performance. This demonstrates the faster learning capability of SAC-GP. Stability: During training, the reward curve of the SAC-GP algorithm exhibits only two significant fluctuations, compared to four fluctuations observed with the SAC algorithm. This indicates that the SAC-GP algorithm achieves more stable training and policy performance. In summary, with respect to convergence speed and stability, the SAC-GP algorithm again demonstrates clear advantages over the original SAC algorithm for the door handle pulling subtask.

Across all three subtasks, the SAC-GP algorithm consistently achieves a better balance between exploration and exploitation, resulting in faster and more effective policy learning. Additionally, it demonstrates greater training stability, as evidenced by fewer significant fluctuations in the reward curves compared to the SAC algorithm. Collectively, these results demonstrate that SAC-GP outperforms the standard SAC algorithm in terms of both learning efficiency and stability for the robotic door-opening task. These advantages, reflected in the key performance metrics of convergence episodes and reward stability, confirm the overall superiority of the SAC-GP algorithm across all subtasks.

The SAC-GP algorithm proposed in this paper aims to address specific issues in the field of RL. To further evaluate the performance of the SAC-GP algorithm, comparative experiments were conducted with popular algorithms such as SAC, deep deterministic policy gradient (DDPG), and twin-delayed DDPG (TD3). These algorithms were used to perform door handle grasping experiments, and the experimental results are shown.

The SAC-GP algorithm proposed in this paper aims to address specific challenges in RL. To further evaluate its performance, comparative experiments were conducted against several popular algorithms, including SAC, DDPG, and TD3. These algorithms were each applied to the door handle grasping task, and the corresponding experimental results are presented in Figure 14. As shown in the figure, the SAC-GP algorithm not only achieves faster convergence to higher reward values during training but also maintains stable performance during testing. This suggests that SAC-GP achieves a more effective balance between exploration and exploitation, resulting in improved learning efficiency and performance. Overall, the experimental results on multiple benchmark tasks consistently demonstrate the superior performance of the SAC-GP algorithm. The comparative evaluation with other state-of-the-art algorithms further confirms its effectiveness and practical value in RL applications.

In addition, experiments were conducted in the MuJoCo + RoboSuite simulation environment, where a robotic arm was tasked with grasping a door handle. The reward curves for the GAIL + SAC-GP algorithm and the SAC-GP algorithm are shown in Figure 15 for comparison. From these results, it is evident that combining the GAIL algorithm with SAC-GP significantly accelerates the convergence of training and achieves greater stability compared to using SAC-GP alone. This demonstrates the enhanced effectiveness of the GAIL + SAC-GP algorithm for the robotic door handle grasping task.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

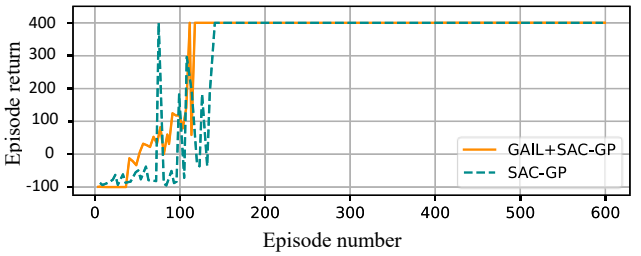


Figure 15. *Comparison of reward curves between GAIL + SAC-GP algorithm and SAC-GP algorithm.*

