biomimetics

MDPI

Article

Reinforcement Learning with Task Decomposition and Task-Specific Reward System for Automation of High-Level Tasks

Gunam Kwon, Byeongjun Kim and Nam Kyu Kwon *

Department of Electronic Engineering, Yeungnam University, Gyeongsan 38541, Republic of Korea; nineman@yu.ac.kr (G.K.); slim7928@ynu.ac.kr (B.K.)
* Correspondence: namkyu@yu.ac.kr; Tel.: +82-53-810-3095

Abstract: This paper introduces a reinforcement learning method that leverages task decomposition and a task-specific reward system to address complex high-level tasks, such as door opening, block stacking, and nut assembly. These tasks are decomposed into various subtasks, with the grasping and putting tasks executed through single joint and gripper actions, while other tasks are trained using the SAC algorithm alongside the task-specific reward system. The task-specific reward system aims to increase the learning speed, enhance the success rate, and enable more efficient task execution. The experimental results demonstrate the efficacy of the proposed method, achieving success rates of 99.9% for door opening, 95.25% for block stacking, 80.8% for square-nut assembly, and 90.9% for round-nut assembly. Overall, this method presents a promising solution to address the challenges associated with complex tasks, offering improvements over the traditional end-to-end approach.

Keywords: deep reinforcement learning; soft actor–critic; task decomposition; high-level task

Citation: Kwon, G.; Kim, B.; Kwon, N.K. Reinforcement Learning with Task Decomposition and Task-Specific Reward System for Automation of High-Level Tasks. *Biomimetics* **2024**, *9*, 196. https://doi.org/10.3390/biomimetics9040196

Academic Editor: Simon X. Yang

Received: 29 February 2024
Revised: 14 March 2024
Accepted: 20 March 2024
Published: 26 March 2024

1. Introduction

Robot manipulators are characterized by high efficiency and accuracy in performing repetitive and precise tasks, thereby augmenting their considerable potential for application across various fields such as manufacturing, logistics, and service industries [1]. Robot manipulators, constructed with a structure resembling that of the human arm, consist of joints like wrists and elbows, which are interconnected by links, enabling them to execute movements similar to those of humans [2]. The structural design of these manipulators enables a range of motion and rotation capabilities, providing both flexibility and precision. These attributes allow robot manipulators to have the ability to adapt to a wide range of human behaviors and demands. This capability is considered an essential characteristic in the field of human–robot interaction (HRI) [3]. The properties of robot manipulators can complement human work abilities and contribute to increasing not only efficiency but also safety in more complex and diverse work environments.

In general, robots exhibit superior abilities to humans in repetitive and precise tasks, whereas humans are capable of comprehensive thinking and judgment [4]. Consequently, collaboration between humans and robots not only compensates for the weaknesses of each but also maximizes their strengths, leading to mutual synergistic effects that extend the application range of robots and enhance work efficiency [5]. To enhance this human–robot synergy effect, continuous research and development has been studied, resulting in the emergence of the concept of collaborative robots (cobots) and their expansion into various fields. Unlike traditional industrial robots that are primarily used in automated manufacturing and logistics industries, cobots are increasingly being applied to a wider range of tasks due to their capability to execute high-difficulty operations [6]. Therefore, cobots are expanding into diverse fields such as food and beverages, service, and healthcare, becoming commonplace in daily life in various forms such as robots that make coffee, open doors,

or assist in medical surgeries [7,8]. Through these cases, interaction between humans and robots in unstructured environments is recognized as an important issue [9]. To implement effective interaction, it is necessary to develop various robot control algorithms that are customized to specific tasks and situations.

Meanwhile, control techniques based on system models, such as linear quadratic control [10], sliding mode control [11], computed-torque control [12], and model predictive control [13], allow for the precise prediction and control of robot movements. This requires the mathematical modeling of the system's physical and mechanical characteristics and tuning of parameters like control gains. However, it is very difficult to build a perfect mathematical model that considers all situations and the physical and dynamic characteristics of the system, and parameter tuning for optimal control performance not only takes a lot of time but also has the possibility of failure. To overcome these limitations, various control techniques, including direct teaching, teleoperation, and reinforcement learning, have been investigated. Direct teaching is a method where the user manually manipulates the position or path of the robot arm to teach the robot the desired task [14]. Teleoperation technology utilizes various interfaces such as keyboards, joysticks, and touchscreens to control the robot [15]. The advantage of these approaches is that it does not require the mathematical modeling of the system or gain tuning. However, while control methods through direct teaching or teleoperation can be effective for simple tasks, they may have limitations for complex tasks or unexpected situations, potentially requiring additional programming or human intervention [16]. To address these limitations, researchers are trying to give robots the ability to think and act on their own with active decision making. For example, technology has been developed to enable robots to make flexible and intelligent decisions, such as responding to environmental changes, recognizing obstacles, and automatically adjusting their paths as needed. In particular, reinforcement learning algorithms are one of the primary methods to achieve this objective and have recently received significant attention among researchers.

Reinforcement learning is a machine learning technique that trains an agent through trial and error to actively decide on actions that maximize cumulative rewards within an environment [17]. Through this technique, the agent learns to efficiently respond and adapt to achieve its goals in various situations. This approach is similar to how humans learn, where the agent updates its behavior through trial and error and ultimately establishes an action policy that maximizes cumulative rewards [18]. This process enables the agent to robustly respond to various environmental changes and acquire the ability to actively recognize and solve problems like humans [19]. Moreover, since reinforcement learning is achieved through interaction with the environment, it has the advantage of not requiring gain tuning or mathematical modeling. Based on these characteristics of reinforcement learning, del Real Torres et al. [20] suggest that there is potential and promise for the application of reinforcement learning algorithms in automation fields, particularly in smart factories and robotics. Despite these advantages, reinforcement learning still faces challenges in terms of adapting to complex environments, ensuring stability, and maintaining predictability, especially in complex tasks such as block stacking and pick and place during robot manipulator tasks [21].

The Hierarchical Reinforcement Learning (HRL) approach is one of the methods to overcome these challenges. HRL utilizes a hierarchical structure in the learning process to effectively solve complex and high-dimensional tasks [22]. The authors of [23] utilized the HRL approach, which divides pick-and-place tasks into three subtasks: approach, manipulate, and retract. These three subtasks are trained by the Deep Deterministic Policy Gradient (DDPG) algorithm [24] with the Hindsight Experience Replay (HER) technique [25] and are coordinated using a High-Level Choreographer (HLC). Through this, pick-and-place tasks are successfully performed. Additionally, another research article [26] decomposes pick-and-place tasks into two reaching tasks and one grasping task. The two reaching tasks are trained using soft actor–critic (SAC), and the grasping task is implemented with a simple method of applying force to the gripper. Afterward,

the pick-and-place task is successfully performed by sequentially connecting the subtasks: reaching, grasping, and reaching. Previous research applied task decomposition techniques and a task-specific reward system only to pick-and-place tasks among high-level tasks. To the best of our knowledge, no existing research has demonstrated a high success rate in the performance of the other high-level tasks through the application of task decomposition technology and a task-specific reward system, which motivates this article.

This paper proposes a reinforcement learning method based on task decomposition and a task-specific reward system for performing tasks more complex than the pick-and-place task, such as door opening, block stacking, and nut assembly. Initially, the door-opening task is subdivided into the processes of reaching–grasping–turning–pulling, the block-stacking task into reaching–grasping–reaching–putting, and the nut assembly task into reaching–aligning–reaching–grasping–assembling–putting. Among the subdivided subtasks, the grasping and putting tasks are implemented through single joint and gripper actions, while the remaining tasks establish optimal policies through the SAC algorithm and a task-specific reward system. Here, the task-specific reward system is used to increase the training speed of the agent, improve success rates, and facilitate the subsequent tasks of grasping and putting. Subsequently, by sequentially connecting the established policies, we confirm the successful execution of high-level tasks such as door opening, block stacking, and nut assembly. This demonstrates that the proposed method overcomes the limitations of the end-to-end approach and represents a useful solution for solving various complex and challenging tasks.

