electronics

MDPI

Article

Robotic Manipulator in Dynamic Environment with SAC Combing Attention Mechanism and LSTM

Xinghong Kuang * and Sucheng Zhou

School of Engineering, Shanghai Ocean University, Shanghai 201306, China; m210811357@st.shou.edu.cn
* Correspondence: xhkuang@shou.edu.cn

Abstract: The motion planning task of the manipulator in a dynamic environment is relatively complex. This paper uses the improved Soft Actor Critic Algorithm (SAC) with the maximum entropy advantage as the benchmark algorithm to implement the motion planning of the manipulator. In order to solve the problem of insufficient robustness in dynamic environments and difficulty in adapting to environmental changes, it is proposed to combine Euclidean distance and distance difference to improve the accuracy of approaching the target. In addition, in order to solve the problem of non-stability and uncertainty of the input state in the dynamic environment, which leads to the inability to fully express the state information, we propose an attention network fused with Long Short-Term Memory (LSTM) to improve the SAC algorithm. We conducted simulation experiments and present the experimental results. The results prove that the use of fused neural network functions improved the success rate of approaching the target and improved the SAC algorithm at the same time, which improved the convergence speed, success rate, and avoidance capabilities of the algorithm.

Keywords: reinforcement learning; dynamic environment; reward function; motion planning

Citation: Kuang, X.; Zhou, S. Robotic Manipulator in Dynamic Environment with SAC Combing Attention Mechanism and LSTM. *Electronics* **2024**, *13*, 1969. https://doi.org/10.3390/electronics13101969

Academic Editors: Padma Iyenghar and Elke Pulvermüller

Received: 10 April 2024
Revised: 13 May 2024
Accepted: 15 May 2024
Published: 17 May 2024

1. Introduction

In modern smart factories, automated robots are gradually replacing manual labor and are widely used in production to implement various operations. Most industrial robots work in a specific environment without interference from the outside world. As the industrial environment becomes increasingly complex, there are also more moving obstacles in the environment, such as Automated Guided Vehicles (AGVs) or technicians [1,2].

Machinery needs to avoid obstacles while completing tasks. For static obstacles around the work area, robots can avoid the obstacles and find a suitable path most of the time [3]. Compared to environments with only static obstacles, the status of obstacles in these environments is constantly changing, and their positions are difficult to predict, making manipulator motion planning a difficulty [4,5]. Traditional algorithms are very suitable for static scenarios but not suitable for complex dynamic scenarios. For example, the motion planning algorithm Rapid Random Tree (RRT), based on probability sampling, can quickly search high-dimensional spaces, but because it explores the environment less, it performs poorly when dealing with dynamic obstacles and completing specific tasks, so effective training methods are crucial [6,7]. Tao and others introduced the Artificial Potential Field (APF) algorithm and breadth-first search methods to solve the problems of blind areas in search directions and too many search nodes in the picking manipulator, and they verified the effectiveness and repeatability of the algorithm [8]. However, based on the above-mentioned shortcomings of traditional algorithms applied to dynamic environments, research on reinforcement learning has received more attention [9].

The fast progress of deep learning and computing power has made deep reinforcement learning a reality, and it has been gradually used in games, robot navigation, industrial production, and other fields [10]. Although deep reinforcement learning algorithms can

achieve certain results in simple dynamic environments, when the environment is complex, the algorithm's perception of the environment is weakened, resulting in reduced performance. The reinforcement learning algorithm still faces various challenges, such as the algorithm's tendency to lead to the over-exploration of boundary values and slow convergence [11]. It is not enough to improve the algorithm to obtain the best motion planning results. In the face of dynamic environments, the optimization of data quality and the design of reward functions become very important for motion planning [12]. The quality of input data refers to the efficiency of the algorithm in processing the data and the expressiveness of the data [13]. In dynamic environments, the direct training of input data may lead to overfitting or slow training due to the relative complexity of environmental information.

In this paper, we first build a comprehensive reward mechanism that combines Euclidean distance and distance difference methods based on the soft action critic algorithm based on the policy and value function methods to improve the performance of the manipulator approaching the target while ensuring its avoidance of obstacles. In addition, to handle the problem of the poor utilization of data by manipulators in dynamic environments, an attention mechanism network combined with LSTM is proposed to improve the actor network and Q-value network of the SAC algorithm. The attention mechanism is skilled in extracting important information in the input state, and the LSTM network can use historical information and the current state message to improve the agent's decision-making ability [14]. Combining the advantages of the two mechanisms improves the performance of the agent's motion planning. In addition, the reward mechanism and improved algorithm were tested in two environments with different difficulties.

2. Related Work

As mentioned in the introduction, deep reinforcement learning algorithms can be used for manipulator motion planning in complex environments. The movement of the robotic arm can be considered a Markov Decision Process (MDP), so we can use tuples to describe the MDP of the manipulator [15]. The following are the reward function improvements and manipulator motion control algorithm research related to our research.

Motion planning for robotic arms usually uses either sparse rewards or dense rewards [16]. When using sparse rewards, in order to efficiently utilize the data, the recently Prioritized Experience Replay (PER) and Hindsight Experience Replay (HER) can achieve good results [17,18]. Chen et al. [19] used the shortest distance from the manipulator to the obstacle and the Euclidean distance from the end of the manipulator to the target point as the reward function. When considering that obstacle avoidance goes hand in hand with reaching, the Euclidean distance encourages the robotic arm to take the shortest path to approach the target, and the robotic arm falls into the local optimal solution. While Lindner et al. [20] used sparse rewards for the optimization of the static obstacle avoidance of robotic arms, this is not reflected in the dynamic environments of manipulators.

In order to improve the obstacle avoidance ability of the manipulator in the case of a narrow channel, Li et al. [21] proposed a reward function design using the APF algorithm, and then utilized the Deep Deterministic Policy Gradient (DDPG) algorithm for training, which greatly improved the convergence efficiency. Fu et al. [22], in order to solve the problem of low sample utilization when the DDPG algorithm is employed for manipulator motion planning, proposed a multi-experience delayed pool sampling mechanism and designed a position reward function, which effectively improves the learning efficiency. ZHENG et al. [23], in response to the problem of the hand–eye calibration of robotic arms, proposed a method combining a neural dynamics adaptive reward and action function and experimentally proved its effectiveness. Luo et al. [24], for robotic arm operation tasks using a fixed reward function when the dense reward function affects the performance, used a network reset and converted the experience into sparse rewards, greatly improving the performance. Aiming at the problem of moving obstacles in the environment, Chen et al. [25] got inspiration from the cylinder envelope method and cuboid envelope method, simplifying the connecting link of the manipulator into a straight line and simul-

taneously stacking the simplified volume of the robotic arm link onto the obstacle. This method reduced the computational difficulty. The difference between the current distance and the historical distance between the robotic arm and the obstacle is used as a reward method for obstacle avoidance, effectively avoiding collisions between the robotic arm and moving obstacles. For fixed target obstacles in the reward function, the authors used the Euclidean distance as the reward criterion. In this paper, in order to make the manipulator better track to the moving target and consider the instability effect caused by the small distance change when tracking the moving target, we were inspired by [25]. We introduce the distance difference method based on the Euclidean distance reward function to provide information about object position changes. This method is designed to ensure obstacle avoidance while improving the efficiency of completing tasks.