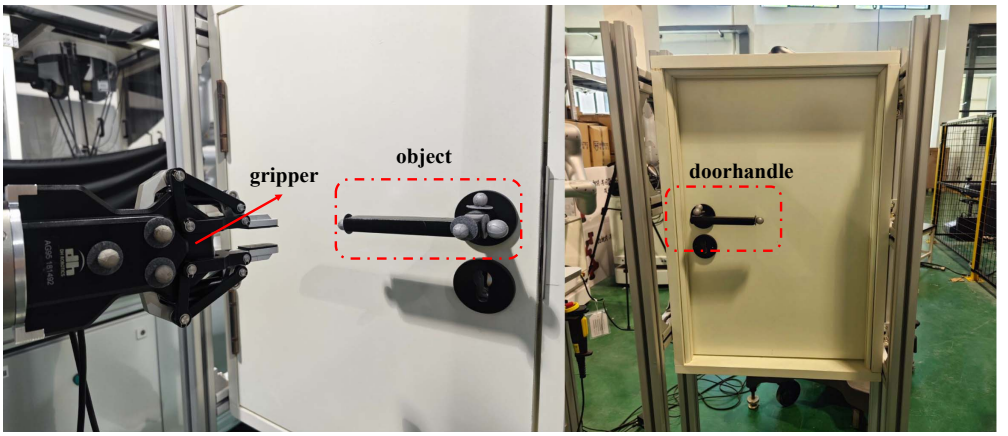


Figure 16. *Position of attaching reflective targets such as grippers and door handles.*

5.3. Experiments in a real-world environment

In the field of robotics, it is a highly effective method to first train agents in a virtual environment and then apply them to real-world environments through Sim2Real. The control strategies learned by the agent in the virtual environment can be directly applied to the real-world environment or fine-tuned in the real-world environment to adapt to environmental differences.

In this study, the agent was first pretrained in a virtual environment. Then, through Sim2Real [28–32], the policy network trained in the virtual environment was transplanted to the real-world environment, followed by fine-tuning and optimization of the transferred policy network.

Rigid body models such as grippers and door handles were created in the motion capture software, with the positions of the reflective markers attached as shown in Figure 16. The capture effect within the motion capture software is illustrated in Figure 17.

5.3.1. Robot arm grasping door handle experiment

A total of 30 sets of door handle grasping experiments were conducted to test the reliability of the trained model, and the operation success rate was calculated as shown in Table V.

In the 30 sets of robot arm grasping door handle experiments, the robot arm was initialized to different positions within the operational space, and the grasping success rate was 97%. Among the failed cases, the claw knocked off the reflective target ball on the door handle, resulting in a failed grasp. Therefore, it can be concluded that the experiment has achieved relatively satisfactory results.

One set of successful experimental data was selected for result analysis, and the actual trajectory of the end point position was collected, as shown in Figure 18.

5.3.2. Robotic arm revolving door handle experiment

A total of 30 sets of rotating door handle experiments were conducted to test the reliability of the model. The task success rate was counted and calculated, as shown in Table VI.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table V. *Statistics of the success rate of the robotic arm grabbing the doorknob.*