2. Related Work

2.1. Hierarchical Reinforcement Learning

HRL is an approach that utilizes a hierarchical structure in the training process to effectively address complex and high-dimensional tasks within the field of reinforcement learning. Traditional reinforcement learning involves an agent interacting with the environment and updating its action policy in response to rewards or penalties received as feedback. However, in complex tasks, the high dimensionality of the problem makes it challenging for agents to train and find the optimal policy. HRL addresses these issues by introducing a hierarchical structure into the decision-making process of the agent. The main idea of HRL is that, instead of learning a single complex policy for the entire task, it focuses on learning sets of subpolicies for specific aspects or subtasks. These subpolicies are organized in the hierarchical method, enabling more efficient learning and decision making. In summary, HRL is a more effective approach than traditional reinforcement learning by utilizing structured decision-making layers to solve problems arising in complex, high-dimensional tasks.

2.2. Soft Actor–Critic

SAC is an off-policy, model-free approach in reinforcement learning. This algorithm focuses on maximizing cumulative rewards while learning stochastic policies in continuous state space. It particularly shows outstanding performance in training agents with high-dimensional states and continuous action space, such as robotic arms and autonomous driving. Furthermore, by incorporating the maximum entropy method and soft Q-functions, it enables exploration to occur effectively for a variety of experiences in uncertain environments. The core principle of maximum entropy RL is to integrate the concept of entropy into the decision-making process. Therefore, maximum entropy RL aims to establish policies that not only maximize cumulative rewards over time but also ensure a diverse distribution of actions in each state. This approach achieves a harmonious balance between exploration and exploitation based on the state. A policy optimization formula of SAC is represented as follows:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t \mathrm{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,\, a_t)+\alpha H(\pi(\,\cdot\,|s_t))], \tag{1}$$

where $r(s_t, a_t)$ denotes an immediate reward function when the state is s_t and action is a_t. For a given policy π and state s_t, the term $H(\pi(\cdot|s_t))$ denotes the entropy that encourages the exploration. α is the temperature parameter that adjusts the level of randomness in the chosen policy and reflects the relative importance of the entropy component in the overall structure of the policy. This approach is adjusted according to the reward context of each state: in states with high rewards, policies with low entropy (i.e., more predictable actions) are considered sufficient, while in states with low rewards, policies with high entropy (i.e., more exploratory actions) are preferred to encourage broader exploration. Moreover, SAC employs a soft Q-function to facilitate exploration and prevent the policy from becoming overly biased, thereby adding flexibility to the decision-making process. The SAC algorithm utilizes an actor–critic structure, where the actor decides the actions of the agent based on the current state, and the critic evaluates the action determined by the current state. In the SAC research in [27], the actor and critic are implemented as deep neural networks, with the policy network functioning as the actor and the soft Q-network serving as the critic. The objective function of the critic network is defined using mean squared error (MSE) between the Q-value and the target Q-value, with the goal of minimizing this MSE. The Q-value is estimated through the soft Q-network and the target Q-value is approximated through the target soft Q-network. The formula is as follows:

$$J_Q(\theta) = E_{(s_t,\, a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,\, a_t) - \left(Q_{\overline{\theta}}(s_t, a_t)\right)\right)^2\right], \tag{2}$$

Furthermore, the parameters of the soft Q-network are denoted by θ, and the parameters of the target soft Q-network are represented by $\overline{\theta}$. The parameters $\overline{\theta}$ is updated by copying from θ at fixed intervals, a process that enhances the stability of the critic network's training. Additionally, optimization of θ employs the stochastic gradient descent (SGD) method to minimize the Bellman residual. The gradient of the objective function is expressed as follows:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma(Q_{\overline{\theta}}(s_{t+1}, a_t) - \alpha \log(\pi_\phi(a_{t+1}|s_{t+1}))))). \tag{3}$$

Next, the objective function of the actor network is expressed as Equation (4), based on the Kullback–Leibler divergence (KLD) between the entropy of the policy and the Q-value, where ϕ represents the parameters of the actor network.

$$J_\pi(\phi) = E_{s_t\sim D}[E_{a_t\sim\pi_\phi}[\alpha \log(\pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)]]. \tag{4}$$

Before the gradient is calculated, the objective function of the actor network is modified through the reparameterization trick, represented as $a_t = f_\phi(\epsilon_t; s_t)$. The policy of SAC leverages a differentiable Q-function as its target, thus enabling the attainment of low variance through the reparameterization trick. This method significantly enhances the convergence speed of policy. Furthermore, utilizing this trick leads to the reparameterization of the actor's objective function as the following form:

$$J_\pi(\phi) = E_{s_t\sim D, \epsilon_t\sim N}[\alpha \log \pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))], \tag{5}$$

Then, the actor network is updated through SGD to minimize the KLD between the entropy of the policy and the Q-values. The gradient of the objective function is as follows:

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \alpha \log(\pi_\phi(a_t|s_t)) + (\nabla_{a_t} \alpha \log(\pi_\phi(a_t|s_t)) - \nabla_{a_t} Q(s_t, a_t))\nabla_\phi f_\phi(\epsilon_t; s_t). \tag{6}$$

Through this process, the agent establishes an optimal policy that maximizes both rewards and entropy by utilizing experiences gained from the variety of actions.

3. Proposed Method: Task Decomposition for High-Level Task

Task decomposition is a technique to simplify complex high-level tasks by dividing them into simpler and easier low-level tasks. In the Robosuite environment, high-level tasks

such as door opening, block stacking, and nut assembly are decomposed into subtasks, and SAC is utilized to train agents to establish optimal policies for each decomposed subtask. Afterwards, the established subpolicies are modularized and connected sequentially to enable efficient performance of complex tasks. This method allows for the experimental validation of the performance of the agent. All tasks begin training from an initial state as shown in Figure 1.

(**a**) (**b**) (**c**)

Figure 1. The initial state of each task provided by Robosuite: (**a**) door-opening task; (**b**) block-stacking task; (**c**) nut assembly task.

3.1. Door Opening

The door-opening task is one of the common benchmark tasks in the fields of robotics and automation. The environment for the door-opening task is designed around a scenario in which a robot manipulator interacts with a door handle to open the door. The goal of this task is to enable the robot manipulator to accurately grasp the door handle and successfully open the door. In this paper, the door-opening task is divided into the following four subtasks.

- Reaching task: reaching the door handle;
- Grasping task: grasping the door handle;
- Turning task: turning the door handle to unlock the door;
- Pulling task: pulling the door open while holding the handle.

The door-opening task is performed by sequentially linking the four tasks. Except for the grasping task, the reaching, turning, and pulling tasks are trained using the SAC algorithm. The first reaching agent is trained to perform the end effector reaches the position of the door handle, which is generated randomly. The agent of the grasping is designed to perform the end effector grasps of the handle through a single gripper action without any training. The turning agent is trained to release the lock by turning the handle. Finally, the agent for the pulling is trained to pull the door. Figure 2 shows the implementation process of the door-opening task using the proposed method.

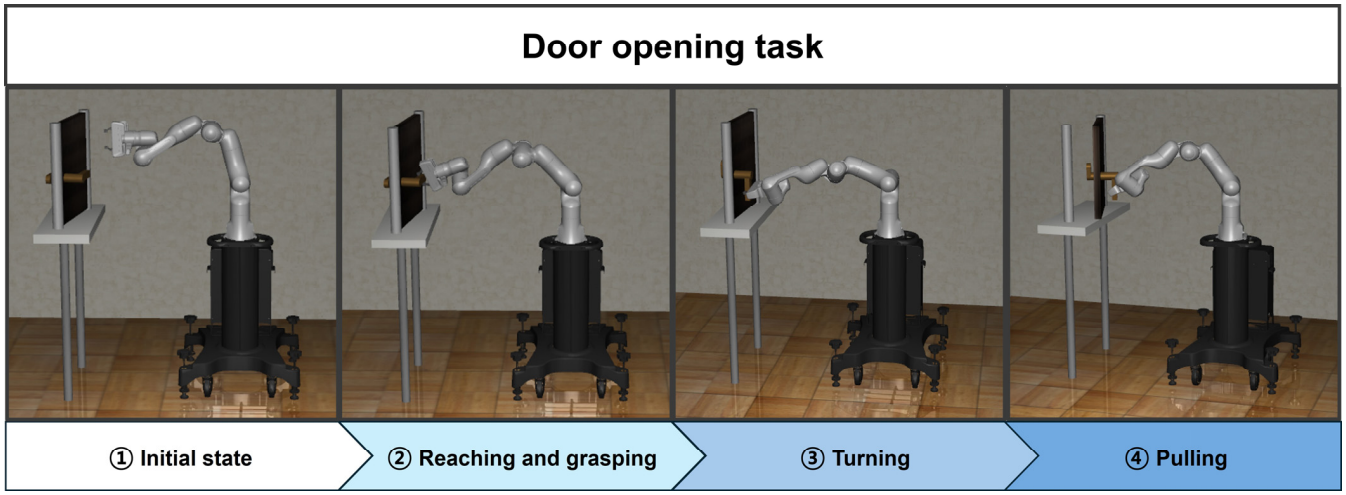


Figure 2. The overall workflow of the proposed method for the door-opening task.