The existing research, on the other hand, has utilized algorithmic improvements to enhance the predictive capability of models for uncertain environments. Zhou et al. [26] combined an attention mechanism with a deep learning model based on Bidirectional Long Short Term Memory (Bi-LSTM) and used an attention mechanism module between the residual module and Bi-LSTM, achieving great results in the application of the motion trajectory prediction of anthropomorphic robotic arms. Pu et al. [27], in order to handle the problems of complex multi-agent interaction and limited communication, proposed an attention mechanism-enhanced reinforcement learning method to solve the problem of spatial information loss and the weakening of the learned spatial structure caused by the LSTM network. Xu et al. [28] designed a dynamic grasping module and a trajectory prediction module based on LSTM for dynamic grasping, which improved the dynamic grasping ability of the robotic arm. Also, in the study of the dynamic gripping problem of a robotic arm, Akinola utilized recurrent neural networks to predict the movement of objects. Park et al. [29] predicted the future position of obstacles with an LSTM structure, achieved good obstacle avoidance results, and output the hidden and memorized states of the current time step by receiving the current input as well as the hidden and memorized states of the previous time step as the input. Chen et al. [30] used Bi-LSTM combined with an attention mechanism to receive the output and achieved better results in robot-assisted outpatient services. To summarize, LSTM has good performance in object position prediction. In this paper, we were inspired by the method of Xiao H [31], where the authors used Bi-LSTM instead of a fully connected layer of the network and used an attention mechanism to optimize the Bi-LSTM for solving the under-performance and convergence problems in the optimal energy management of unmanned ships. Instead, to enhance the decision-making ability of the network of reinforcement learning algorithms, we combined an attention mechanism and an LSTM network. The LSTM network was used to optimize the input state after importance redistribution, and it was applied to the SAC algorithm to study the performance changes of manipulator motion planning in dynamic environments.

3. Method

3.1. Attn-LSTM Model Implement

Due to the influence of dynamic factors in the environment, when the state information in the complex environment is extracted for RL training through a reinforcement learning algorithm policy network, the state information in the environment changes frequently. Simple MLP networks can only perform linear fitting according to the state and environment, and their expressive ability in complex environments is weak [31]. The use of LSTMs to optimize the network has become a common method, and they are used for prediction based on historical status and current status information to improve learning and strategy effects.

However, the output value of LSTMs is the last state related to the hidden layer state of the unit. The output will be partially lost, and because of the frequent changes of some state information in complex environments, a DRL algorithm that only integrates the LSTM network cannot handle randomness well. Judgments and decisions are based on changing factors. The attention mechanism allocates attention scores to dynamic obstacles

and moving target points in the environment, allowing the model to better focus on the key information of the input state [32], such as the position and moving speed of the obstacle closest to the manipulator body and the position and moving speed of the target point. Attention is reduced to unimportant information, such as the position and speed information of obstacles far away from the body of the manipulator. In Equation (1), the attention mechanism first extracts features from the state in the complex environment, and MLP coding is used to obtain the state feature directions, e_Q, e_K, and e_V, by encoding the state of the dynamic obstacles and target points in the state information. ω_Q, ω_K and ω_V are the weight coefficients corresponding to state feature directions. The attention weight α of the input state is calculated based on the state feature vector. Finally, the attention weight and the state feature are multiplied to obtain the state vector with the highest weight H_{attn} in the input state.

$$e_i = \varphi_e(H; \omega_{e_i})|_{i=Q,K,V} \tag{1}$$

$$\alpha = softmax\left(\frac{e_Q \cdot e_{K^T}}{\sqrt{dim}}\right) \tag{2}$$

$$H_{attn} = \alpha^T \cdot e_V \tag{3}$$

The attention mechanism helps the LSTM encode the key information of the input state in the environment into the hidden state to provide global context information, which is input to the LSTM network to extract the time-domain features among the data, ensuring that the LSTM network will not change over time during long-term training and lose key information. In Figure 1, the internal structure of the LSTM network includes a forgetting gate, input gate, and output gate. The input gate is used to control the LSTM structure to receive the output data from the port of the attention mechanism. The input information of the LSTM, H_{attn}, is the information that is processed by the attention mechanism. The structure of the LSTM is shown in the figure. The following equations represent the internal rules about the output results of the LSTM in the Attn-LSTM structure:

$$\mathrm{C} = [h_{t-1}, H_{attn}] \tag{4}$$

$$f_t = \sigma\left(\omega_f \cdot \mathrm{C} + b_f\right) \tag{5}$$

$$i_t = \sigma(\omega_i \cdot \mathrm{C} + b_i) \tag{6}$$

$$o_t = \sigma(\omega_o \cdot \mathrm{C} + b_o) \tag{7}$$

$$c_t = f_t \bigodot c_{t-1} + i_t \bigodot tanh(\omega_c \cdot \mathrm{C} + b_c) \tag{8}$$

where C is intermediate state, which is concatenated by the hidden state of the previous time step, h_{t-1}, and the new state information, H_{attn}. c_{t-1} is the cell state of the previous time step. ω_f, ω_i, ω_o, and ω_c are the weights corresponding to the cell states of the oblivion gate, the input gate, the output gate, and the output. b_f, b_i, b_o, and b_c are the biases. σ and tanh represent the sigmoid activation function and tangent activation function, respectively. $\bigodot$ is a dot-multiplication operation between the matrices.

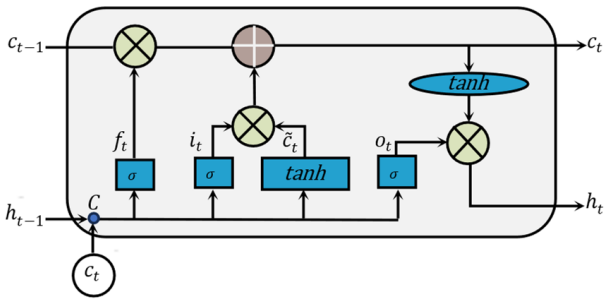


Figure 1. LSTM network construction.

As can be seen in Figure 2, the specific implementation process of the Attn-LSTM is as follows. After the dimensions of the state data in the environment are expanded, they are changed from two-dimensional data to three-dimensional data and then input into the attention mechanism to extract the state features. According to the above analysis in the attention mechanism process, the output of the attention mechanism is the input to the LSTM network for processing, and the output of the last time step of the LSTM is taken as the output of the SAC algorithm. The number of neurons in the embedding layer is 256. The number of neurons used in the single-layer LSTM is 256. And the scaling factor of the dot product attention mechanism is 0.5 to the power of the number of hidden layer neurons.

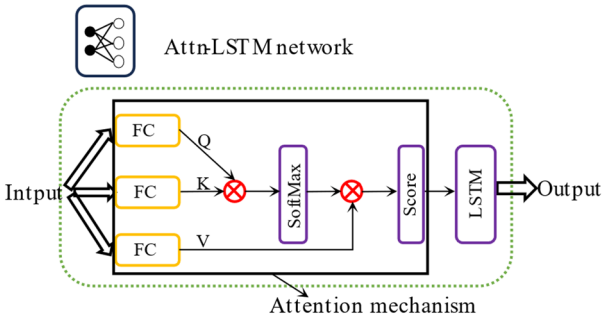


Figure 2. Attn-LSTM network construction.

3.2. Modification of Arm Manipulator Reinforcement Learning Environment

3.2.1. State Space and Action Space

For the dynamic environment of robotic arm motion planning, the state contains the end position coordinates P_{end}, the end velocity of the robotic manipulator V_{end}, the position coordinates of the target point P_{target}, the velocity of the target point V_{target}, the position coordinates P_{obsts}, and the end velocity of the obstacles V_{obsts}. The state space s can be represented in Equation (9)

$$\{P_{end}, V_{end}, P_{target}, V_{target}, P_{obsts}, V_{obsts}\} \tag{9}$$

In Equation (10), the action space of the agent is the end of the robotic arm in the x, y, and z directions. The state of the end of the robotic arm is shown in the equation [30]. When controlling the movement of the robotic arm, the three coordinates of the new action are multiplied by a factor from the original.

$$a = \{x, y, z\} \tag{10}$$

3.2.2. Rewards

The manipulator receives real-time rewards from the environment at each time step. When designing the reward function, this paper refers to the design method of the development platform in [19]. The difference is that we directly use Euclidean distance as part of the distance reward criterion. Secondly, the Euclidean distance reward mechanism is combined with the distance difference method. Our reward function is defined in Equation (11):

$$reward = \begin{cases} r_g & if\ d_t < \delta \\ r_c & if\ d_o < 0.01 \\ r_t + r_{diff} + r_o & else \end{cases} \tag{11}$$

When the distance d_t between the end of the manipulator and the target point is less than the threshold δ_1, a positive reward r_g is given. When the minimum distance between the manipulator and the obstacle is less than 0.01, a negative reward r_c is given. d_o is the shortest distance between the manipulator and the obstacle. r_o is the reward function between the robotic arm and the obstacle, as shown in Equation (12). r_t and r_{diff} is the

Euclidean distance reward and distance difference reward between the manipulator and the target point. The expression is as follows in Equations (13) and (14):

$$r_o = w_2\left(\frac{1}{1+d_o}\right)^p \tag{12}$$

$$r_t = w_1 d_t \tag{13}$$

$$r_{diff} = \omega_3 \cdot \left(d_{t-1} - d_t\right) \tag{14}$$

3.3. Soft Actor–Critic-Based Path Planning for Uncertain Environments with Attn-LSTM

When using the SAC algorithm for path planning, the initial state is first obtained from the environment, and actions are randomly sampled from the environment. The current action is fed back to the environment to obtain the reward value and the next state corresponding to the action. The actor network and critic network use the Attn--LSTM structure to predict actions and values, respectively.

