Received 20 May 2023, accepted 11 June 2023, date of publication 16 June 2023, date of current version 21 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3287098

RESEARCH ARTICLE

Neurodynamics Adaptive Reward and Action for Hand-to-Eye Calibration With Deep Reinforcement Learning

ZHENG ZHENG[1], MENGFEI YU[1], PENGFEI GUO[2,3], (Member, IEEE), AND DELU ZENG[4]

[1]School of Mathematics, South China University of Technology, Guangzhou 510641, China
[2]School of Computational Science, Zhongkai University of Agriculture and Engineering, Guangzhou 510115, China
[3]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[4]School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China

Corresponding author: Delu Zeng (dlzeng@scut.edu.cn)

This work was supported by the Fundamental Research Program 401 of Guangdong, China, under Grant 2020B1515310023.

ABSTRACT Calibration performed by a robotic manipulator is crucial in the field of industrial intelligent production, as it ensures precise and accurate measurements. In this paper, we present a new method for addressing the hand-to-eye calibration problem using deep reinforcement learning. Our proposed algorithm utilizes an actor-critic framework and incorporates neurodynamics adaptive reward and action functions, which allows for better convergence, reduces the dependence on the initial value, and overcomes the local convergence issues of traditional deep reinforcement learning method. Additionally, we introduce a step-wise mechanism under the guidance of the attention mechanism, and zero stability to handle the complexity of the calibration task in challenging environments. A number of experiments were conducted to demonstrate the validity of the proposed algorithm. The experimental results show that our proposed algorithm can achieve a nearly 100% success rate after training phase. Additionally, we compared our proposed algorithm with other widely used methods, such as deterministic deep policy gradient (DDPG) and soft actor-critic (SAC) to further demonstrate its effectiveness.

INDEX TERMS Calibration, deep reinforcement learning, actor-critic, neurodynamics adaptive method.

I. INTRODUCTION

Robotic manipulators are the most widely used automated mechanical devices in the field of robotics technology, finding applications in medical treatment [1], industrial manufacturing [2], military [3], semiconductor manufacturing [4], and space exploration [5]. They are particularly useful for performing dangerous tasks like carrying heavy objects and hazardous materials. Moreover, robotic manipulators guarantee speed, efficiency, and lower production costs in industrial settings.

Although manipulators have been widely promoted and applied in today's industrial field, with the continuous advancement of intelligent manufacturing, the requirements for intelligent manipulators in the industrial field are also getting higher and higher. For instance, robotics production like ABB in Switzerland [6] and Kuka in Germany [7] can achieve precise motion control. However, most of these robotic manipulator applications are based on given assignments and motion trajectories and lack adaptability to the target's dynamic changes, making most such robotic manipulators gradually obsolete. Therefore, to change this situation, robotic manipulators with self-learning ability have become one of the ways to realize intelligent robotic manipulators.

In this paper, we proposed a deep reinforcement learning (DRL) method for solving calibration tasks [8]. Generally, DRL is not used for ordinary linear or nonlinear optimization problems due to computational efficiency and effectiveness issues. However, the advantage of deep reinforcement learning is that its training process is a process from scratch. Unlike other machine learning methods, such as supervised learning and unsupervised learning [9], reinforcement learning can obtain data from the environment so that its ultimate goal is not to find the law of these data but to enable agents to

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Alshabi.

maximize the rewards in the environment. Moreover, the way of thinking adopted by reinforcement learning is closer to the learning way of real life and is more intelligent than other machine learning methods. Therefore, to realize the robotic manipulator's automatic and intelligent calibration task, the deep reinforcement learning (DRL) method has been applied to robotic manipulator control.

Through experiments, we discovered that using a static reward function and action function setting was not the optimal solution for the complex calibration environment as the agent cannot accurately identify the target and achieve the expected calibration result. To alleviate this problem, we proposed a neurodynamics adaptive reward function and action function setup as a core component of our deep reinforcement learning algorithm. It is inspired by brain-like mechanisms, and zero stability [10], [11] as we want the robotic manipulator to behave like a human based on its own experience. On this basis, inspired by the problem of the bottleneck of information processing in cognitive science, we further design a stepwise reward and action function based on a neurodynamics adaptive mechanism inspired by attention methods in cognitive science. This mechanism is often referred to as the attention mechanism. The experimental results show that this method can effectively solve the calibration problem in the high-dimensional target domain. Multiple sets of experimental results on V-rep platforms show that the dynamic adaptation method can effectively solve the calibration problem of high-dimensional target domains. Additionally, we conducted a comparative analysis with other well-known and widely used algorithms in the field of deep reinforcement learning, such as DDPG and SAC algorithms, to further demonstrate the effectiveness of our proposed algorithms. Consequently, the success of our algorithm design can be viewed as a prerequisite for advancing the application of deep reinforcement learning in practical scenarios involving the UR-10 robotic manipulator.