Turns	Successful turns	Success rate
30	29	96.7%

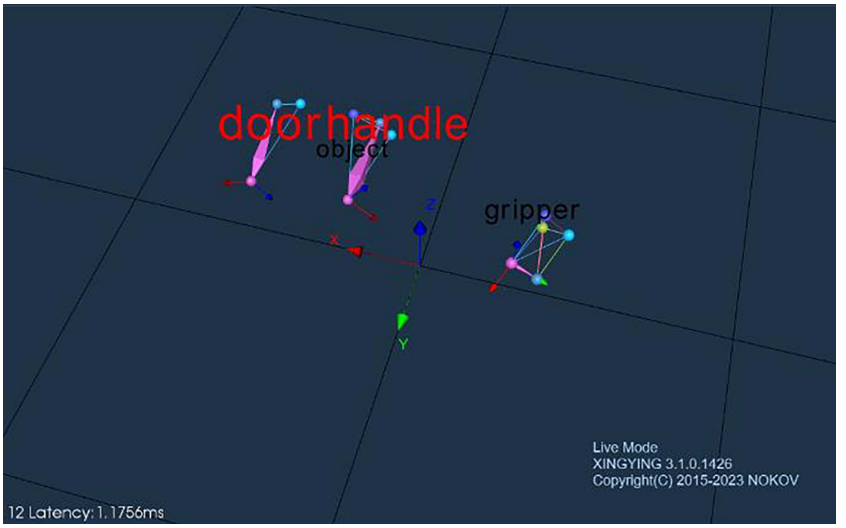


Figure 17. *Software recognition result under the motion capture camera.*

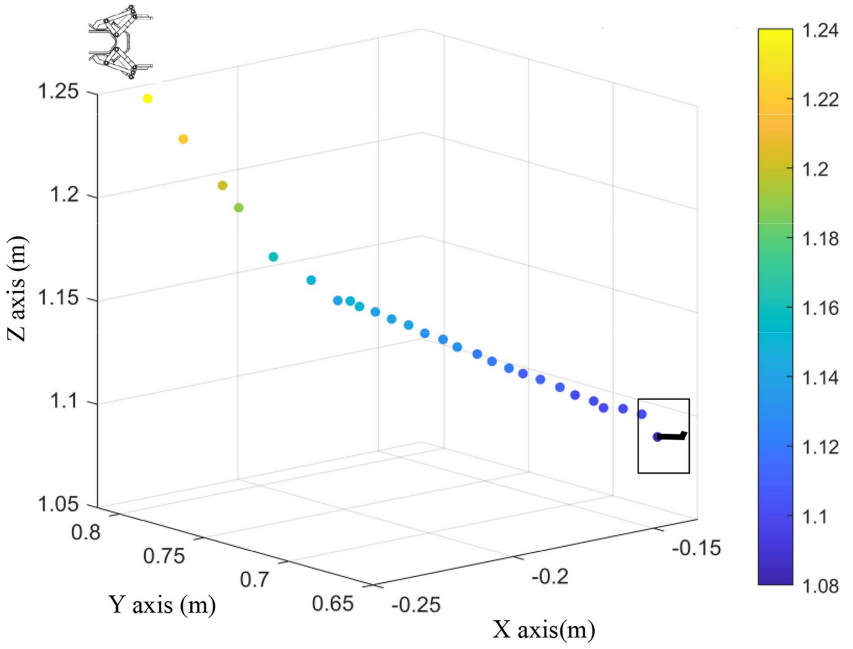


Figure 18. *The actual running track of the end point of the manipulator grabbing the door handle.*

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table VI. *Statistics of the success rate of robotic arm revolving door handles.*

Turns	Successful turns	Success rate
30	28	93.3%

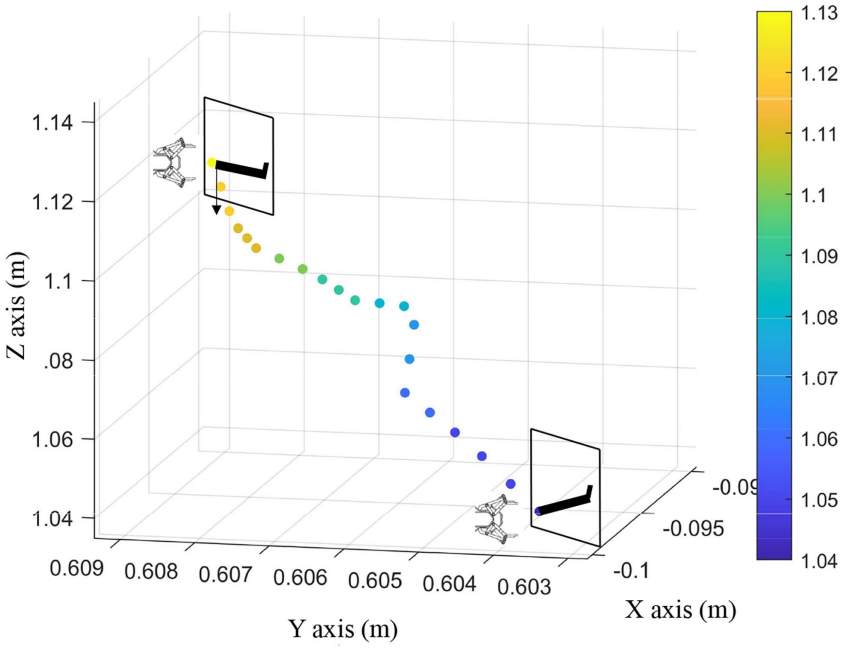


Figure 19. *Trajectory of the end-effector of the robotic arm in the X Y Z spatial coordinate system.*

In the 30 sets of robot arm rotating door handle experiments, the pose after successfully grasping the door handle from different positions within the operational space was used as the initial state for each set of experiments. The success rate of rotating the door handle reached 93.3%, achieving good results. Among the failed cases, one was due to the claw detaching from the door handle, and the other was caused by the claw not rotating the door handle to a sufficient angle to fully unlock the lock. Overall, the experiment has achieved relatively satisfactory results.