3.2. Block Stacking

Block stacking is the task of stacking or loading blocks. It is a major task that can be used in various application fields such as manufacturing, logistics, and construction. The environment for the block-stacking task includes a scenario where the robot manipulator interacts with multiple blocks to stack them. The objective is for the robot manipulator to accurately manipulate and stack the blocks at the target position. In this research, the block-stacking task is divided into the following four subtasks.

- Reaching task: reaching the target block;
- Grasping task: grasping the target block;
- Reaching task: stacking the grasped target block on top of the base block;
- Putting task: putting the target block and checking the state of the stacked blocks.

The block-stacking task is implemented by sequentially linking four subtasks. During this process, two reaching agents are trained using the SAC algorithm. The agent designed for grasping is configured to enable the end effector to grasp the target block using a single gripper action, without any training. The second reaching agent is trained to reach the grasped target block to the target position, which is on top of the base block. The putting agent is designed to check the stacked state of target blocks with single joint and gripper actions, not requiring any training. Figure 3 illustrates the implementation process of the block-stacking task using the proposed method.

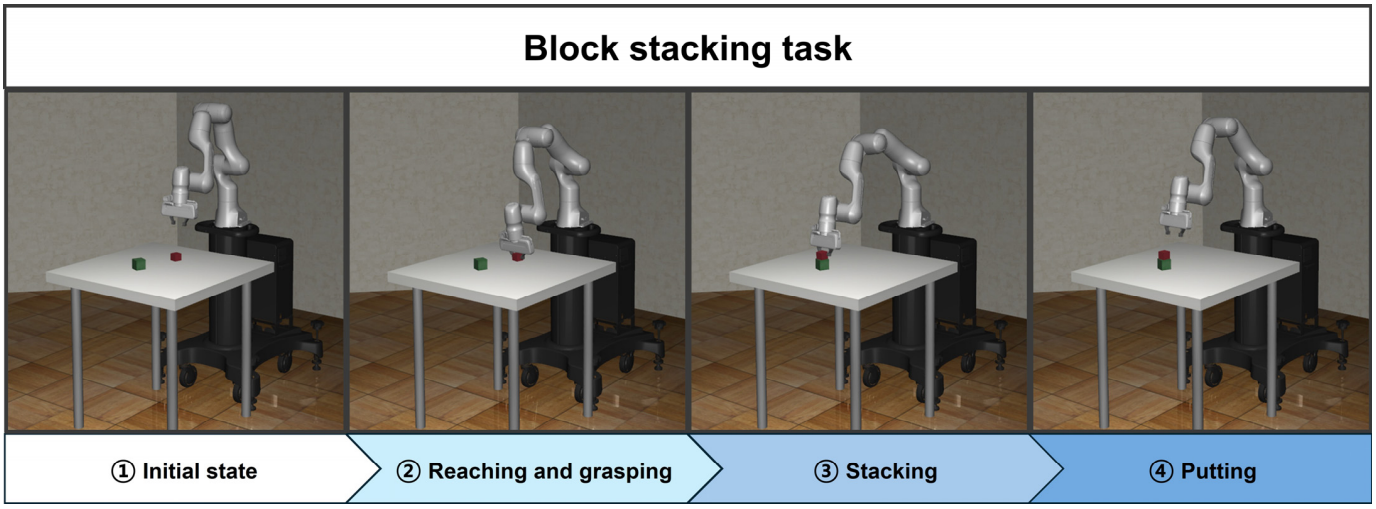


Figure 3. The overall workflow of the proposed method for the block-stacking task.

3.3. Nut Assembly

The nut assembly task is the main task for the mechanical automation of robot manipulators. The environment of the nut assembly task includes a robot manipulator, nuts, and pegs that the nuts are to be fitted onto. The goal of this task is for the robot manipulator to accurately grasp the nut, move it, and align it with the peg. Through the proposed methodology, nut assembly can be divided into six subtasks as follows:

- Reaching task: reaching above the nut handle;
- Aligning task: aligning the orientation of the nut and the end effector;
- Reaching task: reaching the handle of the nut;
- Grasping task: grasping the handle of the nut;
- Assembly task: assembling the nut onto the peg;
- Putting task: checking the peg-in-hole state.

The nut assembly task is implemented by sequentially linking these subtasks. Additionally, all subtasks except for the grasping and putting tasks are trained using SAC. The first reaching agent is trained to reach the target position set directly above the handle of the nut. The aligning agent is trained to adjust the orientation of the end effector and the nut. The second reaching agent is trained to reach the handle of the nut. The agent for the grasping is configured to enable the end effector to grasp the handle using a single gripper action, without the need for any training. The assembling agent is trained to precisely peg-in-hole the nut onto the peg. Finally, the putting agent is designed to check the state of the assembled nut through single joint and gripper actions, without any training. Figure 4 shows the process of implementing the nut assembly task using the proposed method.

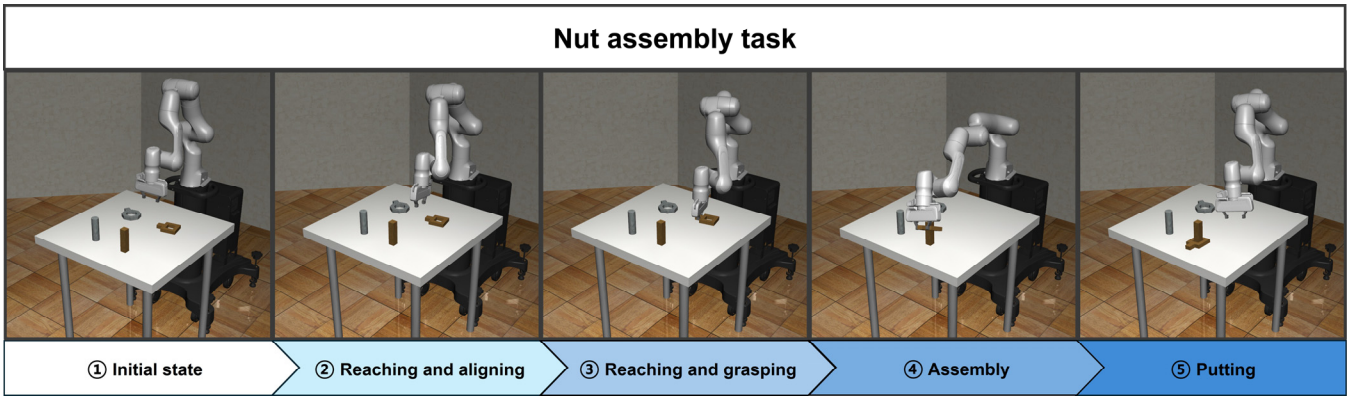


Figure 4. The overall workflow of the proposed method for the nut assembly task.