Each trajectory defined consists of a state, an action, a reward, and the next state in order. Training starts when the batch size is filled. The manipulator interacts with the environment through the trajectory to update the agent action network. The execution of the action will cause the manipulator to transition into another state. The environment will provide feedback to the robot's actions, and the robot will receive rewards. The standard of the reward is determined by the reward function in the environment, including the completion of the task and the stability of the movement. Through continuous trial and error, the robotic arm learns optimization strategies from historical experience to maximize the cumulative reward value and obtain the optimal strategy. Compared to the DRL algorithm with a deterministic policy, the SAC algorithm avoids the problem of the excessive exploration of boundary values by the deterministic policy algorithm. While maximizing the reward value and entropy accumulated in the future, the SAC algorithm adopts a random strategy to make the strategy as random as possible and to prevent the manipulator from repeatedly exploring an action.

The Attn-LSTM network is used in the policy network and the soft Q network for the action and value prediction, respectively. Figure 3 shows the structures and sizes of the neural networks used in this paper for reinforcement learning. The structures of the policy network and the value network are the same. The difference is that the input of the policy network is the state, the dimension is batch*30, and the output of the network is an action vector of size 3. The input of the soft Q network is the splicing vector of the state and action, the dimension is batch*33, and the network output is a numerical value.

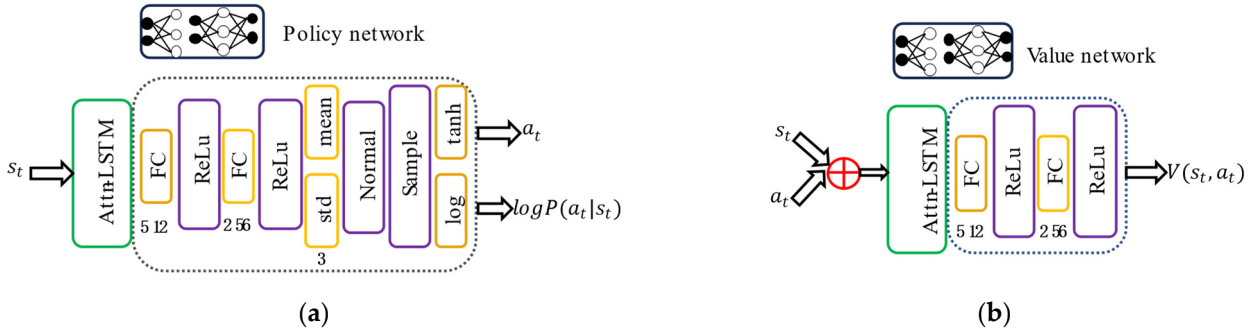


Figure 3. The structures of (**a**) the strategy network and (**b**) the value network. Both the value network and the strategy network are composed of an Attn-LSTM structure and MLP (multilayer perceptron).

Each trajectory defined consists of a state, an action, a reward, and the next state in order. Training starts when the batch size is filled and the manipulator interacts with the environment through the trajectory to update the agent action network. The execution of an action causes the manipulator to transition to another state. The environment provides

feedback on the robot's actions, and the robot receives rewards, which are determined by a reward function in the environment, including task completion and the stability of movement. The robotic arm learns to optimize its strategy through trial and error and historical experience to maximize the cumulative reward value, thus obtaining the optimal strategy.

Compared to the DRL algorithm with deterministic strategies, the SAC algorithm avoids the problem of over-exploring boundary values in the deterministic strategy algorithm. The structure diagram of the improved SAC algorithm is shown in Figure 4. While maximizing the future accumulated reward value and entropy, the SAC algorithm employs a stochastic strategy to make the strategy as random as possible to prevent the robotic arm from exploring an action repeatedly.

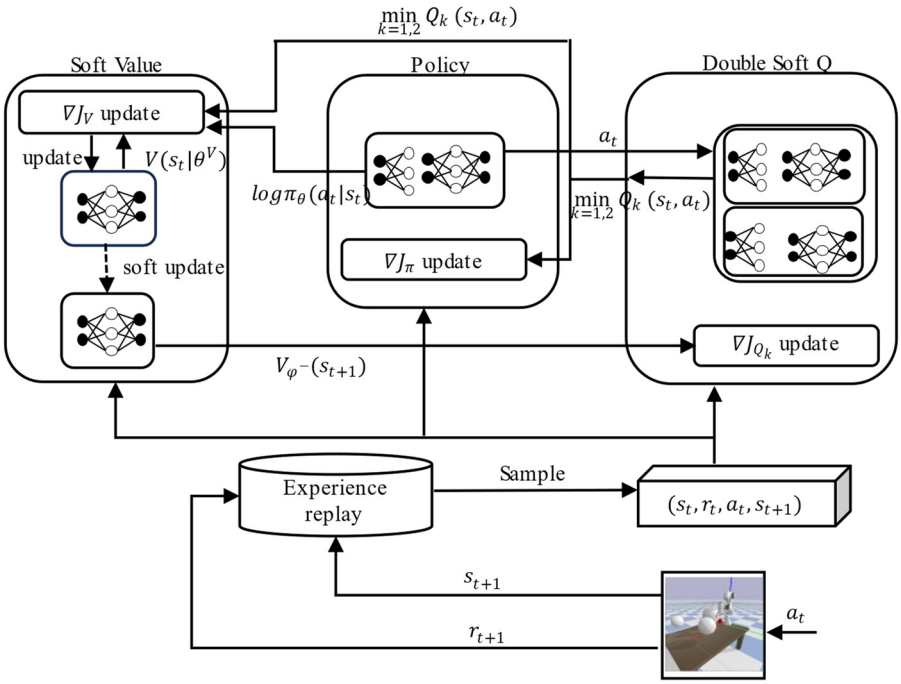


Figure 4. SAC-based path planning algorithm with Attn-LSTM(AL-SAC).

SAC aims to maximize the reward value and entropy accumulated in the future, making the strategy as random as possible and preventing the robotic arm from exploring a movement repeatedly [33]. SAC contains a policy network, $\pi_\theta(a_t, s_t)$, two soft Q-networks, $Q_{1,2}(s_t, a_t)$, a soft-value network, $V_\varphi(s_t)$, and an objective-value network $V_\varphi^-(s_t)$. For high-dimensional action space environments, the actor network generates the mean and standard deviation of a Gaussian distribution. Equation (15) denotes that the policy network is updated by minimizing the Kullback–Leibler (KL) scatter and parameterized by the parameters θ by minimizing the objective function.

$$J(\pi) = E_{s_t \sim D}\left[log\pi_\theta(a_t|s_t) - \min_{k=1,2} Q_k(s_t, a_t)\right] \tag{15}$$

Equation (16) denotes that the algorithm takes the form of a double Q-network similar to that in this article, which helps to avoid overestimating inappropriate Q-values. What is important in the SAC algorithm is that the entropy regularization coefficient is used to control the degree of randomness of the optimal policy and measure the importance of entropy relative to rewards.

$$J_Q(\varnothing) = E_{(s_t, a_t) \sim D}\left[\frac{1}{2}\left(Q_{\theta=1,2}(s_t, a_t) - \left[r(s_t, a_t) + \gamma V_{\varphi^-}(s_{t+1})\right]\right)^2\right] \tag{16}$$

To find the optimal policy, Equation (17) uses a policy network, $\pi_\theta(a_t, s_t)$, a state-value function, $V_\varphi(s_t)$, an objective function, $V_\varphi^-(s_t)$, and an action-value function, $Q_{\theta 1,2}(a_t, s_t)$, and applies stochastic gradient descent to the objective function.

$$J_V(\pi) = E_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(V_{\varphi=1,2}(s_t) - E_{a_t\sim\pi_\theta}\left[\min_{k=1,2} Q_k(s_t, a_t) - log\pi_\theta(a_t|s_t)\right]\right)^2\right] \quad (17)$$

The execution of an action leads to the transition of the robotic arm to another state, which is a manifestation of robot kinematics. The environment provides feedback on the robot's actions and the robot is given a reward, which is based on criteria determined by a reward function in the environment, including task completion and the stability of movement. The robot is trained to optimize its strategy and learn from its historical experience, such that the accumulated reward value is maximized. Until a given task is reached, this is achieved by the robot through continuous interaction with the environment. The overall algorithm flow is shown in Algorithm 1.