One set of model test data was selected for analysis, and the actual trajectory of the target object was obtained, as shown in Figures 19 and 20.

5.3.3. Robotic arm pulling door handle experiment

A total of 30 sets of door handle pulling experiments were conducted to test the reliability of the model. The task success rate was counted and calculated, as shown in Table VII.

In the 30 sets of experiments on door handle pulling using a robotic arm, each set of experiments used different positions in the operating space of the robotic arm to grasp the door handle and rotate the door handle's pose as the initial state. The success rate of pulling the door open reached 96.7%, achieving good results. Among them, the cases of failure were as follows: the gripper detached from the door handle during the process of pulling the door handle, but the door was already open. As shown in Table VII, the experiment has achieved relatively ideal results.

A set of model test data was selected for analysis to obtain the actual running trajectory of the target object, as shown in Figures 21 and 22.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Table VII. *Statistics on the success rate of the manipulator pulling door handles.*

Turns	Successful turns	Success rate
30	29	96.7%

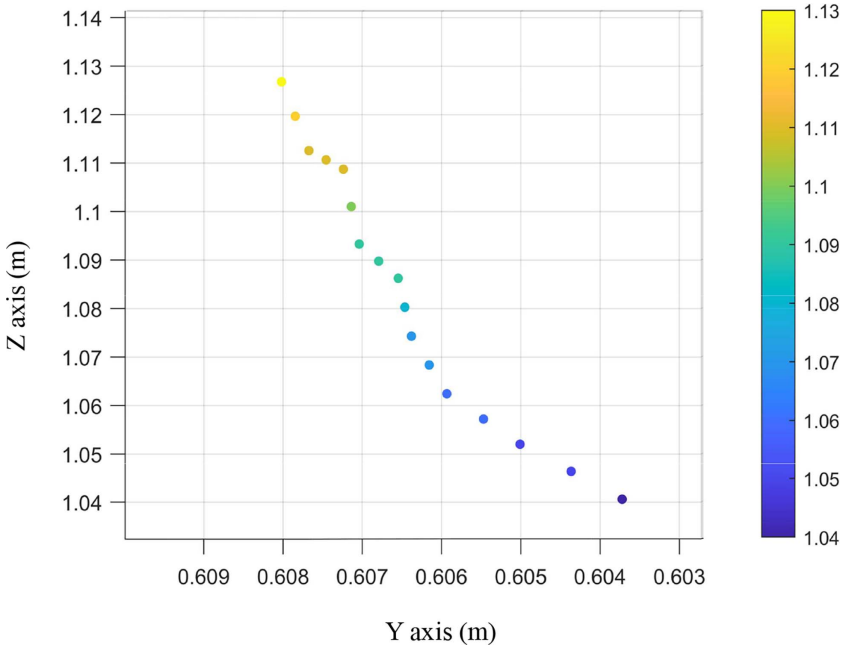


Figure 20. *Trajectory of the end-effector of the robotic arm within the X Y plane.*