3.4. Designing States, Actions, and Task-Specific Reward System

3.4.1. States and Actions

In this subsection, we describe the state and action for each task. Table 1 details the state of the robot manipulator and the state of the object for each task. In this paper, the Panda robot manipulator is used as the agent in all experimental environments. Since the Panda robot manipulator consists of seven joints and one gripper, the action $a_t \in R^8$ is defined for all tasks as in Equation (7) and represented by eight torques τ_i as in Equation (8). The direction of motion for each joint is determined by the sign of its action value. A positive action value indicates counterclockwise rotation and a negative value indicates clockwise rotation. Similarly, the action value of the gripper is categorized into opening and closing operations based on its sign. A positive action value corresponds to a closing operation, while a negative value corresponds to an opening operation. Additionally, the number of states varies depending on the task, with 46 for door opening, 51 for block stacking, and 57 for nut assembly.

$$a_t = [\tau_{1_t}, \tau_{2_t}, \tau_{3_t}, \tau_{4_t}, \tau_{5_t}, \tau_{6_t}, \tau_{7_t}, \tau_{8_t}], \quad (7)$$

$$-1 \leq \tau_i \leq 1, \forall i \in 1, 2, \cdots, 8. \quad (8)$$

First, the definition of the state that is commonly utilized across all tasks is as follows: the state of the robot manipulator's end effector, which is denoted as $S_{ee} = [P_{ee}, Q_{ee}] \in R^7$, comprises its current position and orientation. Here, the position is represented by $P_{ee} \in R^3$, and the orientation is expressed in quaternion form as $Q_{ee} \in R^4$. The cosine and sine values of the robot manipulator's joint angles are represented by $\Theta_{cos} \in R^7$ and $\Theta_{sin} \in R^7$. The joint velocities are denoted by $\Theta_{vel} \in R^7$. Lastly, the position and velocity of the manipulator's gripper are indicated by $S_g \in R^4$. In this study, both the initial position of the robot manipulator and the position of the generated object are defined within the Cartesian coordinate system.

Table 1. This table represents the state of the robot manipulator commonly used for all tasks and the state of the object used for each task.

Posture of the Robot Manipulator		
S_{ee}: position and orientation of the end effector Θ_{cos}: cosine values of the joint angles Θ_{sin}: sine values of the joint angles Θ_{vel}: joint velocity S_g: position and velocity of the gripper		
Door opening	Block stacking	Nut assembly
P_h: door handle position $\widetilde{P}_h$: positional difference between handle and end effector P_d: door position $\widetilde{P}_d$: positional difference between door and end effector Θ_i: hinge angle Θ_h: handle angle	S_b: position and orientation of the target block $\widetilde{P}_b$: positional difference between target block and end effector P_s: position and orientation of the base block $\widetilde{P}_s$: positional difference between base block and end effector $\widetilde{P}_o$: positional difference between target block and base block	S_n: position and orientation of the nut $\widetilde{S}_n$: positional difference between nut and end effector P_{nh}: nut handle position $\widetilde{P}_{nh}$: positional difference between nut handle and end effector P_p: peg position $\widetilde{P}_p$: positional difference between nut and peg

Next, the definition of the state used for each task is as follows: First, in the door-opening task, $P_h \in R^3$ denotes the position of the door handle that the robot manipulator aims to reach. The positional difference between the end effector and the door handle is denoted as $\widetilde{P}_h = P_{ee} - P_h$. Additionally, $P_d \in R^3$ represents the position of the door. The positional difference between the end effector and the door is represented by $\widetilde{P}_s = P_{ee} - P_d$. Moreover, Θ_i and Θ_h are the angles of the door's hinge and handle, both expressed in radians. Second, in the block-stacking task, $S_b = [P_b, Q_b] \in R^7$ is comprised of the position and orientation of the target block. Here, $P_b \in R^3$ represents the position of the target block, which is the objective for the first reaching task, and $Q_b \in \mathbb{R}^4$ denotes the orientation of the target block, which is expressed in quaternion form. $\widetilde{P}_b = P_{ee} - P_b$ means the positional difference between the end effector and the target block. $P_s \in R^3$ denotes the position of another block that is designated for stacking, termed the base block. The difference in position between the end effector and base block is expressed as $\widetilde{P}_s = P_{ee} - P_s$. The positional difference between the target block, which is the objective for the first reaching task, and the base block is denoted as $\widetilde{P}_o = P_b - P_s$. Finally, in the task of nut assembly, $S_n = [P_n, Q_n] \in R^7$ is a vector consisting of the nut's current position $P_n \in R^3$ and orientation $Q_n \in R^4$. The difference in position and orientation between the end effector and the nut is denoted by $\widetilde{S}_n = S_{ee} - S_n$. $P_{nh} \in R^3$ indicates the position of the nut handle, which is to be grasped next. The positional difference between the end effector and the nut handle is expressed as $\widetilde{P}_{nh} = P_{ee} - P_{nh}$. $P_p \in R^3$ is the position of the peg designed to fit the nut. The positional difference between the nut and the peg is represented by $\widetilde{P}_p = P_n - P_p$.

3.4.2. Task-Specific Reward System for Reaching

The task-specific reward system for the reaching task is designed by considering the difference between the end effector's position and the target position, as well as the energy consumption of the robot manipulator. It is defined as $r = r_p + r_e$, where

$$r_p = P_e^T K P_e,\ P_e = P_{ee} - P_t = \begin{bmatrix} x_{ee} - x_t \\ y_{ee} - y_t \\ z_{ee} - z_t \end{bmatrix},\ K = \begin{bmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{bmatrix},\ r_e = -0.00003 \times \sum_{i=1}^{8} |F_i|. \quad (9)$$

Here, Table 2 provides detailed descriptions of the target position and reward feedback parameters for each subtask. Considering the robot manipulator's initial position, feedback parameters K_x, K_y and K_z are established to precisely reach the target destinations for each task. These parameters adjust the robot's dynamic movements to achieve more precise targeting. All K feedback parameters are initially set intuitively to suit each task, and then through a process of trial and error, the parameters are adjusted to optimized values.

Table 2. This table indicates that the target position and reward feedback parameters vary for each subtask within the reaching task.

Subtask	Target Position	K_x	K_y	K_z
Reaching task in door opening	P_h: door handle	-0.5	-2	-4
First reaching task in block stacking	P_b: block	-2	-2	-1
Second reaching task in block stacking	P_s: above target block	-2	-2	-1
First reaching task in nut assembly	P_r: above nut handle	-2	-2	-1
Second reaching task in nut assembly	P_{nh}: nut handle	-2	-2	-1

In the door-opening task, to reach the door handle, the most important factor is aligning the z-position of the end effector with that of the door handle, followed by aligning the y-position. Therefore, according to priority, the absolute values of the feedback parameters are set large in the order of K_z, K_y, and K_x, as shown in Table 2. This is essential for the gripper to properly grip the door handle. Additionally, in the block-stacking and nut assembly tasks, to increase the accuracy of arriving at the (x, y) position of the object, the values of the feedback parameters K_x and K_y are increased to be higher than K_z. Aligning the z position of the end effector is important, but more emphasis should be placed on matching the (x, y) position first in reaching the task. This is because precisely aligning the (x, y) position allows for accurate object grasping through the simple gripper action. Lastly, energy reward (9) is used as the actuator force F_i that is measured at each joint through a simulation tool during the movement of the robot manipulator.

3.4.3. Task-Specific Reward System for Turning and Pulling

In this section, the task-specific reward system for the turning task is designed by considering the difference between the door handle angle and the target angle for unlocking. Similarly, for the pulling task, the task-specific reward system is defined utilizing the difference between the hinge angle and the target angle for pulling. As well as the energy consumption of the robot manipulator. It is defined as $r = r_a + r_e$, where $r_a = K_o(\Theta_o - \Theta)^2$.

Table 3 represents the target angle and feedback parameter according to the subtask. Depending on the task, the target angle is set to the unlock angle or the door-opening angle, and the feedback parameter K_o is used the same for both tasks and set to the high value to increase sensitivity to angle changes. Also, the formula for the energy reward is the same as Equation (9).

Table 3. The target angle and reward feedback parameters for each subtask within the turning and pulling task.