Algorithm 1: AL-SAC

Initialize Critic network θ_1, θ_2, Actor network θ_π
Copy parameters $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$
Initialize replay buffer R
For episode $i < N$ **do**
    get initial state s_0
    $s_t = LSTM.(attention.(input))$
        select action $a_t = \pi_\theta(s_t)$
    Save $R \leftarrow (s_t, a_t, r_t, s_{t+1})$
    for $k < M$ **do**
        Sample tuples from R
        $y_i = r_i + \gamma min_{j=1,2} Q_{w_j^-}(s_{i+1}, a_{i+1}) - \alpha log\pi_\theta(a_{i+1}|s_{i+1})$
        Update each Critic network:
        Minimize loss function $L = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{w_j}(s_{i+1}, a_{i+1})\right)^2$
        Update each Actor network:
            $L_\pi(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha log\pi_\theta\left(\widetilde{a}_i\middle|s_i\right) - \min_{j=1,2} Q_{w_j}\left(s_i, \widetilde{a}_i\right)\right)$
        Update factor α
        Update target network:
            $\theta_{1,2}' \leftarrow \tau\theta_{1,2} + (1-\tau)\theta_{1,2}'$
    end for
end for

4. Experiments and Evaluation

The initialization environment of the robot is shown in Figure 5. Three obstacle balls of different sizes and a red target point were set up in the environment. While the robot avoids obstacles, the end effector should follow the moving target point. When the distance between the end effector and the moving target point remains within the set threshold, the tracking is successful.

We provided evaluation metrics for the success rate, obstacle avoidance rate, and convergence speed. The maximum number of running steps set in the environment was 100, and the environment was re-initialized when the robotic arm took an action that reached 100 steps. The success rate refers to the probability that the distance between the end of the manipulator and the moving target in the environment is achieved within the set threshold value in 100 steps when the algorithm is trained for one round, as shown in Equation (18). In Equation (19), the reach counters are the number of times that the distance between the manipulator arm and the target is less than the threshold value in 100 steps. And the safety rate refers to the probability that the robotic arm does not collide with the obstacle within 100 steps, as shown in the following equation, where the collision counters

are the number of times the robotic arm collides with the obstacle within 100 steps. To make the success rate smoother, we calculated the average success rate every 20 training rounds. The speed of convergence was measured by comparing the difference in the number of rounds from the start of the training to the number of rounds when the algorithm reached a steady state The lower the number of rounds, the faster the convergence.

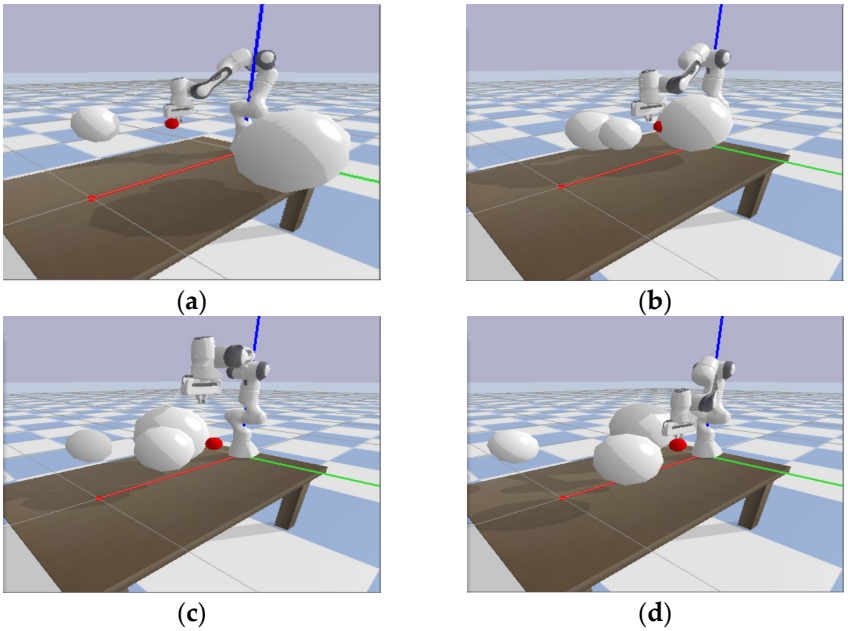


Figure 5. Simulation in the environment. (**a**) The manipulator reaches the target point on the left side of the obstacle; (**b**) the obstacle gradually approaches the manipulator; (**c**) the manipulator avoids the moving obstacle; (**d**) the manipulator reaches the target point on the right side.

$$success\ rate = \frac{reach\ counters}{100\ timesteps} \times 100\% \tag{18}$$

$$safety\ rate = \left(1 - \frac{collision\ counters}{100\ timesteps}\right) \times 100\% \tag{19}$$

We provided a specific description of the established environment. The agent used the Franka Panda manipulator, in which the three white obstacles had radii of 0.1, 0.2, and 0.3, the red target sphere had a radius of 0.05 and moved along the green Y-axis, and all had a speed of 0.25. The reward function, involving the distance between the manipulator arm and the obstacle balls, was calculated by the collision detection in the PyBullet physics engine function.

4.1. Experimental Parameter Setting

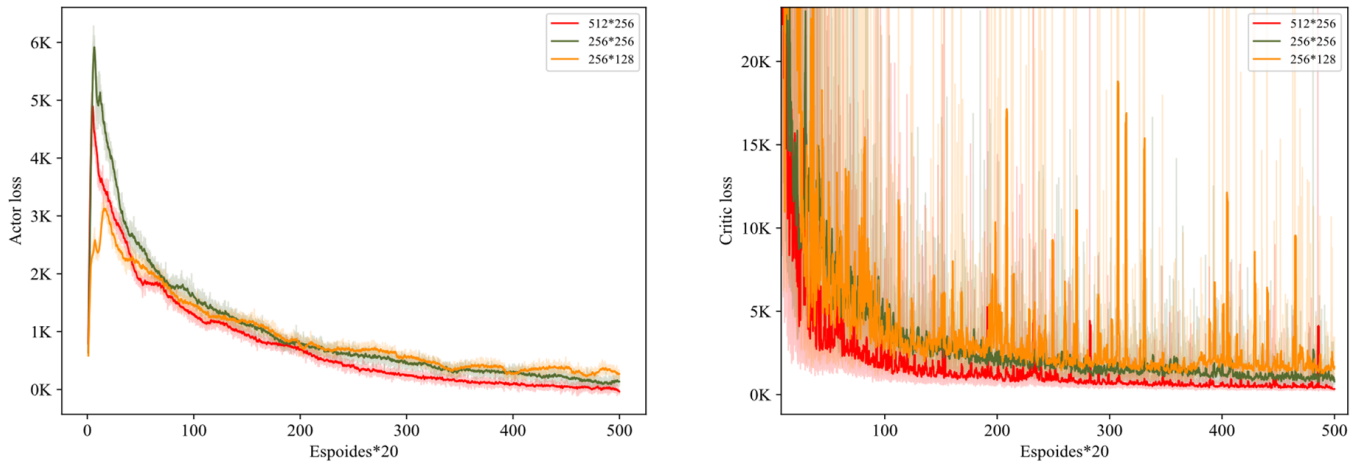
The simulation experiment was conducted using the Windows 11 and Anaconda 3 platforms, and the program used in the experiment was based on the Python 3.7, Pytorch 1.7.1 and CUD 10.1 frameworks. The five neural networks of the SAC algorithm were trained on AMD Ryzen 5 5600H and NVIDIA GeForce GTX 1650. The number of training sets was 10,000. The experiment was conducted under two conditions of different difficulty levels, requiring the robot to keep the distance between the end of the robot manipulator and the moving target point within 0.1 and 0.05, respectively, without colliding with obstacles. At the same time, the average success rate, safety rate, and average reward were calculated every 20 episodes. The hyperparameters of the SAC are shown in Table 1. The learning rates of the actor network, critic network, and value network were all 0.0005. The rewards for successful planning and collision were respectively set as $r_g = +15$ and $r_c = -30$. The

weights corresponding to the reward function were $\omega_1 = -800$, $\omega_2 = -25$, $\omega_3 = +1500$, and $p = +35$.

