ITC 1/54
Information Technology and Control
Vol. 54 / No. 1/ 2025
pp. 254-267
DOI 10.5755/j01.itc.54.1.36362

Accelerated Task Learning Using Q-learning with Reduced State Space and Reward with Bootstrapping Effect

Received 2024/02/16 | Accepted after revision 2024/05/04

HOW TO CITE: Rajamohan, V. P., Jagatheesaperumal, S. K. (2025). Accelerated Task Learning Using Q-learning with Reduced State Space and Reward with Bootstrapping Effect. *Information Technology and Control*, 54(1), 254-267. https://doi.org/10.5755/j01.itc.54.1.36362

Accelerated Task Learning Using Q-learning with Reduced State Space and Reward with Bootstrapping Effect

Varun Prakash Rajamohan, Senthil Kumar Jagatheesaperumal

Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India; e-mails: varunprakash.r@mepcoeng.ac.in; senthilkumarj@mepcoeng.ac.in

Corresponding authors: varunprakash.r@mepcoeng.ac.in

The influence of robots has been rapidly increasing in domestic scenarios. Robots learning in a self-supervised manner will be more efficient than programmed intelligence. In this paper, we present Q-learning-based task learning through interaction with the environment in a table-cleaning scenario. The environment consists of a table partitioned into two segments with a single object on it. The goal of the agent is to learn the sequence of tasks required to clean both segments of the table. Here, the state space is designed in such a way that its size is reduced to achieve better training time and success rate. Besides, in this work, four distinct reward patterns are utilised. The general reward allocation was based on the effect on the environment. Out of the four reward patterns, the one that provides an incentive to attain intermediate success leads to improved performance by the agent compared to the other reward which motivates only upon reaching the terminal state. For the reward with an incentive for intermediate success, the average reward begins to converge around 290 iterations when rewards are allocated for intermediate performance. By 240 iterations, a success rate of around 84% is achieved. Another reward with a small positive value for routine tasks learning performance is poor. Another reward is allotted uniquely by establishing a connection between two consecutive states. This strengthens the bootstrapping effect and exhibits superior performance in comparison to all four reward schemes. The reward associated with the bootstrapping effect demonstrates an average convergence in reward over 220 iterations, with an estimated success rate of 84% achieved over 150 iterations.

KEYWORDS: Interactive learning, Q-learning, cleaning scenario, task planning, reward patterns, bootstrapping.

1. Introduction

Robots are extensively used for object manipulation in various applications, ranging from industry to domestic settings. As population density increases, there is a growing demand for task planning and cleaning robots [23]. These robots play a crucial role in both industrial and domestic environments. Regular cleaning is essential for maintaining the living standards and functionality of these buildings [3]. Therefore, the automation of cleaning processes in building infrastructures has become increasingly necessary. To achieve efficient automation, robotic agents need to be equipped with intelligent algorithms to perform tasks effectively. Infants acquire object manipulation skills at an early stage through interaction. Similarly, Reinforcement Learning (RL) solves problems by making repeated attempts with appropriate rewards. In each instance of interaction, RL performs an action that causes a change in the environment's state. The agent is rewarded accordingly based on the state change. Through repeated actions, the agent maximizes the cumulative reward collected [14].

Mill´an-Arias et al. [20] proposed three different approaches related to object handling by a robotic manipulator. These approaches are soft actor-critic-based Interactive Reinforcement Learning (IRL), Robust Reinforcement Learning (RRL), and Interactive Robust Reinforcement Learning. These techniques were implemented in a simulation environment to classify objects manipulated by the manipulator. In this work, advice is given regarding tasks and the dynamics of the environment. The results show that IRL outperforms the classic RL algorithm. Training episodes in IRL are reduced compared to classic RL. Although RRL requires more training episodes, it achieves good performance even in the presence of external disturbances. IRRL, a combination of IRL and RRL, performs better than RRL in terms of training iterations and maintains robustness even with changes in the dynamics of the environment.

Cruz et al. [8] proposed a learning algorithm for object handling in a simulated domestic cleaning scenario. In this scenario, the agent manipulates obstacles and a sponge to clean the table. The problem was addressed using three different algorithms: classic RL, RL with affordance, and Interactive Reinforcement Learning (IRL). In classic RL, a success rate of 35% was achieved with a thousand training episodes. In the second and third approaches, the number of training episodes was reduced to a hundred. In IRL, even receiving small advice, approximately 10%, enables the robot to complete the cleaning task faster. The above-mentioned work demonstrates better convergence compared to [10]. Munguia-Galeano et al. [22] has undertaken a similar study by utilizing affordance-based RL in the context of Human-Robot Interaction (HRI) applications. Compared to other methods such as Q-learning and Deep-Q network, this approach achieves a higher success rate. The aforementioned works focus on acquiring a sequence of actions to accomplish a given task. In contrast, the authors of [27] concentrated on area coverage using Q-learning. The problem of local optima in Q-learning is overcome by employing predator-prey reward allocation. This approach incorporates three reward functions: Predation Avoidance, Smoothness, and Boundary. The weighted combination of these three rewards provides a better coverage area.

Cheong et al. [6] propose an algorithm for manipulating obstacles to find a collision-free path to the target object. The objects are arranged in a grid-based environment. A deep Q-network is used to learn which object to pick and where to place it in order to obtain a collision-free path to the target. Two different deep learning architectures, single DQN and sequentially separated DQN, are proposed. This work achieves a reduction in execution time and the number of obstacles rearranged by up to 35%. Here, the sequentially separated DQN performs better as the number of obstacles increases. In [18], an interactive perception method for object grasping in a cluttered environment is considered. Initially, the affordance map is obtained from an RGB-D image. This affordance map indicates the confidence level of each pixel for grasping. If the affordance map is inappropriate, a push action is performed until a suitable affordance map is obtained. The exploration strategy is based on deep reinforcement learning. This work demonstrates a higher suction success rate and scene success rate. Authors in [11] proposes a comparable approach without tactile sense, whereas [13] proposes object de-cluttering using mouse-based grasp planning.

Knowledge about handling objects in one task is transferred to other tasks in [1]. Here, probability-based pol-

icy reuse is combined with Q-learning. The learning time for new tasks is reduced due to knowledge transfer. Furthermore, this work can be improved to handle high sensory inputs. In [19, 12], RL is applied to alter the camera viewpoint to improve object detection. Additionally, RL is discussed in robot cleaning tasks in [21, 16, 26, 25], and for navigation with obstacle avoidance in [4]. Cruz et al. in [9], proposed a multimodal feedback for cleaning tasks using a robotic agent. Object detection using deep Q-network and trust region policy optimization is described in [12] and [21] respectively.