Subtask	Θ_o	Target Angle Θ	K_o
Turning task in door opening	Θ_h: door handle angle	$\frac{5}{12}\pi$	-10
Pulling task in door opening	Θ_i: hinge angle	$\frac{1}{6}\pi$	-10

3.4.4. Task-Specific Reward System for Aligning

The task-specific reward system for the aligning task is designed by the orientation difference between the end effector and the nut, as well as the energy consumption of the robot manipulator. It is defined as $r = r_o + r_e$ where,

$$r_o = O_e^T K O_e,\ O_e = O_{ee} - O_n = \begin{bmatrix} \theta_{ee} - \theta_n \\ \phi_{ee} - \phi_n \end{bmatrix},\ K = \begin{bmatrix} K_\theta & 0 \\ 0 & K_\phi \end{bmatrix},\ K_\theta = K_\phi = -1. \quad (10)$$

The quaternion values in the state are converted to Euler form to set the reward. The orientation of the end effector and nut is aligned using the pitch and yaw values. In this subtask, the feedback parameters K_θ and K_ϕ are both equal to -1 (10). The formula for the energy reward is the same as Equation (9).

3.4.5. Task-Specific Reward System for Assembly

The task-specific reward system for assembly task is designed by the positional difference between the nut and peg, as well as the energy consumption of the robot manipulator. It is defined as $r = r_p + r_e$ where,

$$r_p = P_e^T K P_e,\ P_e = P_n - P_p = \begin{bmatrix} x_n - x_p \\ y_n - y_p \\ z_n - z_p \end{bmatrix},\ K = \begin{bmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{bmatrix},\ K_x = K_y = -2,\ K_z = -1. \quad (11)$$

In assembly tasks, it is important to prioritize the accuracy of the $(x,\ y)$ positions not only for achieving precise peg-in-hole alignment between the nut and peg but also for facilitating the assembly process and enhancing efficiency. Therefore, to effectively perform assembly tasks, the feedback parameters K_x and K_y are set to absolute values higher than K_z (11). The formula for the energy reward is the same as Equation (9).

4. Simulation Environment—Robosuite

Robosuite [28] is a framework that is designed for the development of algorithms in robot control and reinforcement learning. This framework provides a comprehensive set of tools for controlling robot systems. Additionally, it offers high compatibility with commonly used manipulators such as the Franka Emika Panda, Kinova3, Jaco, UR5, etc. One of the strengths of Robosuite is its provision of various gripper and object models. Users can utilize Robosuite to experiment and evaluate various control and reinforcement learning algorithms in different tasks such as block stacking, pick and place, and nut assembly. The flexible modular software architecture of Robosuite is designed to allow users to easily define and extend robots and their working environments, significantly enhancing the framework's usability and applicability. Moreover, Robosuite supports a variety of machine learning techniques, not only reinforcement learning but also inverse reinforcement learning and imitation learning. These techniques facilitate the development and testing of robotic intelligence control algorithms, enabling users to address complex task environments and scenarios more effectively.

5. Experimental Results

In this section, we present the experimental results aimed at deriving the optimal policies for each agent to successfully handle complex tasks such as door opening, block stacking, and nut assembly using the proposed method. The training process for all

tasks was repeated four times. The training results are shown in figure, which averages the cumulative rewards for each episode obtained over four trials, and then shows the convergence of rewards and policy optimization trends through moving averages.

5.1. Door Opening

5.1.1. Reaching Agent for Approaching the Door Handle

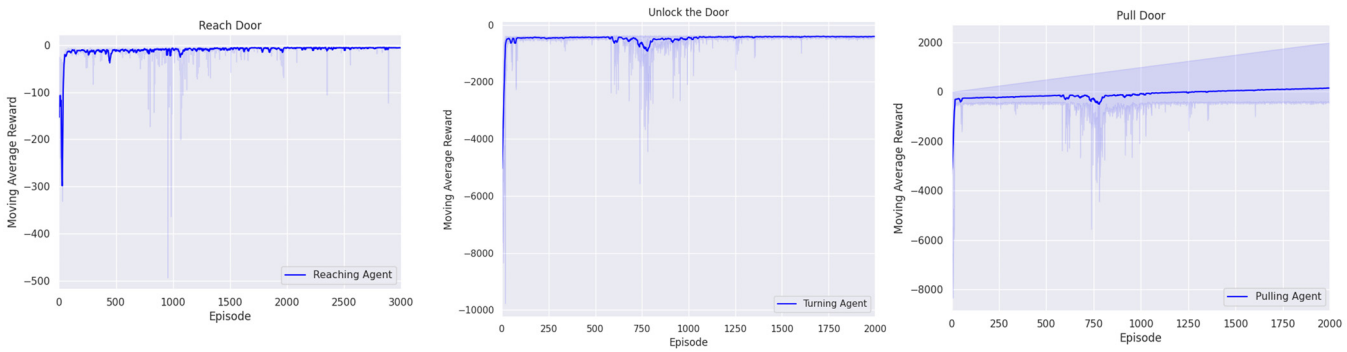
The first step in the door-opening task is to approach the door handle. The policy of the agent was established by the proposed reward system to achieve the goal of reaching the door handle from the initial position. The agent was trained for 3000 episodes with 300 steps per episode. Figure 5a illustrates that the reward converged to zero, indicating that the optimal policy for reaching the handle was established. The episode was considered a success if the end effector reached a certain range from the door handle, as follows:

$$(x_{ee} - x_h)^2 < 7.5 \times 10^{-5}, (y_{ee} - y_h)^2 < 7.5 \times 10^{-5}, (z_{ee} - z_h)^2 < 7.5 \times 10^{-5}. \quad (12)$$

The trained agent was tested for 500 episodes to reach the door handle, and these tests were repeated four times. The performance evaluation was found to be 99.95%, as shown in Table 4.

Table 4. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator reaching the door handle using the established policy.

Trial 1	Trial 2	Trial 3	Trial 4	Average
100%	100%	100%	99.8%	99.95%



(**a**) (**b**) (**c**)

Figure 5. Trend in moving average reward during door-opening task across episodes: (**a**) reaching agent; (**b**) turning agent; (**c**) pulling agent.

5.1.2. Turning Agent for Unlocking

Rotating the handle was the second step in the door-opening task. The initial state of the agent was set to the information when the end effector grasped the door handle through the trained reaching agent and a simple gripper action. In this initial state, the agent was trained over 2000 episodes, with each episode consisting of 300 steps. Additionally, the agent was trained based on the proposed reward system, which aimed to minimize the difference between the current angle of the door handle and the angle required for unlocking. The convergence of the reward to the zero signified the establishment of the optimal unlocking policy, as shown in Figure 5b. Currently, success of the episode was defined as achieving the angle that released the lock. This criterion is represented as follows:

$$\left(\Theta_h - \frac{5}{12}\pi\right)^2 < 0.0025. \quad (13)$$

The performance of trained agents was evaluated four times, with each evaluation consisting of 500 episodes. Through Table 5, it is demonstrated that each test achieved a 100% success rate, with an overall average success rate of 100%.

Table 5. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator turning the door handle using the established policy.

Trial 1	Trial 2	Trial 3	Trial 4	Average
100%	100%	100%	100%	100%

5.1.3. Pulling Agent for Opening Door

Pulling the door open was the third step in the door-opening task. At this point, the initial state of training was set by using the first agent to reach the door handle and grasp it through a simple gripper action, and then using the second agent to turn the door handle, thereby unlocking it. In this initial state, the agent was trained across 2000 episodes, each consisting of 300 steps, based on a reward system for pulling the door. This reward system was designed to minimize the difference between the hinge angle and the angle required to open the door. Figure 5c represents the establishment of the optimal policy for pulling the door, as shown by the convergence of rewards to zero throughout the training process. Additionally, the episode was considered to have achieved its goal when the hinge angle reached the angle required for the door to open, with the following condition:

$$\left(\Theta_i - \frac{1}{6}\pi\right)^2 < 0.02. \tag{14}$$

To evaluate the pulling agent, the trained agent was tested four times, each consisting of 500 episodes. The average success rate was found to be 99.9%, as shown in Table 6. Also, since the door was opened through the pulling agent, the success rate of the pulling agent can be considered the success rate of the door-opening task. Consequently, the average success rate of the door-opening task was 99.9%.

Table 6. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator pulling the door using the established policy.