Table 1. Hyperparameters for SAC.

Hyperparameter	Value
Learning rate	0.0005
Batch size	256
Soft update rate	0.005
Replay memory size	10^6
Discount factor	0.99

The SAC algorithm has a large number of hyperparameters and affects the effectiveness of the algorithm training. We referred to other articles in the same field on hyperparameter settings [34–38] and gave the determinable hyperparameter settings shown in Table 1 below. The hyperparameter explored was the effect of the number of hidden layers on the performance of the algorithm. We discuss this in three cases, 512 × 256, 256 × 256, and 256 × 256. We implemented 10,000 episodes of training in the environment and recorded the loss values of the actor network and the critic network, as shown in Figure 6a,b. Also, the sliding average was recorded every 20 episodes. In Figure 6a, it can be seen that the loss curve of the 512 × 256 hidden layer structure has a smoother curve relative to the other two, while the loss value decreased faster, and the actor network had a better strategy for selecting actions. And Figure 6b shows that the Q-value network of the 512 × 256 structure converged to a relatively stable state, learning a better estimation of the value function, with no large oscillations during the training process. The average rewards corresponding to the three different hidden layer structures are shown in Figure 6, and it can be seen that the 512 × 256 structure represented by the red color had the largest reward value at convergence. Combined with the reward value in Figure 7 and the loss curve in Figure 6, it can be seen that the SAC algorithm with the 512 × 256 structure converged faster and had the best overall algorithm performance. Therefore, the SAC algorithm we used adopted the 512 × 256 hidden layer structure.



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

Figure 6. Loss curves of policy networks and Q-value networks for three different hidden layer structures when using the SAC algorithm as a benchmark algorithm. In order to clearly see the curve changes, the cubic network was taken as a localized method measure. (**a**) Loss curve of actor network; (**b**) loss curve of critic network. ('*' represents a multiplication sign).

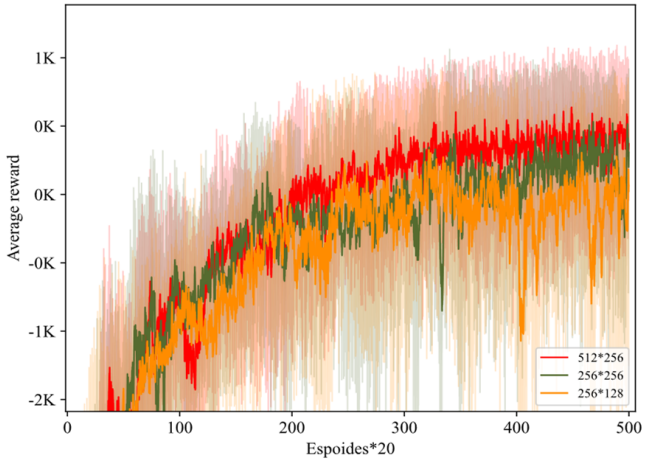


Figure 7. The SAC algorithm uses the average reward value when three hidden layer structures are used, taking a local amplification measure.

4.2. Results and Discussion

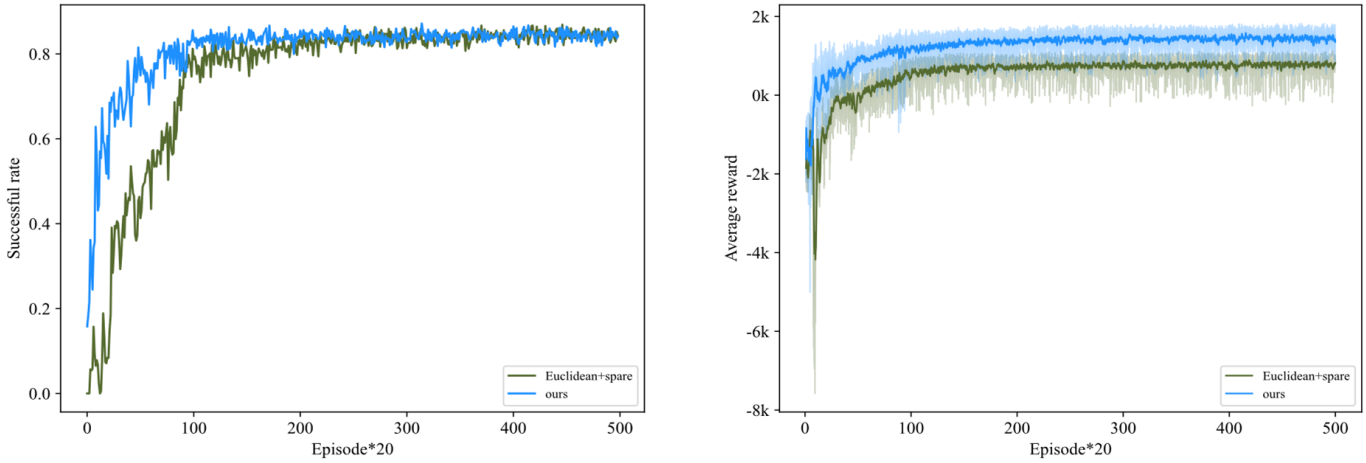
In order to verify the effectiveness of the proposed method, we expanded the experiment into two parts, namely verification of the improved reward function and effectiveness analysis of the SAC algorithm (AL-SAC) based on the Attn-LSTM network. In the first set of experiments, the SAC algorithm used the same code and the same network structure, including the activation function ReLu, the optimizer Adam, a learning rate, batch size, experience pool, discount cost, and other hyperparameters. The 'Euclidean+ sparse' reward function and the improved reward function were used for manipulator motion planning. After 10,000 episodes of algorithm training, the average success rate and reward value were compared.

In the second set of experiments, we used the improved reward function as the baseline to comparatively analyze the performance of the SAC algorithm, the LSTM-based SAC algorithm (LSTM-SAC), and the Attn-LSTM-based algorithm (AL-SAC). The performance indicators evaluated included the average reward value, average reward, and total collision probability. Both sets of experiments set up two environments with different difficulties. The distance thresholds from the end effector of the manipulator to the target point in Environment 1 (Env1) and Environment 2 (Env2) were 0.1 and 0.05. The third set of experiments, based on the above experiments, analyzed the real-time distance from the manipulator to the obstacles and target points.