One of the main problems with RL is the long execution time required for training [17]. An alteration in training approaches aimed at improving learning performance is discussed in [15]. In the work [15], training is conducted in a manner that alternates between unified and separated training approaches to improve the learning experience. The method of population-directed policy search is introduced in [5] to accelerate the learning process of the Deep-RL-based task offloading technique. A key factor that contributes to increased convergence time is a large and complex state space [2]. In this study, Q-learning is applied to a robotic agent to learn the task of cleaning a table with a single object on it. Compared to the other mentioned works, this study focuses on reducing the size of the state space and varying the reward allocation. As a result, it reduces the number of training iterations and increases the success rate in learning the task of table cleaning. The points mentioned below are the significant contributions of this work:

Q-learning algorithm is applied to the task of cleaning the table with reduced state space.

- The agent's learning rate is improved by incorporating task-customized rewards with a bootstrapping effect.
- Impact analysis of discount factor and learning rate on Q-value convergence are compared for various reward patterns.
- The behavior of the agent is connected to human cognition. To facilitate comprehension of human motivation towards task completion.

The rest of this paper is organized as follows: the environment definition and RL implementation towards a table cleaning scenario are discussed in Section 2. The simulation methodology and the results obtained are discussed in Section 3. Finally, the conclusion and future directions are summarized in Section 4.

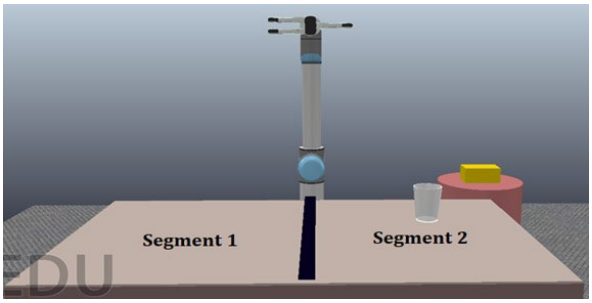
2. Problem Formulation and Methodology

In this work, Q-learning is effectively applied to teach a robotic manipulator (agent) the task of table cleaning. Through trial and error, the robot can learn to adapt its approach based on the effectiveness of its previous actions, resulting in a more efficient learning process for cleaning. This section describes Q-learning and the definition of the environment state for the table cleaning task. A state is represented by a set of variables that describe the agent's location, actions, and the state of the environment. In this work, the size of state space is reduced for improved learning rate and success rate. The reward function in Q-learning defines the agent's goal and provides a measure of success for its actions. This section also explains the distinctive reward allocation mechanism with a bootstrapping effect to benefit the learning of the agent.

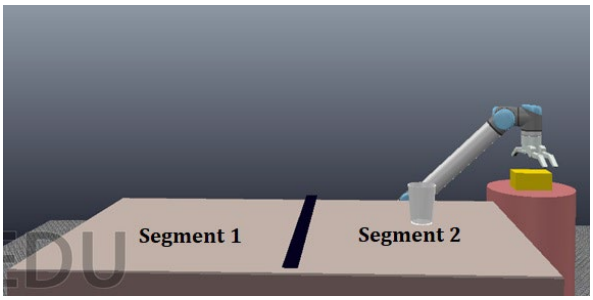
The agent is placed near the table like a janitor ready to clean. The table is partitioned into two segments and contains an object in any one of the segments. Further, a sponge for cleaning is placed in a pre-defined location. The environment is designed in such a way that the size of state space is reduced. In work by Cruz et al. [8] where a similar table cleaning scenario is considered, the cleaning object is placed upon any one of the segments therefore leading to increased state space size. The simulation scenario with the location of the sponge and object is represented in Figure 1. Here, the placement of the sponge in a predefined location reduces the state space by half compared to [8]. Moreover, this way environment not only reduces the state space size but also is more in synchronous with real human cleaning scenarios. In most practical cleaning table scenarios, the sponge is not located on the table rather the janitor comes with a trolley that contains the sponge for cleaning. Thus, the environment design consideration taken in this work is better in terms of state space size and more oriented towards practical table cleaning scenarios.

A state machine is developed to describe the table cleaning scenario with reduced state space. In that, each state is characterised by the location of the object, the agent's location, and the cleanliness status of each segment. For example, in State 1, segments are unclean, the object is located in Segment 2, and the manipulator is positioned above Segment 1. If

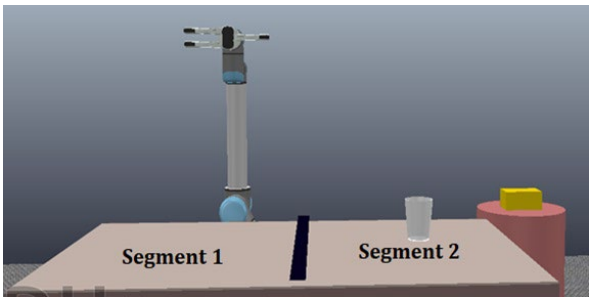
Figure 1
Q-Learning employed in a robotic cleaning task simulation environment



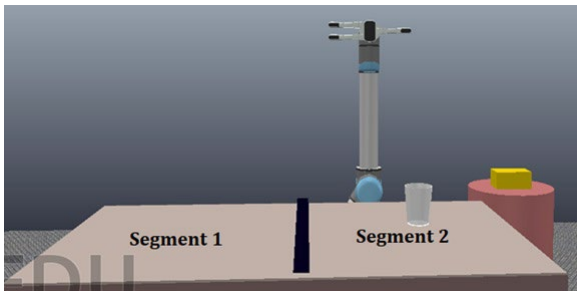
(a) Manipulator in Home location



b) Sponge Location



(c) Manipulator above Segment 1



d) Manipulator above Segment 2

the agent performs the clean action from this state, it transitions to state-7. In state-7, Segment 1 is clean, segment 2 is unclean with the object, and the manipulator is positioned above Segment 1. The complete state description is provided in Table 1. The task for agent learning presented in [8] was similar but had a larger number of states, specifically 45. In this work, state reduction is achieved by confining the sponge to a predefined location, as highlighted in Figure 1(b). The agent is restricted from placing the sponge above any of the segments on the table. This modification in the environmental design has a positive impact on the speed at which agents learn.