Trial 1	Trial 2	Trial 3	Trial 4	Average
99.6%	100%	100%	100%	99.9%

5.2. Block Stacking

5.2.1. Reaching Agent for Approaching the Target Block

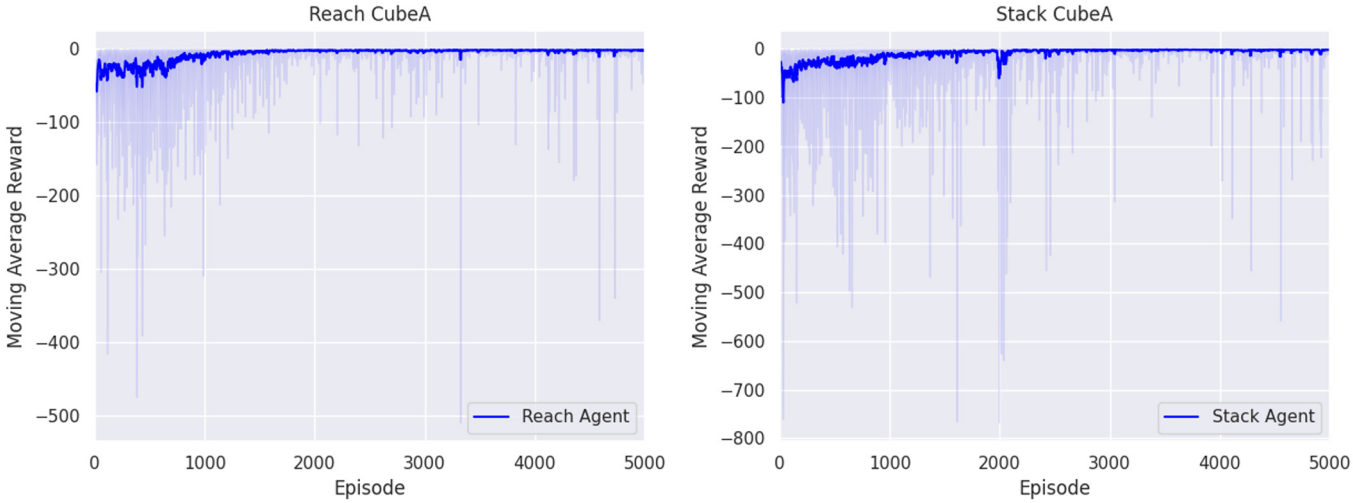
The first objective of the block-stacking task was to reach the block, which is called a target block. The position of the target block was randomly generated within predefined boundaries, and its orientation was also randomly determined by z-axis rotation. The training of the agent, aimed at minimizing the positional difference between the end effector and the target block, spanned 5000 episodes, each comprising 300 steps. Figure 6a illustrates that the reward converged to zero, indicating that the optimal policy was established. Moreover, success was defined as the end effector accurately reaching the generated target block. This criterion is expressed as follows:

$$(x_{ee} - x_b)^2 < 1.25 \times 10^{-4}, (y_{ee} - y_b)^2 < 1.25 \times 10^{-4}, (z_{ee} - z_b)^2 < 1.75 \times 10^{-4}. \tag{15}$$

The trained agent was evaluated four times, each consisting of 500 episodes. As indicated in Table 7, the average success rate for reaching the target block stood at 98.95%.

Table 7. This is the success rate for each trial and the average success rate for all trials. The test process involves the robot manipulator reaching the block using the established policy.

Trial 1	Trial 2	Trial 3	Trial 4	Average
99.0%	99.2%	98.8%	98.8%	98.95%



(**a**) (**b**)

Figure 6. Trend in moving average reward during block-stacking task across episodes: (**a**) first reaching agent; (**b**) second reaching agent.

5.2.2. Reaching Agent for Stacking the Block

Holding the target block and reaching over the base block in the target position was the second objective of the block-stacking task. In this case, the initial state was set when the end effector reached and grasped the target block by the first agent and simple gripper manipulation. From the initial state, the agent was trained across 5000 episodes, each consisting of 300 steps, utilizing the same reward system as the first reaching agent to minimize the positional difference between the target position and the end effector. Figure 6b shows the moving average trend initially fluctuating before gradually stabilizing, through which we can see that the reward converges to zero. In addition, the episode was considered a success when the end effector grasped the target block and reached the target position above the base block. This criterion is represented as follows:

$$(x_{ee} - x_s)^2 < 1.25 \times 10^{-4}, (y_{ee} - y_s)^2 < 1.25 \times 10^{-4}, (z_{ee} - z_s)^2 < 1.75 \times 10^{-4}. \quad (16)$$

The performance of the trained agent was evaluated in four trials with 500 episode tests to evaluate its performance in reaching the target position, resulting in an average success rate of 95.25%, as shown in Table 8.

Table 8. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator stacking the block using the established policy.

Trial 1	Trial 2	Trial 3	Trial 4	Average
96.0%	95.2%	94.6%	95.2%	95.25%

Additionally, the block was stacked through the second agent, and the purpose of the putting task was to verify the state of the stacked blocks. Therefore, the success rate of the second agent and the putting task was the same, so the average success rate of the block-stacking task was 95.25%.

5.3. Nut Assembly

5.3.1. Reaching Agent for Approaching Nut Handle Above

The first step of nut assembly was to approach the area above the nut handle. To achieve this, training for the agent spanned 2000 episodes, each consisting of 300 steps, and the agent was trained according to the proposed reward system. This system was designed to reach above the nut handle from the initial position by minimizing the difference in position between the target position and the end effector. Figure 7a illustrates the convergence of rewards to zero, which indicates the establishment of the optimal policy for reaching the target position. The performance of the trained agent was evaluated based on the success rate, which was determined by whether the end effector reached the target position according to the following criterion:

$$(x_{ee} - x_r)^2 < 1.25 \times 10^{-4}, (y_{ee} - y_r)^2 < 1.25 \times 10^{-4}, (z_{ee} - z_r)^2 < 1.75 \times 10^{-4}. \quad (17)$$