The contributions of our study can be summarized as follows:

- We proposed a neurodynamics adaptive reward function and action function to replace the static reward function and action function setting.
- To achieve the zero stability, we further design a stepwise reward and action function based on the neurodynamics adaptive reward function and action function.

The rest of this article is organized as follows. Section II. presents the background of deep reinforcement learning, and classical hand-to-eye calibration method. Section III. presents the problem of hand-to-eye calibration tasks and the performance of static deep reinforcement learning. Section IV. demonstrates the proposed algorithm architecture for calibration. Section V. describes the experiment setting and result. Section VI. evaluates the comparison between two different reward and action function setting,and comparison analysis with DDPG, and SAC algorithms. Finally, main finding and future works are summarized in Section VII.

II. PRELIMINARY

In this section, we recall the deep reinforcement learning algorithm and demonstrate the classical hand-to-eye calibration method.

A. DEEP REINFORCEMENT LEARNING

Reinforcement learning (RL) is a subfield of machine learning allowing agents to interact with the environment without supervision. The purpose of reinforcement learning is to maximize the cumulative reward and obtain optimal behavior so that agents can fully understand the environment. Almost all reinforcement learning algorithm can be formulated as markov decision process (MDP) [12], the mathematical model for agents to evaluate the decision, where the probability distribution randomly chooses outcomes. In the markov decision process, there is a tuple $(S, A, P_M(s_{t+1}|s_t, a_t), R, \gamma, \pi)$, where S is the state space, A is the action space, $P_M(s_{t+1}|s_t, a_t)$ represents the probability distribution of transition from (s_t, a_t) to new state s_{t+1}, R is the reward function, which determined by the state-action pair (s, a), π is represent as the probability of choosing action a give by the state s, and finally, the γ is the discount factor, which can emphasize the important of current reward, and weaken the influence of future rewards on the current state. In each step of markov decision process, the agents are to seek the best possible action with respect to a policy π.

Similar to reinforcement learning, deep reinforcement learning is still formulated as a markov decision process, but it uses the neural network to output the possible action instead of selecting possible action by following the $\pi(s|a)$. One of the essential elements of deep reinforcement learning is experience replay. Due to the high correlation between the samples obtained by the algorithm, and the deep neural network requires the data to satisfy the independent and identical distribution, the algorithm cannot directly use the data in the neural network, so we use the experience replay mechanism, which allows us to build a memory to store all training samples. When the memory is full, we randomly select a small batch of samples as input to the neural network, which outputs the probability of action selection.

As the rising of the deep reinforcement learning, many algorithms with outstanding contributions, such as DQN (Deep Q-learning) [13], PG (Policy Gradient) [14], DPG (Deterministic Policy Gradient) [15], DDPG (Deep Deterministic Policy Gradient) [16], TD3 (Twin Delayed Deep Deterministic Policy Gradient) [17], PPO (Proximal Policy Optimization) [18], HER (Hindsight Experience Replay) [19], TRPO (Trust Region Policy Optimization) [20], SAC (Soft Actor Critic) [21] have been proposed by major research teams, and all these outstanding algorithms have been applied to multiple areas.