The actions performed by the agent are as follows: move, pick, place, and clean. When the agent receives the ”move” action, it transitions from the current segment to the other segment, which can be either from Segment 1 to Segment 2 or vice versa. During the pick action, the agent picks up the object in its current segment. If the current segment does not contain the object, no change in the state occurs. Similarly, the place action involves placing the object held by the manipulator onto the current segment. The location of the sponge is fixed and predefined. Therefore, when the clean action is performed, the agent takes the sponge from the predefined location, cleans the current segment of the manipulator, and then places the sponge back in the home location.

In this work, the focus is not on the ability to grasp, actuate, and reach mechanisms, but rather on learning the correct sequence of actions to be performed by the agent with minimum training iterations and an improved success rate. For example, if the object and manipulator are located in Segment 1 of the table, the desired sequence for successful task completion is as follows: move (right), clean (right), move (left), pick (object), move (right), place (object), move (left), and clean (left). Therefore, a minimum of eight steps is required to reach the terminal state. The objective of the agent is to clean both sides by following the correct sequence of steps. In this work, the learning is attained with minimal training iterations and a high success rate. Improvement in learning rate is ob-

Table 1
State definition for table cleaning scenario with reduced state space for improved learning rate

State Count	Segment 1	Segment 2	Manipulator Location
1	Unclean	Unclean with object	Segment 1
2	Unclean	Unclean with object	Segment 2
3	Unclean	Unclean with object	Segment 2 holding the object
4	Unclean with object	Unclean	Segment 1
5	Unclean with object	Unclean	Segment 2
6	Unclean with object	Unclean	Segment 1 holding the object
7	Clean	Unclean with object	Segment 1
8	Clean	Unclean with object	Segment 2
9	Clean	Unclean with object	Segment 2 holding the object
10	Unclean with object	Clean	Segment 1
11	Unclean with object	Clean	Segment 2
12	Unclean with object	Clean	Segment 1 holding the object
13	Clean	Clean with object	Segment 1
14	Clean	Clean with object	Segment 2
15	Clean	Clean with object	Segment 2 holding the object
16	Clean with object	Clean	Segment 1
17	Clean with object	Clean	Segment 2
18	Clean with object	Clean	Segment 1 holding the object

tained by reducing the size of state space as described above. Further, the learning rate and success percentage are enhanced by allowing task-customised rewards with a bootstrapping effect and encouraging in-between success.

2.1. Strategy for Reward Pattern Formulation

Agent's each interaction with the environment it receives reward with respect to the state change. This encourages learning by performing action and observing its effect on the environment. The object of reinforcement learning is to maximise the reward received over some time. The rewards relate to observations obtained from the surroundings. Additionally, it can be allocated interactively through feedback mechanisms. This enhances the agent's learning with changes in environmental conditions. However, this work aims to enhance the agent's learning by modifying the preprogrammed reward in response to environmental observations. Appropriate instant reward allotment has a major impact on the agent's learning. In this work, different rewards were being considered. The rationale is that rewards that are not appropriately sized may not provide enough motivation for the agent to gain new knowledge. Conversely, excessively big rewards may cause the agent to engage in risky behaviors. Enhancing the rewards is a crucial aspect of developing an effective reinforcement learning algorithm. Therefore, the present investigation examines several reward patterns and analyses their performance. The four different reward patterns, namely r_1, r_2, r_3, and r_4, were used to train the agent as given in Table 2. The reward r_1 allows the highest positive value only when the terminal state is reached, but

Table 2
Multiple Reward Allotment Patterns

Reward	Terminal State	Relevant Cleaning	Irrelevant Cleaning	Pick Place Move	Improper Actions	Looping Penalty
r1	1	-0.01	-0.01	-0.01	-1	-
r2	1	1	-0.01	-0.01	-1	-
r3	1	1	-0.01	0.01	-1	-
r4	1	1	-0.01	0.01	-1	-0.01

the reward r_2 assigns the highest positive value for successes that occur during the intermediate stages. Reward r_3 provides a slight positive incentive for routine actions, in addition, reward r_4 penalises looping by relating two consecutive states.

Reward r_4 is allotted in a new way relating two states thereby enhancing the bootstrapping effect. Similarly, r_2 proves to be better by encouraging in-between success. These reward patterns are described in Table 2, where the terminal state represents the completion of cleaning both segments of the table. "Relevant cleaning" refers to cleaning an unclean segment, while "Irrelevant Cleaning" refers to cleaning an already cleaned segment of the table. Pick, Place, and Move are routine tasks that need to be performed meaningfully to progress toward the terminal state. "Improper action" refers to performing the clean action while holding the object in hand or cleaning the segment where the object is already present, among other cases. Finally, the "looping action" parameter is implemented in a distinct manner where it connects two consecutive states to enhance the bootstrapping effect. This means that the agent receives a penalty when it selects actions that result in switching back and forth between two identical states

In the reward pattern r_1, a high positive reward of 1 is allocated only when the agent reaches the terminal state, i.e., when both segments of the table are clean. The maximum negative reward is given for improper actions performed by the agent from certain states. For all other actions (move, pick, place, clean), a less significant negative reward of −0.01 is awarded. This specific reward pattern has been implemented in the work mentioned in reference [8]. The reward pattern r_2 is also described in Table 2. In r_2, a high negative reward of −1 is given for improper actions. A small negative reward is assigned for all actions, such as moving, picking, placing, and cleaning an already clean segment. The difference compared to r_1 is that a high positive reward of 1 is allocated for cleaning even one unclean segment of the table. In r_1, the high reward of 1 is only given when the terminal state is achieved.

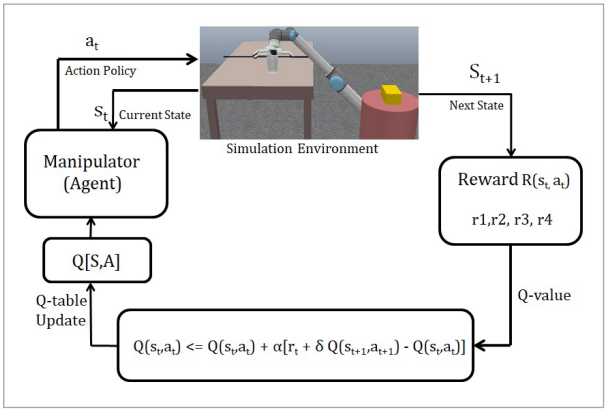
Reward *r3*, as shown in Table 2, is similar to r_2 in many ways in terms of awarding high positive rewards and less significant negative rewards. The difference is that a less significant positive reward of 0.01 is given for routine actions like pick and place. Reward r_3 follows the same pattern as r_2 but with the addition of a small positive reward for routine actions. However, a new aspect is introduced in one of the reward allocations, which relates to two consecutive states. In other words, the agent receives a small penalty of −0.01 for looping between the same states. Looping between the same states occurs, when the agent alternates between move actions or pick and place actions, resulting the agent being stuck between the same two states. This penalty is included in reward r_4 to prevent infinite looping.