4.2.1. Performance Evaluation of Reward Functions

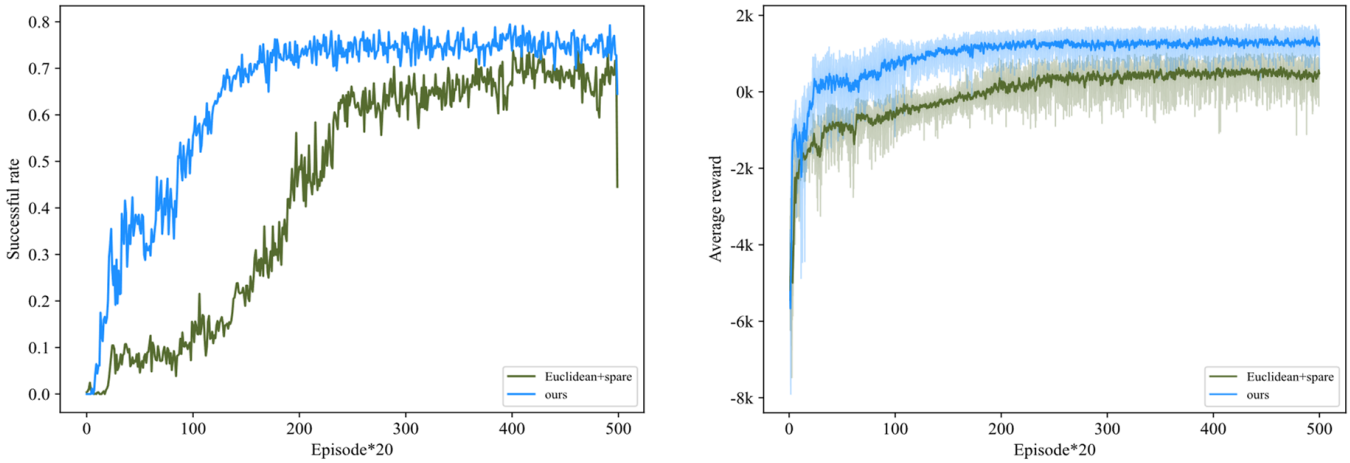
It can be seen in Figures 8a and 9a that our reward function had a high convergence speed and success rate. Compared to the SAC algorithm, which used the contrastive reward function, calculated in terms of running episodes, the convergence states were reached by 480 episodes and 2600 episodes earlier, respectively. After completing exploration and starting to converge, the success rates increased by 1% and 7.4%, respectively.

Figures 8b and 9b show the average rewards in Environment 1 and Environment 2, respectively. It can be seen that the changing trend of the average reward curve is consistent with the success rate curve. Using our algorithm's reward function of the shadow-filled area obtained a higher total reward, and the average reward curve in the early stage can better explore the environment. By combining the analysis of the success rate and reward, it is shown that the reward function combining the Euclidean distance and distance difference had a faster training speed and better success rate effect. The Euclidean distance can provide distance information for the reward function, while the distance difference method provides dynamic position change information, which synthesizes the position accuracy and dynamic adaptability of the manipulator in a dynamic environment.



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

Figure 8. Comparison of 'Euclidean + spare' and our reward function in Env1. (**a**) Average success rate; (**b**) average reward.



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

Figure 9. Comparison of 'Euclidean + spare' and our reward function in Env2. (**a**) Average success rate; (**b**) average reward.

4.2.2. Performance Evaluation of Algorithms

The hyperparameters and training conditions of the SAC algorithm part are the same. Figures 10a,b and 11a,b show the changing trends of the average success rate and average reward of the three algorithms under Environment 1 and Environment 2, respectively. Table 2 shows the number of convergence steps, the average success rate after convergence, and the average reward indicators of the three algorithms in Environment 1 and Environment 2. We compared the performance of SAC, PPO, and TD3 in a dynamic environment. It can be seen in Figure 10a that the PPO reached convergence at a much higher number of rounds relative to the SAC algorithm. TD3 maintained the lowest success rate compared to the SAC. The success rate curve of SAC has a larger slope, indicating faster convergence relative to PPO and TD3. In Figure 11a, TD3 has lower success rate and number of converged rounds than PPO in the early stage, but after reaching the converged state, the success rate is slightly higher than PPO. But both were always lower than the SAC algorithm. The latter introduces entropy regularization, which provides more choices for the action exploration of the strategy network, and the combination of the strategy and value functions further optimizes the algorithm performance. It can be seen that the SAC

algorithm training was more stable in a high-dimensional space, with a higher success rate and faster convergence speed.

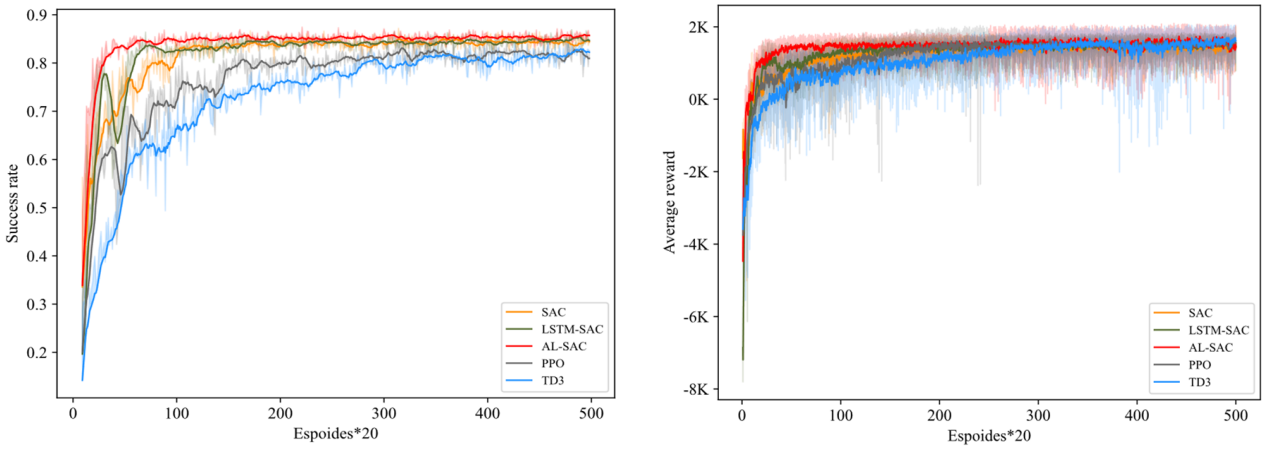


Figure 10. Comparison of three algorithms in Env1. (**a**) Average success rate; (**b**) average reward.

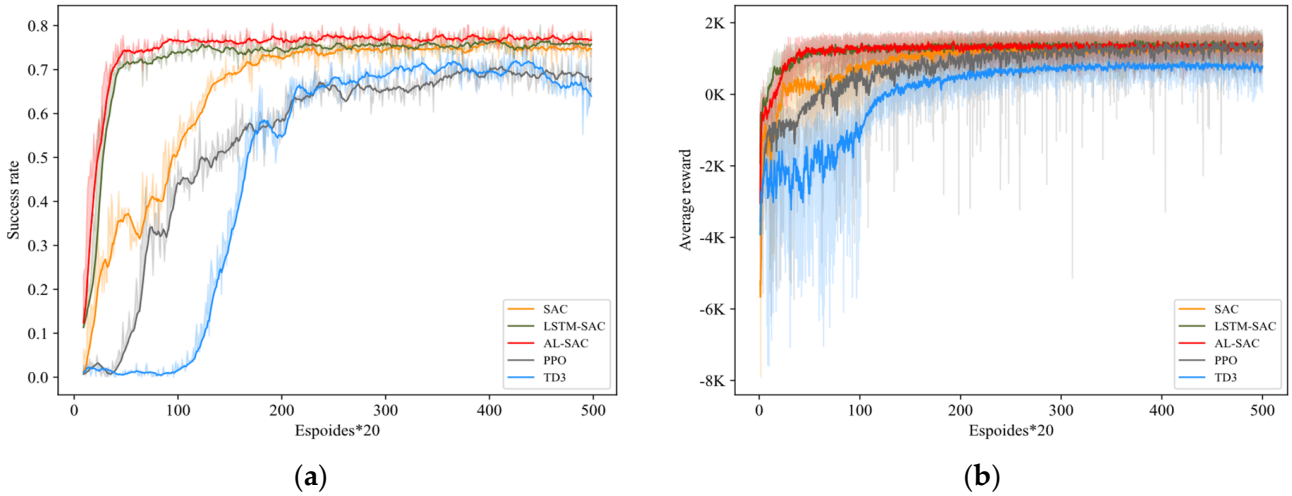


Figure 11. Comparison of three algorithms in Env2. (**a**) Average success rate; (**b**) average reward.

Table 2. Performance comparison of three algorithms.