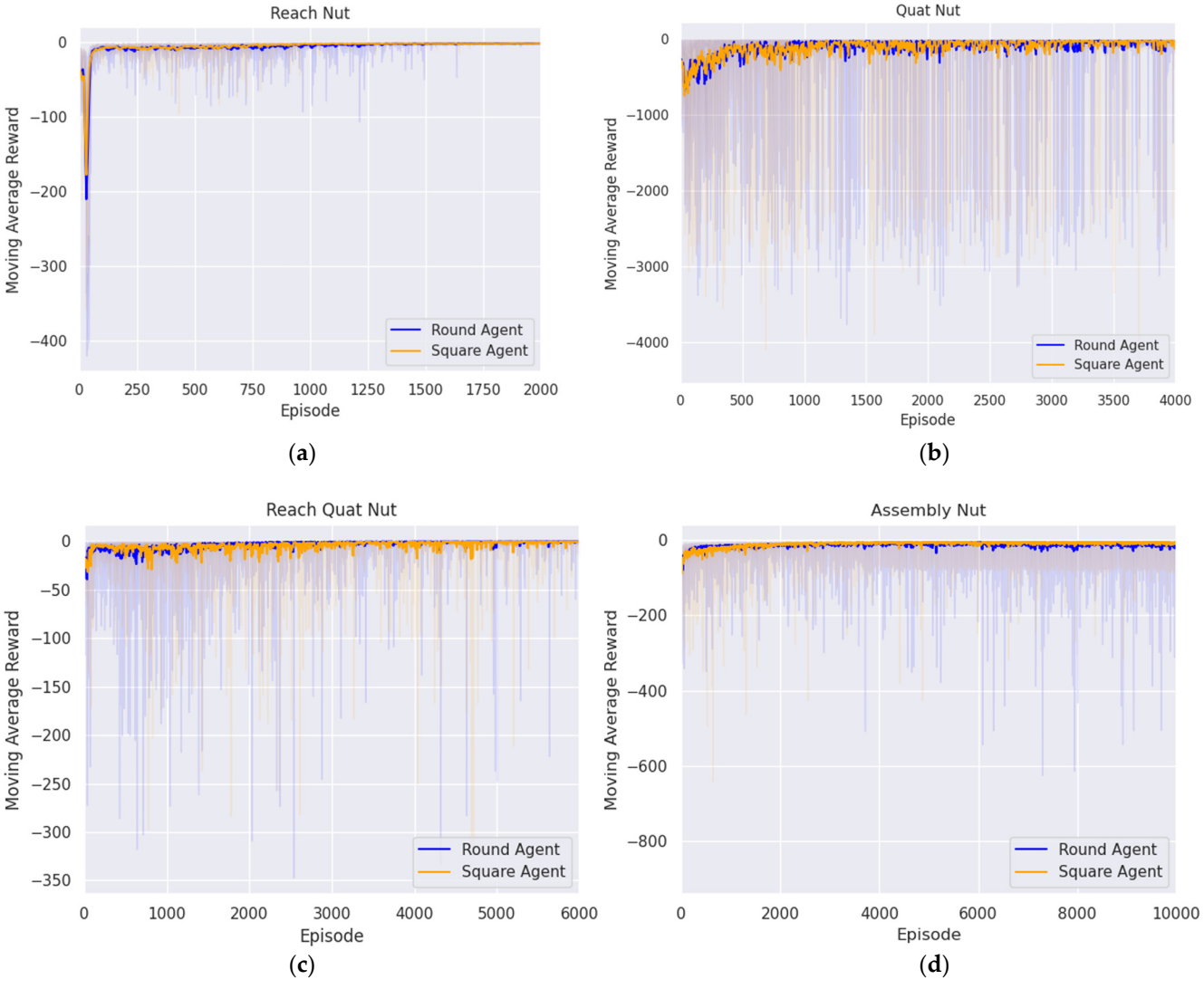


Figure 7. Trend in moving average reward during nut assembly task across episodes: (**a**) first reaching agent; (**b**) aligning agent; (**c**) second reaching agent; (**d**) assembly agent.

The trained agent's ability to reach the target location was evaluated over 500 episodes four times. As shown in Table 9, the average success rates were 99.9% for the square nut and 99.875% for the round nut.

Table 9. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator reaching above the nut handle using the established policy.

	Trial 1	Trial 2	Trial 3	Trial 4	Average
Square	99.8%	100%	100%	99.8%	99.9%
Round	99.9%	99.6%	100%	100%	99.875%

5.3.2. Aligning Agent for Rotating Nut

To match the orientation of the end effector and the nut was the second step of nut assembly. The initial position of the end effector was set above the nut handle through the first agent. In this initial state, the agent was trained across 2000 episodes, each comprising 300 steps, with the goal of minimizing the difference in pitch and yaw between the end effector and the nut. Figure 7b illustrates that the reward converges to zero, which demonstrates that the optimal policy for orientation alignment was found. The episode was considered a success when the difference in pitch and yaw between the end effector and the nut satisfied the following conditions:

$$(\theta_{ee} - \theta_n)^2 < 0.01, (\phi_{ee} - \phi_n)^2 < 0.001. \quad (18)$$

As represented in Table 10, the performance of the trained agent was evaluated over 500 episodes, repeated four times, demonstrating an average success rate of 97.45% for the square nut and 97.4% for the round nut, as indicated in Table 10.

Table 10. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator aligning the nut using the established policy.

	Trial 1	Trial 2	Trial 3	Trial 4	Average
Square	98.6%	97.6%	97.0%	98.0%	97.45%
Round	97.2%	97.8%	96.6%	98.0%	97.4%

5.3.3. Reaching Agent for Approaching Nut Handle

The third step was to approach the nut handle. Initially, the end effector reached above the nut handle via the first agent, and then, its orientation with the nut was aligned by the second agent to establish the initial state. In this initial state, the agent was trained over 6000 episodes with 300 steps per episode according to the proposed reward system to reach the nut handle. In Figure 7c, it can be seen that the reward converges to zero, which means that the optimal policy for reaching the nut handle was established. The success of the episode was determined by whether the end effector reached the nut handle based on the following condition:

$$(x_{ee} - x_{nh})^2 < 1.25 \times 10^{-4}, (y_{ee} - y_{nh})^2 < 1.25 \times 10^{-4}, (z_{ee} - z_{nh})^2 < 1.75 \times 10^{-4}. \quad (19)$$

The performance of the trained agent to reach the nut handle was evaluated in 500 episodes, repeated four times. As a result, the average success rates were 95.7% for square nut and 98.45% for round nut, as indicated in Table 11.

Table 11. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator reaching the nut handle using the established policy.

	Trial 1	Trial 2	Trial 3	Trial 4	Average
Square	96.0%	95.0%	94.2%	97.6%	95.7%
Round	98.8%	97.6%	99.0%	98.4%	98.45%

5.3.4. Assembly Agent for Peg in Hole

The fourth step of nut assembly was completing the peg-in-hole task with the nut. First, the end effector reached above the nut handle, aligned its orientation with the nut, and then reached the nut handle. This was accomplished using the first, second, and third agents, respectively. Subsequently, the nut handle was grasped through a simple gripper action. Using this state as the initial state, the agent was trained for 10,000 episodes, with 300 steps per episode, through the proposed reward system that was designed to perform peg-in-hole task with precision. Figure 7d shows that the reward converges to zero, indicating that the optimal policy was established. At this point, success was defined as precisely fitting the nut onto the peg and ensuring the difference was within the predefined threshold as follows:

$$(x_n - x_p)^2 < 1.25 \times 10^{-4}, \left(y_n - y_p\right)^2 < 1.25 \times 10^{-4}, (z_n - z_p)^2 < 1.75 \times 10^{-4}. \quad (20)$$

The agent trained to accurately perform the peg-in-hole task was tested for 500 episodes, repeated four times. Finally, Table 12 shows the success rate and average success rate for each trial. Through this, the average success rate was 80.8% for square nut and 90.9% for round nut.

Table 12. This is the success rate for each trial and the average success rate for all trials. The test process involved the robot manipulator assembly of the nut using the established policy.

	Trial 1	**Trial 2**	**Trial 3**	**Trial 4**	**Average**
Square	80.4%	81.8%	80.2%	80.8%	80.8%
Round	89.8%	90.8%	91.0%	92.0%	90.9%

Furthermore, the peg-in-hole task was achieved through the fourth agent, and the purpose of the putting task was to verify the peg-in-hole state. So, since the success rate of the fourth agent and the putting task was the same, the average success rate for the square-nut assembly was 80.8%, and that for the round-nut assembly was 90.9%.

Remark 1. *As mentioned in the experimental results, except for the first task, subsequent tasks used the agent that was trained in the previous subtask to create the initial state. For example, in the door-opening task, the process of reaching the door handle was implemented using a reaching agent, and the door handle was grasped through a simple gripper action. This state of holding the door handle was then used as the initial state for the turning agent. Another example is in the nut assembly task, where the first reaching agent was used to reach above the nut handle, and an aligning agent was used to align the orientation of the end effector with the nut. Subsequently, the nut handle was reached using the second reaching agent. The nut handle was grasped through a simple gripper action. The state of holding the nut handle was used as the initial state for the assembly agent. Thus, for tasks following the first, initial states were set using agents from the previous subtask.*

6. Discussion

Through the proposed method of this research, high success rates of 99.9%, 95.25%, 90.9%, and 80.8% were achieved in door-opening, block-stacking, and round- and square-nut assembly tasks, respectively. As a result, it can be considered that higher performance was achieved compared to the benchmarking results of Robosuite. In particular, the Robosuite benchmarking results for the block stacking and nut assembly tasks showed poor learning performance. However, applying the proposed method for the same tasks resulted in achieving high accuracy and learning rates. Moreover, comparing the proposed method to the excellent performance of the door-opening task in Robosuite indicates that the proposed method obtained similar or even superior results. This suggests that the proposed method exhibits strengths not only in complex tasks but also in less complex

tasks. This demonstrates that the proposed method is versatile and effective in performing complex and diverse tasks.

However, there are some limitations to the methodology of this study. Unlike round-nut assembly, achieving precise peg-in-hole fitting in square-nut assembly tasks requires setting a reward that considers the orientation of the square nut. However, by setting it to the same as that for the round nut, the success rate was relatively lower compared to the round-nut assembly task. In other words, the proposed method has the advantage of improving the performance of the reinforcement learning algorithm by designing a reward system for a specific task. However, it may be sensitive to changes in the environment of the task, making it difficult to generalize the reward system. Additionally, the approach of sequentially connecting optimal policies to perform complex tasks has the problem that failures of subtasks accumulate and affect the final performance.