2.2. Cognitive Learning Using RL

Infants continuously learn about different objects in their environment by interacting with them. Through interaction, they learn various attributes such as color, shape, and usage. This learning occurs naturally as they observe the cause-and-effect relationships. Similarly, the agents in the RL environment learn through repeated interactions with the environment. Like other machine learning algorithms, RL's learning improves with experience. However, unlike other algorithms, the data for learning in RL is collected by the agent performing repeated actions on the environment. As a result, learning in RL happens incrementally in a completely unknown environment.

RL aims to maximise the reward in a given scenario to reach the terminal state. Q-learning is a popular RL

Figure 2
Q-Learning employed in a robotic cleaning task simulation environment



algorithm used to solve a wide range of problems in robotics and other domains. Q-learning is an off-policy RL algorithm that can identify the optimal action to take in a given state without requiring a model of the environment. The agent decides on the next action based on its current location within the environment.

Figure 2 illustrates the complete flow of the learning sequence in the Q-learning algorithm. The agent maintains a table whose dimensions are determined by the size of the state space and the number of available actions. This table is referred to as the Q- table, represented as Q [S, A], where S is the set of possible states, and A is the set of possible actions that the agent can take in those states. The values in the Q-table indicate the expected cumulative rewards that the agent will receive by taking a particular action in a specific state. The Q-table is updated during each iteration of the learning process as the agent interacts with the environment and receives feedback in the form of rewards. This updating of the Q-table during learning iterations enables the agent to improve its estimates and make better decisions. The Q-table is updated using Equation (1), which is commonly known as the Bellman equation.

$$Q^{\varpi}(s_t,a_t)=E[r_{t+1+}\gamma r_{t+2}+\gamma^2 r_{t+2}+....]. \quad (1)$$

Initially, all the values in the Q-table are set to zero, and they are updated as training iterations progress. The agent takes an action and receives a reward based on the effect it has on the environment. The Q-value is calculated based on the current Q-value and the reward received in response to the performed action. The Q- value is computed according to Equation (2). By updating the Q-table in this manner, the value function Q is maximized, enabling the agent to make better decisions. This allows the agent to select actions that are more likely to result in the highest cumulative rewards

$$Q(s_t, a_t) <= Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(_{st+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2)$$

$$Q^*(s, a) = max\, Q^{\varpi}(s, a). \quad (3)$$

In Equation (5), at represents the current action chosen from a set of available actions A that can be performed by the agent on the environment. Similarly, s_t is the current state from a set of states S representing the environment. Among all the states in set S, one state S_T is called a terminal state, marking the end of the training episode. In this work, the terminal state is achieved when both segments of the table have been cleaned by the agent. r_t represents the immediate reward received according to a predefined pattern. With each action performed by the agent, there is a state transition in the environment. Consequently, a positive or negative reward is received as feedback for each action and corresponding state transition. The objective of Q-learning is to learn the policy that maximizes the Q-value for a state action pair as represented in Equation (3).

In this work, out of the four rewards that are allocated, one is given in a distinct manner that depends on the relationship between two consecutive states. The agent, present in the environment in the current state $s_t \in S$, performs the action $a_t \in A$, receives the reward r_t, and undergoes a state transition to the next state s_{t+1}. The parameter γ is the discount factor, determining the weight given to future rewards. The value of γ influences the agent's consideration of long-term versus immediate rewards. The parameter α is the learning rate, which determines how much new information is incorporated into the action selection policy. Striking a balance between these parameters allows the agent to appropriately value future and current rewards.

Temporal Difference (TD) learning, as given in Equation (4), is used to update the Q-value estimate of a state-action pair. TD learning updates the Q-value by bootstrapping the observed value and the estimated value of the next state. The estimates of future rewards are obtained by using the current estimate of the value function, reducing variance in the estimates. This approach aids in the convergence of the optimal value, even in cases of non-deterministic

transitions in the environment. The temporal difference though includes a bootstrapping effect by relating two states, this is further enhanced in reward r_4 to make the learning faster with a high success rate.

$$V(s) <= V(s) + \alpha\,[r + \gamma V(s' - V(s)]. \quad (4)$$

Q-learning aims to determine a policy that maximizes the cumulative reward over time. An optimal policy, denoted as $\pi*$, is one that is either the best or equal to all other policies. Such policies share the same optimal action-value function, denoted as Q^* (as given in Equation (3)). Proper reward allocation is crucial in Q-learning because rewards have a significant impact on the optimal policy that the agent will learn. The agent's objective is to maximize the total reward it receives over time. Proper reward allocation is crucial in Q-learning because rewards play a key role in shaping the optimal policy that the agent will learn. Rewards should be designed to provide a clear signal to the agent about which actions are desirable in a given state. If the rewards are ambiguous or do not align well with the agent's objective, the agent may learn a sub-optimal policy. Therefore, it is important to carefully design the reward function to ensure proper reward allocation. The reward function should be customized to suit the specific task that the agent is attempting to accomplish.