	Env1			Env2		
	Convergence Episode	**Average Success Rate**	**Average Reward**	**Convergence Episode**	**Average Success Rate**	**Average Reward**
SAC	1920	80.7%	1388	4000	64.49%	1224.1
LSTM−SAC	1100	81.09%	1415.4	1960	71.8%	1302.45
AL−SAC	600	83.15%	1483.1	1000	74.1%	1343.5
PPO	3500	82%	1452.4	7500	69.12%	1278.08
TD3	5000	80.54%	1467.5	6000	69.3%	776.78

The AL−SAC algorithm maintained the fastest convergence speed compared to the other two algorithms; by around 600 episodes and 1000 episodes, respectively, the exploration was completed and convergence began, and the success rate after convergence was the highest. Compared to the SAC algorithm, the convergence speed and success rate of the LSTM−SAC algorithm were also improved. It completed exploration and started to

converge in 1100 episodes and 1960 episodes, respectively. And it can be seen in Figure 10b that the AL−SAC algorithm reward value curve is smoother, indicating that the algorithm training was more stable compared to the other algorithms. Comparing the AL−SAC algorithm and the LSTM−SAC algorithm, the combination of the attention mechanism and the LSTM improved the performance and efficiency of the motion planning of the manipulator in the dynamic environment to a certain extent.

Combining the analysis of the changing trends of the average success rate, average reward value, and total collision rate in Figure 12. The Attn−LSTM can significantly improve the convergence speed of manipulator motion planning and improve performance compared to the LSTM. Due to the introduction of the attention mechanism and the LSTM fusion network in the SAC algorithm, the manipulator can achieve significant distribution of complex and changeable state information during the training process and extract more key information, thereby speeding up the robotic arm and convergence speed and improving the performance of manipulator motion planning.

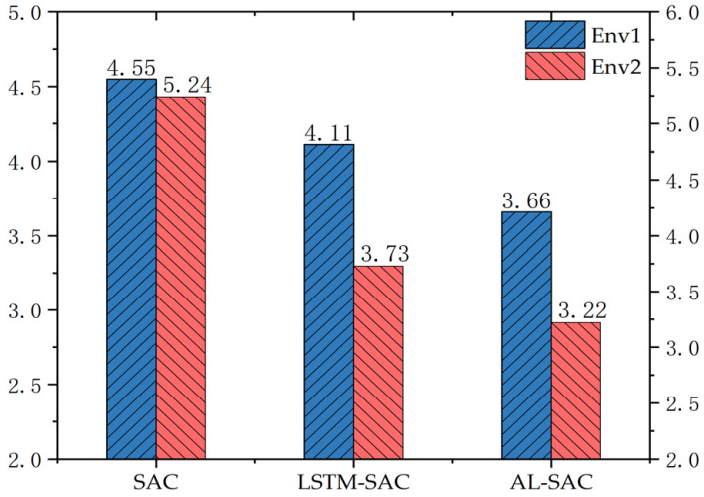
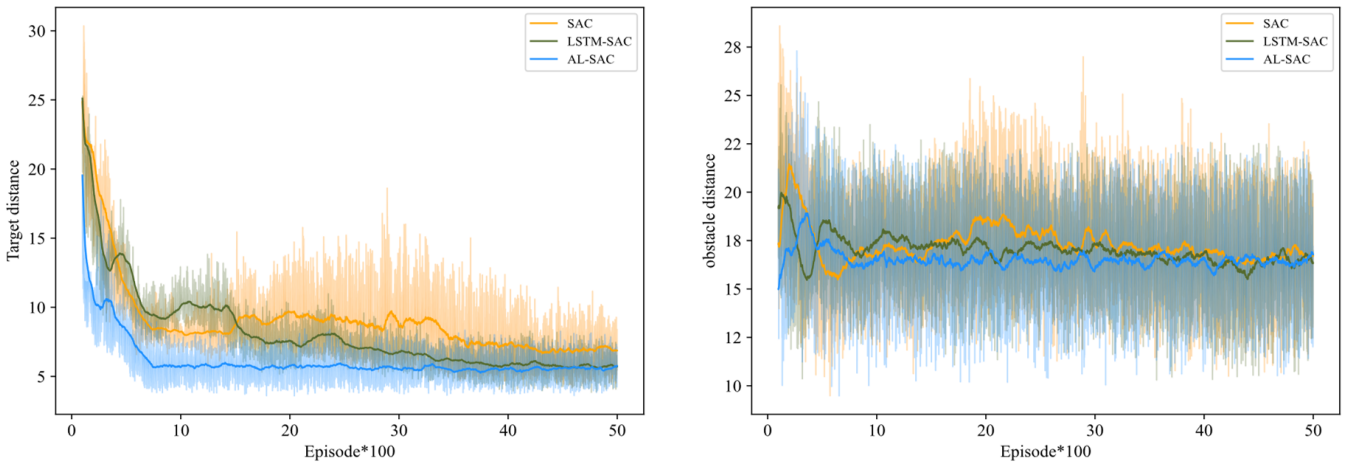


Figure 12. The performance of the three algorithms in terms of collision rate, shown in the bar chart, is the sum of all rounds of collision rate after 10,000 training sessions in Environments 1 and 2.

In order to more intuitively observe the movement status of the robotic arm in a complex environment, the test was conducted in Environment 2, which had a more difficult task. Figure 13 shows the graph of the distance between the end manipulator of the robotic arm and the target point and the shortest distance to the obstacle after 5000 rounds of training of the robotic arm. The shaded part is the recorded value of 5000 episodes, and the curve indicates the average value recorded every 100 episodes. Each distance record value represents the superposition value of the maximum 100 running steps in the environment under one episode. It can be seen in Figure 13a that the curve change slope of the AL−SAC algorithm is larger, indicating that it had a faster convergence speed, and it began to converge and reached a stable state at 800 episodes. Curve in Figure 13b shows that in the same episodes of the AL−SAC algorithm, the distance between the manipulator and the obstacle remained in a stable state, and the distance was shorter than the other two algorithms. Combined with the analysis of Experiment 2 above, this shows that the introduction of the Attn−LSTM structure into the SAC algorithm can extract more important information during the training process, accelerate the convergence speed of the agent, and improve the success rate. At the same time, the stability of algorithm performance is increased.



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

Figure 13. Comparison of distance between three algorithms in 5000 episodes of training. (**a**) The distance between the end of the manipulator and the moving target. (**b**) The shortest distance between the body of manipulator and the moving obstacle.

5. Conclusions

This paper proposes a motion planning method for a robotic arm in a complex environment based on the SAC algorithm, so that the robotic arm can better contact the target point while avoiding obstacles in an environment with dynamic obstacles and dynamic target points. In view of the shortcomings of the traditional Euclidean distance reward function, a reward function that combines distance difference and Euclidean distance is proposed to improve the motion planning efficiency of the manipulator. At the same time, according to the dynamic characteristics of the uncertain environment, an attention mechanism network fused with an LSTM is introduced to improve the effect and convergence of the SAC algorithm. The comparative analysis of the simulation experiments proves that this method can improve the success rate and convergence speed of the manipulator and reduce the probability of collision, indicating that this method has certain effectiveness and advantages. The current method has a large training time cost, and it may become possible to use transfer learning or environment sampling parallelization in future research. In addition, since the robotic arm needs further verification in a real environment, we plan to build a suitable real-life platform for experiments in the future.

Author Contributions: X.K. supervised the implementation process of algorithm; S.Z. analyzed and improved the reward function and SAC algorithm. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2022YFC3104001.

Data Availability Statement: The environment and algorithms implemented in the text will later be uploaded to https://github.com/Zhousucheng4811/paper-code (accessed on 5 May 2024).

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

References

1. Bhuiyan, T.; Kästner, L.; Hu, Y.; Kutschank, B.; Lambrecht, J. Deep-Reinforcement-Learning-based Path Planning for Industrial Robots using Distance Sensors as Observation. In Proceedings of the 2023 8th International Conference on Control and Robotics Engineering (ICCRE), Niigata, Japan, 21–23 April 2023; pp. 204–210.
2. Matulis, M.; Harvey, C. A robot arm digital twin utilising reinforcement learning. *Comput. Graph.* **2021**, *95*, 106–114. [CrossRef]
3. Said, A.; Talj, R.; Francis, C.; Shraim, H. Local trajectory planning for autonomous vehicle with static and dynamic obstacles avoidance. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; pp. 410–416.