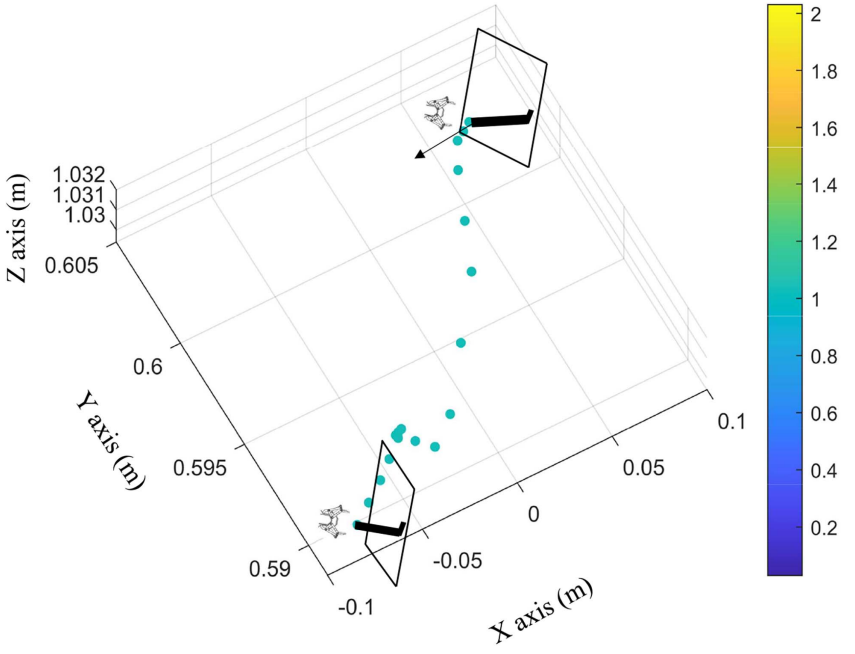


Figure 21. *Trajectory of the end-effector of the robotic arm in the X Y Z spatial coordinate system.*

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

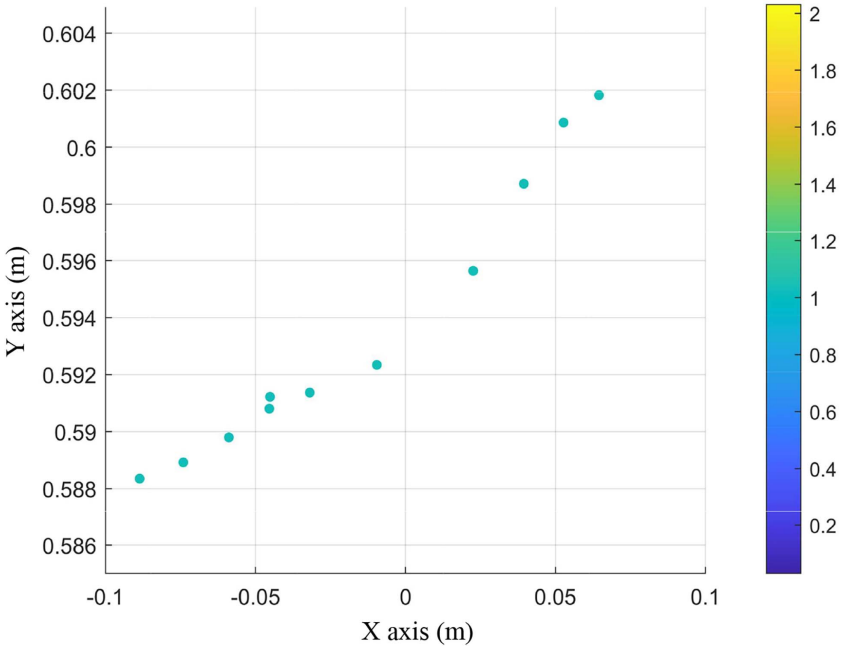


Figure 22. *Trajectory of the end-effector of the robotic arm within the X Y plane.*