Algorithm 1:
Q-Learning for Table Cleaning (Two Segments)

Input: Q[S,A], State Table, List of Actions (A), Reward r_t
Output: Q[S,A] with optimised Q-values

for each episode **do**
  Select $a_t \leftarrow$ ε-greedyActionSelection(s_t)
  Perform the action a_t on the table
  $s_{t+1} \leftarrow$ StateTransit(s_t, a_t)
  Allot Reward r_t
  Update Q[S,A] using equation 2
  itrCnt ← itrCnt + 1
  if s_{t+1} is Terminal State then
    End episode
  else if itrCnt ≥ maxItr then
    End episode
  else if s_{t+1} is a failed State then
    $s_{t+1} \leftarrow s_t$
  end if
  $s_t \leftarrow s_{t+1}$
end for

Algorithm 1 presents the framework of Q-learning in action for the table cleaning application. It outlines the flow of a single episode, which consists of multiple iterations. Each episode requires several inputs: a state table, action definitions, reward pattern, and Q-table initialization. The state table is constructed based on the environment specifications, which are described in detail in Table 1. Next, a list of actions performed by the agent on the environment needs to be provided. To quantify the impact of actions on the environment, an appropriate reward pattern is designed. Finally, the Q-table denoted as Q[S, A], is initially populated with zero values, with the table dimensions corresponding to the number of states and actions. The Q-learning algorithm takes these inputs and updates the Q-table Q[S, A] during each iteration. As a result, the Q-learning algorithm produces an optimized Q-table that guides the agent in selecting the appropriate action for each state.

$$a_t = argmax\ Q(S_t, a). \quad (5)$$

Each iterations, begins with ϵ-greedy-based action selection, where the value of ϵ is set to 0.1. Equation (5) is primarily used to determine the next action. In this equation, S_t represents the current state of the environment, a is an action, and A corresponds to the list of all possible actions. The agent then performs the chosen action on the table, and the state transition in the environment is observed. The Q-value is computed based on the received reward, followed by an update to the Q-table. The iteration count is then incremented. If the next state is the terminal state, the current episode is stopped, and a new episode begins. Similarly, if the iteration count reaches its maximum value, the episode is terminated. This termination is implemented to avoid infinite looping within a single episode. When the above two conditions are not met, the current state is updated to the next state, and the algorithm proceeds to the next iteration within the same episode. If the next state is a failed state, the current state remains the same, and the episode continues. Therefore, a set of iterations constitutes an episode, and a series of multiple episodes is referred to as a training run.

3. Simulations and Outcomes

The findings obtained from the application of Q-learning-based cleaning with a reduced state space are discussed in this section. The simulation investi-

gations were carried out using the CoppeliaSim software [24]. The internal inverse kinematics module of CoppeliaSim is used for planning manipulator movements.

3.1. Discussions on Impact of Reward Patterns on Learning Speed of Agent

The agent performing an action on the environment, causing a state change, is considered a single iteration. A collection of iterations is referred to as an episode. Each episode ends only when the agent reaches the terminal state or after performing a certain number of iterations. If the agent performs an improper action, the training continues from the previous state instead of restarting from the initial state. Therefore, each episode is terminated after a predefined number of iterations, even if the agent has not reached the terminal state, to avoid infinite looping within an episode. The iteration count may be higher in certain cases, even if the episode count is lower. Therefore, in the results graph, the average reward and success rate are plotted against iterations rather than episodes. A collection of episodes is considered a single run. In the graphs, the average reward and success rate values represent averages across different runs. The agent's learning is quantified using two parameters namely average reward convergence and success rate. Average reward convergence tells about the stability and speed at which the agent learns. Success rate analysis gives inference about the maximum efficiency reached by the agent.

The success rate obtained for different reward variations is shown in Figure 3. It can be observed that the agent achieves a success rate of approximately 84% for all four types of rewards applied to the table cleaning scenario. However, the difference lies in the number of training iterations required to reach the maximum success rate. Rewards r1 and r2 exhibit similar patterns with small differences, as shown in Table 2. With this variation in reward allocation, r2 enables the agent to reach the maximum success rate 400 iterations earlier than r1. The slope of the curve for reward r2 is steeper compared to the slope for reward r1. A similar trend can be observed in the convergence curve of the average reward value, as depicted in Figure 4. Reward r2 converges 400 iterations earlier compared to reward r1. The maximum average reward value for r2 is 0.3, while it is 0.17 for r1. This decrease in the overall average reward for r1 is due to the limited allocation of high positive reward, which is only given upon reaching the terminal state. This reduction in average reward leads to a drop in success percentage during earlier iterations. By comparing the results obtained from applying reward patterns r1 and r2, it can be inferred that the agent learns better when intermediate success states are appropriately rewarded.

Figure 3
Success rate observed for various iterations with variations in rewards (Q-Learning)

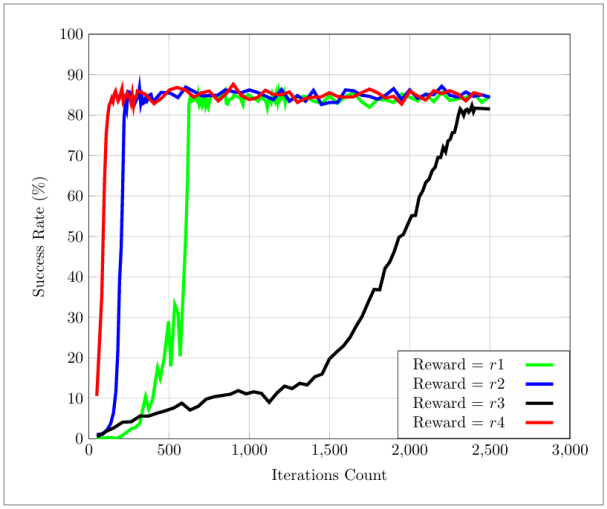


Figure 4
Average Reward observed for various iterations with variations in rewards (Q-Learning)

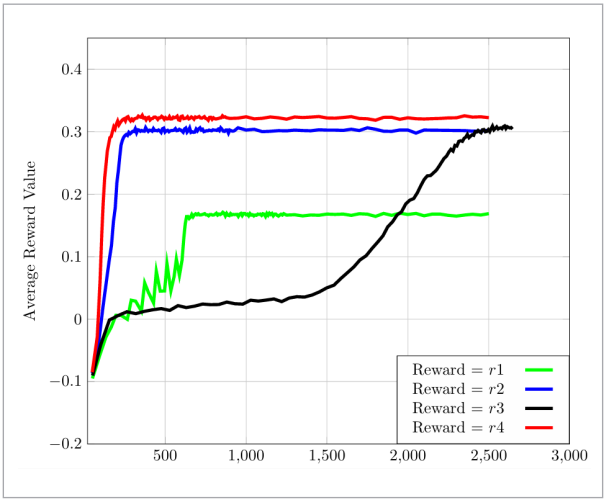
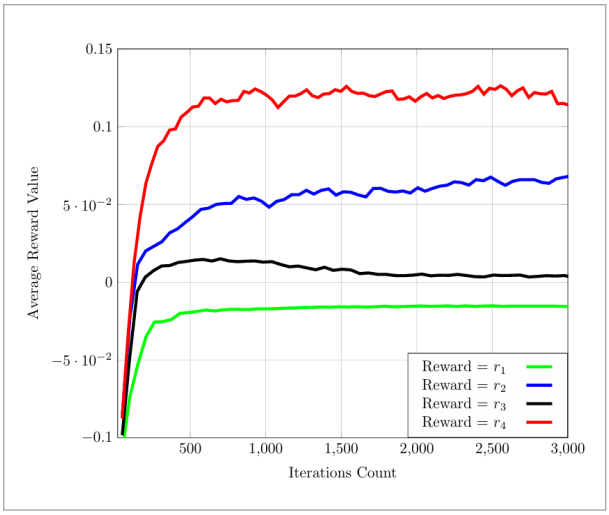