4. Palmieri, G.; Scoccia, C. Motion planning and control of redundant manipulators for dynamical obstacle avoidance. *Machines* **2021**, *9*, 121. [CrossRef]
5. Azizi, M.R.; Rastegarpanah, A.; Stolkin, R. Motion planning and control of an omnidirectional mobile robot in dynamic environments. *Robotics* **2021**, *10*, 48. [CrossRef]
6. Ding, J.; Zhou, Y.; Huang, X.; Song, K.; Lu, S.; Wang, L. An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling. *J. Comput. Sci.* **2023**, *67*, 101937. [CrossRef]
7. Ma, H.; Meng, F.; Ye, C.; Wang, J.; Meng, M.Q.-H. Bi-Risk-RRT based efficient motion planning for autonomous ground vehicles. *IEEE Trans. Intell. Veh.* **2022**, *7*, 722–733. [CrossRef]
8. Tao, L.; Chen, C.; Pan, W. On Obstacle Avoidance Motion Planning of Picking Manipu-lator Arm based on Improved RRT. *J. Hefei Univ. (Compr. Ed.)* **2023**, *40*, 95–101+110.
9. Semnani, S.H.; Liu, H.; Everett, M.; Ruiter, A.; How, J. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3221–3226. [CrossRef]
10. Finean, M.N.; Petrović, L.; Merkt, W.; Marković, I.; Havoutis, I. Motion planning in dynamic environments using context-aware human trajectory prediction. *Robot. Auton. Syst.* **2023**, *166*, 104450. [CrossRef]
11. Zhou, C.; Huang, B.; Hassan, H.; Fränti, P. Attention-based advantage actor-critic algorithm with prioritized experience replay for complex 2-D robotic motion planning. *J. Intell. Manuf.* **2023**, *34*, 151–180. [CrossRef]
12. Huang, Z.; Wang, J.; Pi, L.; Song, X.; Yang, L. LSTM based trajectory prediction model for cyclist utilizing multiple interactions with environment. *Pattern Recognit.* **2021**, *112*, 107800. [CrossRef]
13. Shaili, M.; Anuja, A. A Huber reward function-driven deep reinforcement learning solution for cart-pole balancing problem. *Neural Comput. Appl.* **2023**, *35*, 16705–16722.
14. Xiong, C.; Xiong, J.; Yang, Z.; Hu, W. Path planning method for citrus picking manipulator based on deep reinforcement learning. *J. South China Agric. Univ.* **2023**, *44*, 473–483.
15. Malik, A.; Lischuk, Y.; Henderson, T.; Prazenica, R. A deep reinforcement-learning approach for inverse kinematics solution of a high degree of freedom robotic manipulator. *Robotics* **2022**, *11*, 44. [CrossRef]
16. Zhang, L.; Feng, Y.; Wang, R.; Xu, Y.; Xu, N.; Liu, Z.; Du, H. Efficient experience replay architecture for offline reinforcement learning. *Robot. Intell. Autom.* **2023**, *43*, 35–43. [CrossRef]
17. Sangiovanni, B.; Rendiniello, A.; Incremona, G.P.; Ferrara, A.; Piastra, M. Deep Reinforcement Learning for Collision Avoidance of Robotic Manipulators. In Proceedings of the 2018 European Control Conference (ECC), Limassol, Cyprus, 12–15 June 2018. [CrossRef]
18. Luo, Y.; Wang, Y.; Dong, K.; Zhang, Q.; Cheng, E.; Sun, Z.; Song, B. Relay Hindsight Experience Replay: Self-guided continual reinforcement learning for sequential object manipulation tasks with sparse rewards. *Neurocomputing* **2023**, *557*, 126620. [CrossRef]
19. Chen, L.; Jiang, Z.; Cheng, L.; Knoll, A.C.; Zhou, M. Deep reinforcement learning based trajectory planning under uncertain constraints. *Front. Neurorobotics* **2022**, *16*, 883562. [CrossRef]
20. Lindner, T.; Milecki, A. Reinforcement learning-based algorithm to avoid obstacles by the anthropomorphic robotic arm. *Appl. Sci.* **2022**, *12*, 6629. [CrossRef]
21. Li, Y.; Zhang, C.; Chai, L. Collaborative obstacle avoidance trajectory planning for mobile robotic arms based on artificial potential field DDPG algorithm. *Comput. Integr. Manuf. Syst.* **2023**, 1–15.
22. Fu, Z.; Zheng, W.; Zhang, L.; He, L.; Yuan, L.; Shao, M. Obstacle Avoidance Path Planning Method of Robotic Arm Based on MRD-DDPG. *Modul. Mach. Tool Autom. Manuf. Tech.* **2023**, *7*, 41–45. (In Chinese) [CrossRef]
23. Zheng, Z.; Yu, M.; Guo, P.; Zeng, D. Neurodynamics Adaptive Reward and Action for Hand-to-Eye Calibration with Deep Reinforcement Learning. *IEEE Access* **2023**, *11*, 60292–60304. [CrossRef]
24. Luo, Y.; Wang, Y.; Dong, K.; Liu, Y.; Sun, Z.; Zhang, Q.; Song, B. D2SR: Transferring Dense Reward Function to Sparse by Network Resetting. In Proceedings of the 2023 IEEE International Conference on Real-time Computing and Robotics (RCAR), Datong, China, 17–20 July 2023; pp. 906–911.
25. Chen, P.; Pei, J.; Lu, W.; Li, M. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing* **2022**, *497*, 64–75. [CrossRef]
26. Zhou, H.; Yang, G.; Wang, B.; Li, X.; Wang, R.; Huang, X.; Wu, H.; Wang, X.V. An attention-based deep learning approach for inertial motion recognition and estimation in human-robot collaboration. *J. Manuf. Syst.* **2023**, *67*, 97–110. [CrossRef]
27. Pu, Z.; Wang, H.; Liu, Z.; Yi, J.; Wu, S. Attention enhanced reinforcement learning for multi agent cooperation. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 8235–8249. [CrossRef] [PubMed]
28. Xu, B.; Hassan, T.; Hussain, I. Improving reinforcement learning based moving object grasping with trajectory prediction. *Intell. Serv. Robot.* **2023**, *17*, 265–276. [CrossRef]
29. Park, K.-W.; Kim, M.; Kim, J.-S.; Park, J.-H. Path planning for multi-Arm Manipulators using Soft Actor-Critic algorithm with position prediction of moving obstacles via LSTM. *Appl. Sci.* **2022**, *12*, 9837. [CrossRef]
30. Chen, C.W.; Tseng, S.P.; Kuan, T.W.; Wang, J.F. Outpatient text classification using attention-based bidirectional LSTM for robot-assisted servicing in hospital. *Information* **2020**, *11*, 106. [CrossRef]
31. Xiao, H.; Fu, L.; Shang, C.; Bao, X.; Xu, X.; Guo, W. Ship energy scheduling with DQN-CE algorithm combining bi-directional LSTM and attention mechanism. *Appl. Energy* **2023**, *347*, 121378. [CrossRef]

32. Chen, C.; Liu, Y.; Kreiss, S.; Alahi, A. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6015–6022.
33. Zhou, D.; Jia, R.; Yao, H.; Xie, M. Robotic arm motion planning based on residual reinforcement learning. In Proceedings of the 2021 13th International Conference on Computer and Automation Engineering (ICCAE), Melbourne, Australia, 20–22 March 2021; pp. 89–94.
34. Guo, N.; Li, C.; Gao, T.; Liu, G.; Li, Y.; Wang, D. A fusion method of local path planning for mobile robots based on LSTM neural network and reinforcement learning. *Math. Probl. Eng.* **2021**, *2021*, 5524232. [CrossRef]
35. Mock, J.W.; Muknahallipatna, S.S. A comparison of ppo, td3 and sac reinforcement algorithms for quadruped walking gait generation. *J. Intell. Learn. Syst. Appl.* **2023**, *15*, 36–56. [CrossRef]
36. Zhao, X.; Zhao, H.; Chen, P.; Ding, H. Model accelerated reinforcement learning for high precision robotic assembly. *Int. J. Intell. Robot. Appl.* **2020**, *4*, 202–216. [CrossRef]
37. Zhou, C.; Huang, B.; Fränti, P. Representation learning and reinforcement learning for dynamic complex motion planning system. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–15. [CrossRef] [PubMed]
38. Han, D.; Mulyana, B.; Stankovic, V.; Cheng, S. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors* **2023**, *23*, 3762. [CrossRef] [PubMed]