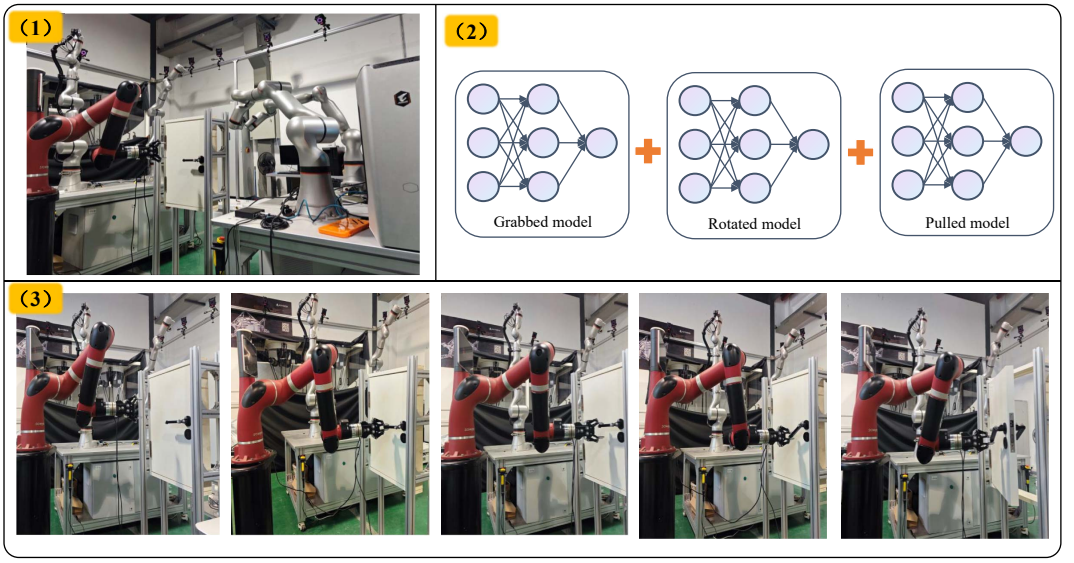


Figure 23. *Experimental process of the robotic arm door-opening task.*

5.3.4. Robotic arm door-opening experiment

Based on the tasks of "grasping the door handle," "rotating the door handle," and "pulling the door handle," this experiment conducted a "robotic arm door opening" experiment. The models trained from the first three tasks were used as "basic action" models for "skill stitching," enabling the agent to complete the door-opening task in its entirety, as shown in Figure 23.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

6. Conclusion

This paper designs and constructs a robotic arm learning system based on multi-source sensor information fusion. By integrating multi-source sensor information and combining demonstration learning with DRL algorithms, the system has successfully transformed human teaching data into empirical strategies for the robotic arm to perform tasks. It has also successfully transferred the agent trained in a virtual environment to the real environment through Sim2Real and completed predefined, complex tasks in the real environment. The main research achievements of this paper are reflected in: robotic arm task modeling and algorithm framework selection; establishing simulation experiments and real experimental environments; implementation, optimization, and experimentation of robotic arm learning algorithms in simulation environments; migrating the well-trained agent from simulation experiments to real environments; and conducting experimental verification.

While this paper has achieved certain results in DRL, demonstration learning, and Sim2Real, there are still several aspects that require further exploration and in-depth research:

1. Algorithm optimization and performance improvement: Although the current system can already complete complex tasks, future research can further improve and optimize the algorithms to enhance the robotic arm's adaptability and robustness in complex environments.
2. Expanding application scenarios: This study mainly focuses on the learning and application of robotic arms in specific tasks. In the future, the system can be applied to a wider range of scenarios, such as home service robots, and combined with large language models (LLM) to fully leverage its capabilities for autonomous learning and adaptation to complex environments. In particular, future work will focus on improving human–robot collaboration by planning robot movements that are aware of human intentions and needs. For example, in door handle manipulation tasks, the robotic arm will be equipped with intention recognition modules and adaptive planning strategies, enabling it to anticipate human actions and synergistically collaborate with humans. Specifically, by training vision-language-action (VLA) models, the robotic arm can better understand and interpret human intentions from multimodal inputs, further enhancing the efficiency, safety, and naturalness of human–robot interaction.
3. Enhancing human–robot interaction and collaboration: In a human–robot collaborative environment, the robotic arm needs to better interact and collaborate with humans. Future research can explore how to enhance the robotic arm's perception and understanding capabilities to enable more natural communication and cooperation with humans.
4. Research on multimodal learning: In addition to spatial position and force information, other modal information (such as sound, touch, etc.) can be explored to enrich the robotic arm's perceptual abilities, further improving its learning efficiency and task execution accuracy. Through continuous optimization of algorithms, expansion of application scenarios, and enhancement of human–robot interaction and collaboration, it is believed that the autonomous learning capabilities of robotic arms will be further improved in the future, bringing more convenience and benefits to human production and life.

Funding. This research was supported by the National Key Research and Development Program of China (2022YFB4700501).

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Data availability and access. Data are contained within the article.

Ethical and informed consent for data used. Not applicable.

Consent for publication. Not applicable.