Figure 5
Average reward value observed for various iterations with variations in rewards (SASRA)



The average reward convergence graph using SARSA (State-action- reward-state-action) RL is given in Figure 5. The mean reward value in SARSA has decreased substantially than Q-learning. This greatly diminishes the success rate in SARSA. The improved performance in Q-learning can be attributed to the utilisation of exploitation behavior in selecting the next state-action pair. Nevertheless, the reward pattern behavior remains consistent in both algorithms, with the observation that the intermediate success in r_2 leads to more effective learning than r_1. Moreover, r_4 relating two states enhances the bootstrapping effect resulting in better performance.

Reward r_3 differs from r_2 by allocating small positive rewards for routine actions like pick and place. This minor change has had a significant negative impact on the agent's learning. With reward r_3, the agent requires an additional 2080 iterations to reach the maximum success rate of approximately 84% (Figure 3). Similarly, the convergence of the average reward with reward r_3 also takes an additional 2080 iterations (Figure 4). The low positive value for routine tasks causes the agent to get occupied with routine tasks rather than actively pushing towards the terminal state. A reward system that aligns with the agent's progress towards the goal, rather than routine actions, would be more effective. The final reward, r_4, introduces changes to reward r_3 by including a loop penalty. This customized reward modification is specific to this particular scenario. With the inclusion of the loop penalty, the agent's training iterations are significantly improved, even slightly surpassing the performance of r_2. The average reward value of r_4 is slightly higher than that of r_2, mainly due to the presence of a less significant positive reward of 0.01. This slight increase tends to better performance of r4 compared to r_2. From reward r_4, it can be inferred that a tailored reward allocation has a highly positive impact on improving the agent's learning speed. The favorable impact in r_4 is attributed to its ability to impose a looping penalty. This phenomenon of allowing a looping penalty establishes a connection between two consecutive states, hence amplifying the bootstrapping effect. This impact is in addition to the existing bootstrapping effect gained from temporal difference learning, as shown in Equation (2).

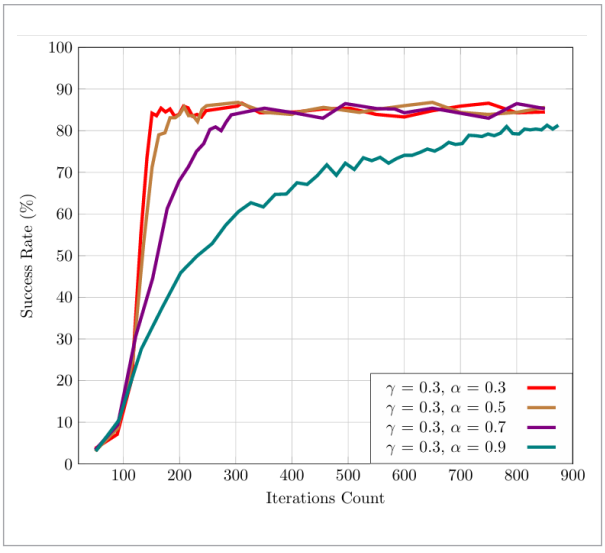
Table 3 presents a comparison of the training iterations required to achieve the maximum possible success percentage across various reward patterns. In this particular environment design, the success rate obtained was approximately 84% for all the reward patterns. The lowest success rate of 81.08% is achieved using reward r3 after 2360 iterations (70 episodes). For reward r1, the agent achieves a success rate of 83.94% after 630 iterations (25 episodes). With reward r2, the agent achieves a success rate of 84.73% after 240 iterations (14 episodes). Similarly, with reward r2, the agent achieves a success rate of 84.73% after 630 iterations (25 episodes). A minor variation in reward r4 results in the agent achieving a success rate of around 84.77% after 142 iterations (10 episodes). A comparable task involving learning to clean a table with a single object on it was performed in [8]. In that work, the learning was performed using classic RL with a reward pattern of r1, and the success rate obtained was 35%. The comparison with similar works is given in table 4. It is observed that the proposed method has faster average reward convergence and a higher success rate compared to other works. Though the method Affordance-RL has a high success rate it needs a lot of pre-knowledge about the environment. In Affordance-RL the pre-knowledge need to be hard-coded into the program. Whereas this work using Q-learning learns incrementally during the training phase itself. Therefore, a small modification in the environment and reward impacts positively the learning efficiency of the agent.

Table 3
Learning Speed Comparisons for Different Reward Patterns

Methodology	Average Reward Convergence		Maximum Success Rate	
	Iterations	Episodes	Iterations	Episodes
r1	680	31	630	25
r2	290	21	240	14
r3	2350	70	2350	70
r4	190	15	140	10

The convergence of average reward for different reward patterns in the table cleaning learning task is also described in Table 3. In the work [8], a similar table cleaning scenario trained using classic RL with reward r_1 is proposed. It takes nearly 1000 episodes for the average reward to converge, and the convergence is not smooth. In this work, with the same reward of r_1, the average reward converges in 31 episodes. Other methods mentioned in [8] are Interactive-RL (IRL) and affordance-based RL, which have average reward convergences of around 40 and 55 episodes, respectively. Except for the reward pattern r_3, the proposed work shows better convergence speed compared to all the methods mentioned in [8]. The reduction in state count has a positive impact on improving the success rate and reducing the training iterations. Furthermore, the variation in reward allocation has positively influenced the reduction in training iterations with a better success rate.