To overcome these limitations, future research could consider training additional agents that effectively connect optimal policies for subtasks. The objective is to increase the overall task success rate and enhance flexibility and adaptability in performing complex tasks. This can be achieved by establishing advanced policies that efficiently integrate and connect tasks, rather than simply sequentially connecting policies for each subtask. Through future research, it is possible to prevent the negative impact on overall performance caused by failures in subtasks while simultaneously enhancing flexibility and adaptability, thereby further improving the ability to perform complex tasks.

7. Conclusions

This paper proposed a reinforcement learning method based on task decomposition and a task-specific reward system for performing complex high-level tasks such as door opening, block stacking, and nut assembly. Initially, the door-opening task was decomposed into subtasks of reaching–grasping–turning–pulling. The block-stacking task was divided into subtasks of reaching–grasping– reaching–putting, and the nut assembly task was divided into subtasks of reaching–aligning–reaching–grasping–assembling–putting. The grasping and putting tasks were implemented with single joint and gripper actions, while agents for the other tasks were trained using the SAC algorithm and a task-specific reward system. Here, the task-specific reward system was used to increase the learning speed of the agents, enhance their success rates, and facilitate the subsequent tasks of grasping and putting more efficiently. The proposed method was validated by successfully completing tasks with a 99.9% success rate for door opening, 95.25% for block stacking, 90.9% for round-nut assembly, and 80.8% for square-nut assembly. This demonstrates a significant improvement over existing methods in performing complex tasks. By overcoming the limitations of end-to-end approaches, it proves to be a valuable solution for solving diverse and complex tasks while also presenting new directions for future research.

Author Contributions: Conceptualization, G.K. and N.K.K.; methodology, G.K.; software, G.K.; validation, G.K., B.K. and N.K.K.; writing—original draft preparation, G.K.; writing—review and editing, B.K. and N.K.K.; visualization, G.K. and B.K.; supervision, N.K.K.; project administration, N.K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00219725), and in part by Yeungnam University through a research grant in 2021.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors express their sincere appreciation to all those who contributed to this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nguyen, H.; La, H. Review of deep reinforcement learning for robot manipulation. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 590–595.
2. Yudha, H.M.; Dewi, T.; Risma, P.; Oktarina, Y. Arm robot manipulator design and control for trajectory tracking; a review. In Proceedings of the 2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Malang, Indonesia, 16–18 October 2018; pp. 304–309.
3. Sheridan, T.B. Human–robot interaction: Status and challenges. *Hum. Factors* **2016**, *58*, 525–532. [CrossRef] [PubMed]
4. Ranz, F.; Hummel, V.; Sihn, W. Capability-based task allocation in human-robot collaboration. *Procedia Manuf.* **2017**, *9*, 182–189. [CrossRef]
5. Kyrarini, M.; Leu, A.; Ristić-Durrant, D.; Gräser, A.; Jackowski, A.; Gebhard, M.; Nelles, J.; Bröhl, C.; Brandl, C.; Mertens, A. Human-Robot Synergy for cooperative robots. *Facta Univ. Ser. Autom. Control. Robot.* **2016**, *15*, 187–204.
6. Ajoudani, A.; Zanchettin, A.M.; Ivaldi, S.; Albu-Schäffer, A.; Kosuge, K.; Khatib, O. Progress and prospects of the human–robot collaboration. *Auton. Robot.* **2018**, *42*, 957–975. [CrossRef]
7. Berezina, K.; Ciftci, O.; Cobanoglu, C. Robots, artificial intelligence, and service automation in restaurants. In *Robots, Artificial Intelligence, and Service Automation in Travel, Tourism and Hospitality*; Emerald Publishing Limited: Bingley, UK, 2019; pp. 185–219.
8. Wilson, G.; Pereyda, C.; Raghunath, N.; de la Cruz, G.; Goel, S.; Nesaei, S.; Minor, B.; Schmitter-Edgecombe, M.; Taylor, M.E.; Cook, D.J. Robot-enabled support of daily activities in smart home environments. *Cogn. Syst. Res.* **2019**, *54*, 258–272. [CrossRef] [PubMed]
9. Bonci, A.; Cen Cheng, P.D.; Indri, M.; Nabissi, G.; Sibona, F. Human-robot perception in industrial environments: A survey. *Sensors* **2021**, *21*, 1571. [CrossRef] [PubMed]
10. Kermorgant, O.; Chaumette, F. Dealing with constraints in sensor-based robot control. *IEEE Trans. Robot.* **2013**, *30*, 244–257. [CrossRef]
11. Kasera, S.; Kumar, A.; Prasad, L.B. Trajectory tracking of 3-DOF industrial robot manipulator by sliding mode control. In Proceedings of the 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), Mathura, India, 26–28 October 2017; pp. 364–369.
12. Santos, L.; Cortesão, R. Computed-torque control for robotic-assisted tele-echography based on perceived stiffness estimation. *IEEE Trans. Autom. Sci. Eng.* **2018**, *15*, 1337–1354. [CrossRef]
13. Xiao, H.; Li, Z.; Yang, C.; Zhang, L.; Yuan, P.; Ding, L.; Wang, T. Robust stabilization of a wheeled mobile robot using model predictive control based on neurodynamics optimization. *IEEE Trans. Ind. Electron.* **2016**, *64*, 505–516. [CrossRef]
14. Demura, S.; Mo, Y.; Nagahama, K.; Yamazaki, K. A trajectory modification method for tool operation based on human demonstration using MITATE technique. In Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), Kuala Lumpur, Malaysia, 12–15 December 2018; pp. 1915–1920.
15. Katyal, K.D.; Brown, C.Y.; Hechtman, S.A.; Para, M.P.; McGee, T.G.; Wolfe, K.C.; Murphy, R.J.; Kutzer, M.D.; Tunstel, E.W.; McLoughlin, M.P. Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 1874–1881.
16. Fang, M.; You, F.; Yao, R. Application of virtual reality technology (VR) in practice teaching of sports rehabilitation major. *J. Phys. Conf. Ser.* **2021**, *1852*, 042007. [CrossRef]
17. Shin, J.; Badgwell, T.A.; Liu, K.-H.; Lee, J.H. Reinforcement learning–overview of recent progress and implications for process control. *Comput. Chem. Eng.* **2019**, *127*, 282–294. [CrossRef]
18. Liu, R.; Nageotte, F.; Zanne, P.; de Mathelin, M.; Dresp-Langley, B. Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics* **2021**, *10*, 22. [CrossRef]
19. Liu, D.; Wang, Z.; Lu, B.; Cong, M.; Yu, H.; Zou, Q. A reinforcement learning-based framework for robot manipulation skill acquisition. *IEEE Access* **2020**, *8*, 108429–108437. [CrossRef]
20. del Real Torres, A.; Andreiana, D.S.; Ojeda Roldán, Á.; Hernández Bustos, A.; Acevedo Galicia, L.E. A review of deep reinforcement learning approaches for smart manufacturing in industry 4.0 and 5.0 framework. *Appl. Sci.* **2022**, *12*, 12377. [CrossRef]
21. Yang, X.; Ji, Z.; Wu, J.; Lai, Y.-K.; Wei, C.; Liu, G.; Setchi, R. Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 4727–4741. [CrossRef] [PubMed]
22. Watanabe, K.; Strong, M.; Eldar, O. SHIRO: Soft Hierarchical Reinforcement Learning. *arXiv* **2022**, arXiv:2212.12786.
23. Marzari, L.; Pore, A.; Dall'Alba, D.; Aragon-Camarasa, G.; Farinelli, A.; Fiorini, P. Towards hierarchical task decomposition using deep reinforcement learning for pick and place subtasks. In Proceedings of the 2021 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 6–10 December 2021; pp. 640–645.
24. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
25. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. arXiv:1707.01495.

26. Kim, B.; Kwon, G.; Park, C.; Kwon, N.K. The Task Decomposition and Dedicated Reward-System-Based Reinforcement Learning Algorithm for Pick-and-Place. *Biomimetics* **2023**, *8*, 240. [CrossRef] [PubMed]
27. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
28. Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; Zhu, Y. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv* **2020**, arXiv:2009.12293.