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

Authors contribution. Conceptualization: Ze Cui; Methodology: Chenzhao Sun; Software and validation: Jingao Huang, Yang Zhang, and Zhenghao Chen; Writing – original draft preparation: Chenzhao Sun; Writing – review and editing: Ze Cui and Zhenghao Chen; Supervision: Ze Cui. All authors have read and agreed to the published version of the manuscript.

References

[1] M. Waseem and Q. Chang, "From Nash Q-learning to nash-MADDPG: Advancements in multiagent control for multiproduct flexible manufacturing systems," *J. Manuf. Syst.* **74**, 129–140 (2024).

[2] M. Boal, C. G. Di Girasole, F. Tesfai, T. E. M. Morrison, S. Higgs, J. Ahmad, A. Arezzo and N. Francis, "Evaluation status of current and emerging minimally invasive robotic surgical platforms," *Surg. Endosc.* **38**(2), 554–585 (2024).

[3] J. Guo, S. Wang, Y. Mao, G. Wang, G. Wu, Y. Wu and Z. Liu, "Supervised learning study on ground classification and state recognition of agricultural robots based on multi-source vibration data fusion," *Comput. Electron. Agr.* **219**, 108791 (2024).

[4] A. Kumar, P. K. Ray and M. H. Ansari, "AI-based self-learning robotic arm using microcontroller," *J. Control Instrum. Eng.* **10**(1), 20–25 (2024).

[5] O. Kroemer, S. Niekum and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms. arXiv preprint arXiv: 1907.03146, 2019.

[6] C. A. Liang, S. A. Munson and J. A. Kientz, "Embracing four tensions in human-computer interaction research with marginalized people," *ACM Trans. Comput.-Hum. Interact.* **28**(2), 1–47 (2021).

[7] M. B. Gawali and S. S. Gawali, "Optimized skill knowledge transfer model using hybrid Chicken Swarm plus Deer Hunting Optimization for human to robot interaction," *Knowl.-BASED Syst.* **220**, 106945 (2021).

[8] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.* **26**(5), 674–691 (2021).

[9] C. Xu, Y.-B. Zhao, Z. Lu and Y. Zhang, "Reinforcement-learning-based algorithms for optimization problems and applications to inverse problems," (2025).

[10] Y. Cao, K. Ni, T. Kawaguchi and S. Hashimoto, "Path following for autonomous mobile robots with deep reinforcement learning," *Sensors* **24**(2), 561 (2024).

[11] L. Lyu, Y. Shen and S. Zhang, "The Advance of Reinforcement Learning and Deep Reinforcement Learning," **In**: *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)* (2022) pp. 644–648.

[12] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, arXiv preprint arXiv: 1801.01290, 2018.

[13] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine, "Soft actor-critic algorithms and applications, arXiv preprint arXiv: 1812.05905, 2018.

[14] H. Guo, Z. Ren, J. Lai, Z. Wu and S. Xie, "Optimal navigation for AGVs: A soft actor–critic-based reinforcement learning approach with composite auxiliary rewards," *Eng. Appl. Artif. Intel.* **124**, 106613 (2023).

[15] H. Li, S. Zhang and D. Guo, "RoboCleaner: Robotic tabletop cleaning via VLM-powered multi-agent collaboration," *IEEE Trans. Autom. Sci. Eng.* **23**, 3651–3660 (2026).

[16] X. Liu, S. Wen, Z. Guo and H. Liu, "Transfer learning-based sparse-reward meta-Q-learning algorithm for active SLAM," *Expert Syst. Appl.* **299**, 129799 (2026).

[17] Y. Liu, Y. Zeng, B. Ma, Y. Pan, H. Gao and Y. Zhang, "Improved demonstration-knowledge utilization in reinforcement learning," *IEEE Trans. Artif. Intell.* **5**(5), 2139–2150 (2024).

[18] T. Huang, K. Chen, B. Li, Y.-H. Liu and Q. Dou, "Demonstration-guided reinforcement learning with efficient exploration for task automation of surgical robot. arXiv preprint arXiv: 2302.09772, 2023.

[19] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills. arXiv preprint arXiv: 1802.09564, 2018.