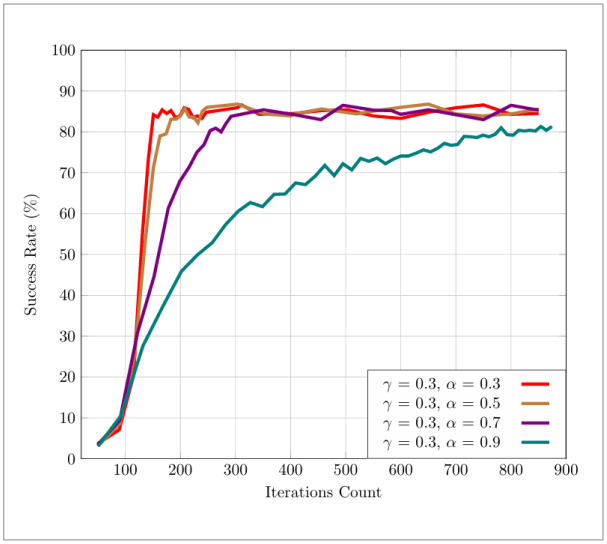
Figure 6
Success rate with variations in learning rate for the reward r4 (Q-Learning)



The impact of the success rate on the alteration of parameters γ and α is given in Figure 6. Despite variations in γ and α, the agent achieves a success rate of around 82%. However, it has an impact on the training iterations. With γ and α values of 0.3, fewer iterations are required, and the agent attains a success rate of 84% by 150 iterations. Subsequently, it consistently reaches the terminal stage with a success rate of around 84%. When the α value is changed to 0.5, an additional 33 iterations are needed to reach the 84% success rate. The agent's performance degrades further with a higher α value. Similar behaviour is observed for the convergence of average reward values in Figure 7. The variation in the γ parameter has little or no impact on the speed of convergence. When choosing a learning rate α, it's important to experiment and adjust accordingly based on factors like the reward system and environment complexity. In this scenario, it was observed through the graphs of average reward and success rate that an α of 0.3 yields better learning compared to its higher values. The learning parameter α has a decaying effect upon the instantaneous reward r_t. This implies that the latest reward significantly influences the estimation more than previous rewards, leading to better learning.

Hence, the use of better environmental design in this study resulted in better learning outcomes by the agent. Then the agent's learning is analysed with diverse set of reward patterns. The conclusion drawn from the varied reward pattern is the utilisation of positive incentives, even in a limited manner, for routine tasks hinders the process of learning. In contrast, minor negative incentives for repetitive tasks result in more effective learning. The agent's learning pace

Figure 7
Average reward with variations in learning rate for the reward r4 (Q-Learning)



is enhanced by providing incentives for achieving intermediate successes. Another reward is granted in a unique way relating two successive states, accelerating the bootstrapping process and leading to better learning. Furthermore, the insight relating the agent's learning to human motivation is described in the upcoming section 3.2.

3.2. Inference Related to Human Cognition

This section relates the agent's learning process to the application of human intelligence. So that the human's cognition can be better understood and trained. The reward r_2 learns better than r_1 with the only difference of awarding maximum reward for relevant cleaning action. That is giving a reward for intermediate success will improve the agent's performance and also motivate the humans as well. The efficiency of r_3 falls drastically just by awarding a small positive reward for routine actions instead of a small negative reward. Thus, to humans, the reward ought to be given for goal-oriented work instead of routine work for better productivity. As seen with r_3, the learning gets dragged by focusing on getting small benefits instead of high-value goals. Finally, in r4 the learning is improved by penalising meaningless redundant work. Again, this is inferred as for productive learning, progress is important than just staying busy with too many activities

4. Conclusions and Future Work

This paper proposes a methodology to apply Q- learning to a table-cleaning scenario with a reduced state space. As a result, the agent's training iterations are reduced while achieving an increased success rate. Furthermore, different reward patterns are specifically altered for the task to obtain a better success rate with fewer training iterations. The reward r_4 is allotted in a distinct manner enhancing the bootstrapping effect of consecutive states. Reward r_2 encourages in-between success and discourages routine actions. The reward allotment of r_2 and r_4 proves to be

Table 4
Success Rate Comparison

Methodology	Average Reward Convergence	Maximum Success Rate	
	Episodes	Success Rate (%)	Episodes
Classic RL [8]	1000	35	1000
Classic RL [7]	-	4	1000
Affordanace RL [8]	48	99.9	-
Interactive RL (L=0.5) [8]	40	-	-
This work with reward r1	31	83.94	25
This work with reward r2	21	84.73	14
This work with reward r3	70	81.08	70
This work with reward r4	15	84.77	10

efficient, as they achieve a maximum success rate of around 84% in 240 and 150 iterations, respectively. Similarly, their average reward convergence occurs in 290 and 190 iterations, respectively. The number of training iterations needed for average reward convergence varies by 2210 iterations between the best-case and worst- case reward patterns mentioned in this work. Thus, this work addresses the major challenge of high training iterations in RL by properly altering the environment design and reward allotment. In this context, rewards are distributed according to a predetermined analysis of the environment. However, the rewards can also be adjusted based on interactive feedback from the environment, which allows for more effective learning in dynamic environmental conditions. Further progress can be made by incorporating the adaptive ϵ-greedy strategy for action selection. Additionally, the agent can be trained to handle multiple static and dynamic objects. With an increase in complexity regarding object count and environment dynamics, Deep-RL may be suitable for efficient learning. The choice of parameters α and γ requires experimentation and tuning, as the optimal values depend on the problem and environment. Therefore, optimisation techniques for selecting parameters with optimal values can be employed

References

1. Boloka, T., Makondo, N., Rosman, B. Knowledge Transfer Using Model-Based Deep Reinforcement Learning. In 2021 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA), 2021, 1-6. https://doi.org/10.1109/SAUPEC/RobMech/PRASA52254.2021.9377247
2. Bou-Ammar, H., Taylor, M. E., Tuyls, K., Weiss, G. Reinforcement Learning Transfer Using a Sparse Coded Inter-Task Mapping. In EUMAS, 2011. https://doi.org/10.1007/978-3-642-34799-3_1
3. Chae, H., Park, G., Lee, J., Kim, K., Kim, T., Kim, H. S., Seo, T., Façade Cleaning Robot with Manipulating and Sensing Devices Equipped on a Gondola. IEEE/ASME Transactions in Mechatronics, 2021, 26(4), 1719-1727. https://doi.org/10.1109/TMECH.2021.3077634
4. Chen, L., Wang, Y., Miao, Z., Mo, Y., Feng, M., Zhou, Z., Wang, H. Transformer-Based Imitative Reinforcement Learning for Multi-Robot Path Planning. IEEE Transactions on Industrial Informatics, 2023, 1-11. https://doi.org/10.1109/TII.2023.3240585
5. Chen, N., Yao, X., Yuan, X., Ou, P. Performance Optimization of Serverless Edge Computing Function Offloading Based on Deep Reinforcement Learning. Available at SSRN 4029467.
6. Cheong, S., Cho, B., Lee, J., Lee, J., Kim, D. H., Nam, C., Kim, C., Park, S. Obstacle Rearrangement for Robotic Manipulation in Clutter Using a Deep Q-Network. Intelligent Service Robotics, 2021, 14, 549-561. https://doi.org/10.1007/s11370-021-00377-4
7. Cruz, F., Magg, S., Weber, C., Wermter, S., Improving Reinforcement Learning with Interactive Feedback and Affordances. In 4th International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB), Genoa, Italy, 2014, 165-170. https://doi.org/10.1109/DEVLRN.2014.6982975
8. Cruz, F., Magg, S., Weber, C., Wermter, S. Training Agents with Interactive Reinforcement Learning and Contextual Affordances. IEEE Transactions on Cognitive and Developmental Systems, 2016, 8(4), 271-284. https://doi.org/10.1109/TCDS.2016.2543839
9. Cruz, F., Parisi, G. I., and Wermter, S. Multi-Modal Feedback for Affordance-Driven Interactive Reinforcement Learning. In 2018 International Joint Conference on Neural Networks (IJCNN), 2018, 1-8. https://doi.org/10.1109/IJCNN.2018.8489237
10. Cruz, F., Twiefel, J., Magg, S., Weber, C., and Wermter, S. Interactive Reinforcement Learning Through Speech Guidance in a Domestic Scenario. In 2015 International Joint Conference on Neural Networks (IJCNN), 2015, 1-8. https://doi.org/10.1109/IJCNN.2015.7280477
11. Deng, Y., Guo, X., Wei, Y., Lu, K., Fang, B., Guo, D., Liu, H., Sun, F., Deep Reinforcement Learning for Robotic Pushing and Picking in Cluttered Environment. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, 619-626. https://doi.org/10.1109/IROS40897.2019.8967899
12. Han, X., Liu, H., Sun, F., Zhang, X. Active Object Detection with Multistep Action Prediction Using Deep Q-Network. IEEE Transactions on Industrial Infor-

matics, 2019, 15(6), 3723-3731. https://doi.org/10.1109/TII.2019.2890849

13. Jagatheesaperumal, S. K., Rajamohan, V. P., Saudagar, A. K. J., AlTameem, A., Sajjad, M., Muhammad, K. MOMO: Mouse-Based Motion Planning for Optimized Grasping to Declutter Objects Using a Mobile Robotic Manipulator. Mathematics, 2023, 11(20), 4371. https://doi.org/10.3390/math11204371
14. Jang, B., Kim, M., Harerimana, G., Kim, J. W. Q-Learning Algorithms: A Comprehensive Classification and Applications. IEEE Access, 2019, 7, 133653-133667. https://doi.org/10.1109/ACCESS.2019.2941229
15. Jiang, W., Han, H., Zhang, Y., Mu, J. Federated Split Learning for Sequential Data in Satellite-Terrestrial Integrated Networks. Information Fusion, 2024, 103, 102141. https://doi.org/10.1016/j.inffus.2023.102141
16. Kim, J., Cauli, N., Vicente, P., Damas, B. D., Bernardino, A., Santos-Victor, J., Cavallo, F. Cleaning Tasks Knowledge Transfer Between Heterogeneous Robots: A Deep Learning Approach. Journal of Intelligent & Robotic Systems, 2019, 98, 191-205. https://doi.org/10.1007/s10846-019-01072-4
17. Knox, W. B., Stone, P. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. In The Fifth International Conference on Knowledge Capture, 2009. https://doi.org/10.1145/1597735.1597738
18. Liu, H., Deng, Y., Guo, D., Fang, B., Sun, F., Yang, W. An Interactive Perception Method for Warehouse Automation in Smart Cities. IEEE Transactions on Industrial Informatics, 2021, 17(2), 830-838. https://doi.org/10.1109/TII.2020.2969680
19. Liu, H., Wu, Y., Sun, F. Extreme Trust Region Policy Optimization for Active Object Recognition. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(6), 2253-2258. https://doi.org/10.1109/TNNLS.2017.2785233
20. Millán-Arias, C., Torres Fernandes, B. J., Cruz, F., Dazeley, R., Fernandes, S. M. M. A Robust Approach for Continuous Interactive Actor-Critic Algorithms. IEEE Access, 2021, 9, 104242-104260. https://doi.org/10.1109/ACCESS.2021.3099071
21. Moon, W., Park, B., Nengroo, S. H., Kim, T., Har, D. Path Planning of Cleaning Robot with Reinforcement Learning. In IEEE Robotics and Sensors Environments (ROSE), 2022, 1-7. https://doi.org/10.1109/ROSE56499.2022.9977430
22. Munguia-Galeano, F., Veeramani, S., Hernández, J. D., Wen, Q., Ji, Z. Affordance-Based Human-Robot Interaction with Reinforcement Learning. IEEE Access, 2023, 11, 31282-31292. https://doi.org/10.1109/ACCESS.2023.3262450
23. Parween, R., Clarissa, L. T. L., Naing, M. Y., Fuad, N. A. F. B. M., Elara, M. R. Modeling and Analysis of the Cleaning System of a Reconfigurable Tiling Robot. IEEE Access, 2020, 8, 137770-137782. https://doi.org/10.1109/ACCESS.2020.3009120
24. Rohmer, E., Singh, S. P. N., Freese, M. V-REP: A Versatile and Scalable Robot Simulation Framework. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, 1321-1326. https://doi.org/10.1109/IROS.2013.6696520
25. Sun, C., van der Tol, R., Melenhorst, R., Ponce Pacheco, L. A., Koerkamp, P. G. Path Planning of Manure-Robot Cleaners Using Grid-Based Reinforcement Learning. Computers and Electronics in Agriculture, 2024, 226, 109456. https://doi.org/10.1016/j.compag.2024.109456
26. Thakar, S., Malhan, R. K., Bhatt, P. M., Gupta, S. K. Area-Coverage Planning for Spray-Based Surface Disinfection with a Mobile Manipulator. Robotics and Autonomous Systems, 2022, 147, 103920. https://doi.org/10.1016/j.robot.2021.103920
27. Zhang, M., Cai, W., Pang, L. Predator-Prey Reward-Based Q-Learning Coverage Path Planning for Mobile Robot. IEEE Access, 2023, 11, 29673-29683. https://doi.org/10.1109/ACCESS.2023.3255007