[20] P. Wu, A. Escontrela, D. Hafner, K. Goldberg and P. Abbeel, "Daydreamer: World models for physical robot learning. arXiv preprint arXiv: 2206.14176, 2022.

[21] M. B. Gawali and S. S. Gawali, "Development of improved coyote optimization with deep neural network for intelligent skill knowledge transfer for human to robot interaction," *Int. J. Intell. Robot. Appl.* **6**(2), 288–305 (2022).

[22] S. Levine, Z. Popović and V. Koltun, "Nonlinear Inverse Reinforcement Learning with Gaussian Processes," **In**: *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS), NIPS'11* (Curran Associates Inc., Red Hook, NY, USA, 2011) pp. 19–27.

[23] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville, "Improved training of wasserstein gans. arXiv preprint arXiv: 1704.00028, 2017.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).

[25] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annu. Rev. Control Robot. Auton. Syst.* **2**, 253–279 (2019).

[26] E. Todorov, T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control," **In**: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2012) pp. 5026–5033.

[27] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany and Y. Zhu, "Robosuite: A modular simulation framework and benchmark for robot learning. arXiv preprint arXiv: 2009.12293, 2020.

[28] Z. Zhu, K. Lin, A. K. Jain and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(11), 13344–13362 (2023).

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press

[29] R. Azzam, M. Chehadeh, O. Abdulhay, I. Boiko and Y. Zweiri, "Learning to navigate through reinforcement across the Sim2Real gap," TechRxiv, (2022).
[30] S. Abeyruwan, L. Graesser, D. B. D'Ambrosio, A. Singh, A. Shankar, A. Bewley, D. Jain, K. Choromanski and P. R. Sanketi, "I-Sim2Real: Reinforcement learning of robotic policies in tight human–robot interaction loops, arXiv preprint arXiv: 2207.06572, 2022.
[31] K. Iyengar, S. M. H. Sadati, C. Bergeles, S. Spurgeon and D. Stoyanov, "Sim2Real transfer of reinforcement learning for concentric tube robots," *IEEE Robot. Autom. Lett.* **8**(10), 6147–6154 (2023).
[32] S. S. Sandha, L. Garcia, B. Balaji, F. Anwar and M. Srivastava, "Sim2Real Transfer for Deep Reinforcement Learning with Stochastic State Transition Delays," **In:** Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, vol. **155** of Proceedings of Machine Learning Research (PMLR, 2021) pp. 1066–1083.
[33] X. Liu, G. Wang, Z. Liu, Y. Liu, Z. Liu and P. Huang, "Hierarchical reinforcement learning integrating with human knowledge for practical robot skill learning in complex multi-stage manipulation," *IEEE Trans. Autom. Sci. Eng.* **21**(3), 3852–3862 (2024).
[34] C. Chen, P. Xiang, J. Zhang, R. Xiong, Y. Wang and H. Lu, "Deep reinforcement learning based co- optimization of morphology and gait for small-scale legged robot," *IEEE/ASME Trans. Mechatronics* **29**(4), 2697–2708 (2024).
[35] L. Chang, L. Shan, W. Zhang and Y. Dai, "Hierarchical multi-robot navigation and formation in unknown environments via deep reinforcement learning and distributed optimization," *Robot. Cim-INT. Manuf.* **83**, 102570 (2023).
[36] W. Tan, X. Fang, W. Zhang, R. Song, T. Chen, Y. Zheng and Y. Li, "A hierarchical framework for quadruped omnidirectional locomotion based on reinforcement learning," *IEEE Trans. Autom. Sci. Eng.* **21**(4), 5367–5378 (2024).
[37] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin and S. Levine, "Reset-Free Reinforcement Learning Via Multi-task Learning: Learning Dexterous Manipulation Behaviors Without Human Intervention," **In**: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021) pp. 6664–6671.

Cite this article: Z. Cui, C. Sun, Z. Chen, J. Huang and Y. Zhang (2026). "Robot autonomous learning system based on multi-source sensor information fusion", Robotica **44**, 540–564. https://doi.org/10.1017/S0263574726103191

https://doi.org/10.1017/S0263574726103191 Published online by Cambridge University Press