B. CLASSICAL HAND-TO-EYE CALIBRATION

Hand-eye calibration, that is, calculating the translation between the robot end-effector and the world coordinate

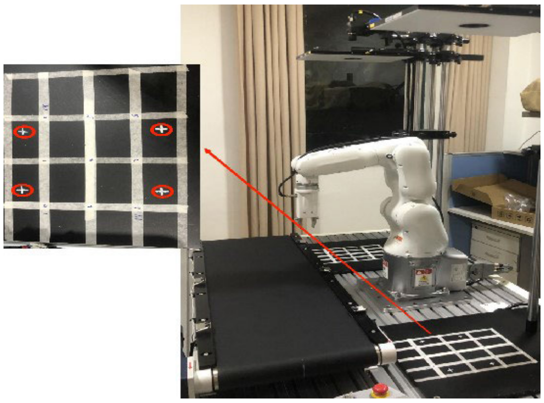

FIGURE 1. Checkerboard, and camera calibration. The red circle on the checkerboard represents the calibration target point.

system [22], is a critical prerequisite for realizing robot hand-to-eye coordination [23], and it is also the key to improving the accuracy of robot control. In the hand eye calibration task, an eye-to-hand method is a common approach. In eye-to-hand systems, the camera is fixed outside the robotic manipulator and does not move with the robotic manipulator. The most common usage for this type of calibration system is the factory production line, where most of the objects to be detected are fixed in certain areas. Although this type of approach has a wide range of applications, it could cause a sizeable relative positioning error when the camera is far away from the robotic manipulator, which limits the scope of the perception operation.

Traditionally, the most direct method for addressing the hand-to-eye calibration is to use high-precision measurement equipment to establish a world coordinate system between the camera and the robotic manipulator, and solving transformation matrix. Although high-precision measurement equipment has high calibration accuracy, it may not meet the needs of fast, labor-saving, and low-cost production in industrial applications. Because precision measuring equipment is usually expensive, manual operation is very cumbersome, and the calibration time required is also long. Alternately, a cheaper solution is to use calibration objects, such as a checkerboard plate (as shown in Fig. 1). Nonetheless, one problem to be solved, regardless of which method is used for solving hand-to-eye calibration, is to compute the relation as follows:

$$T_{base_1}^{end}\,T_{camera_1}^{base_1}\,T_{object}^{camera_1} = T_{base_2}^{end}\,T_{camera_2}^{base_2}\,T_{object}^{camera_2}, \qquad (1)$$

where $T_{base_1}^{end}$ and $T_{base_2}^{end}$ represent the transformation matrix between the robotic manipulator base and end effector, $T_{camera_1}^{base_1}$ and $T_{camera_2}^{base_2}$ represent transformation matrix between robotic manipulator base and camera, $T_{object}^{camera_1}$ and $T_{object}^{camera_2}$ represent transformation matrix between camera and calibration object (as shown in Fig. 2). This formula is obtained by moving any two poses with the robotic manipulator sandwiching the calibration object.

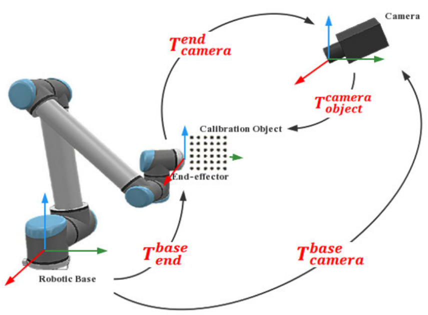

FIGURE 2. Hand-to-eye calibration diagram.

To solve this equation, the only two unknown matrices is $T_{camera_1}^{base_1}$ and $T_{camera_2}^{base_2}$, since $T_{base_1}^{end}$ and $T_{base_2}^{end}$ can be calculated by the coordinate system of the robotic manipulator itself, and $T_{object}^{camera_1}$ and $T_{object}^{camera_2}$ can be calculated by the coordinate system of the camera itself.

Traditionally, there are several approaches to solve the above equation. For example, utilizing a Lie theory in the Euclidean group has been proposed by Park and Martin to solve an unknown transformation matrix [24]. A dual quaternion method has been used in hand-to-eye calibration [25]. A novel metric on SE(3) has been proposed for optimization by Qiu, Wang and Kermani [26]. Chen and Huang [27] used an integrated two-pose calibration method to calculate parameters of both eyes to reduce the model error. Zheng and Yang [28] have created a new system to solve the unknown transformation matrix. Although these solutions were very advanced in finding the unknown transformation matrix at the time, they have gradually become unable to meet the requirements of modern industry due to their limitations, which will be discussed in details in later sections. Therefore, there is an urgent need for the industry to find a new efficient and accurate hand-to-eye calibration method.

III. TASKS FORMULATION

In this section, we demonstrate the problem of classical hand-to-eye calibration method. Moreover, we also analyze the performance of static deep reinforcement learning algorithms for solving hand-to-eye calibration.

A. PROBLEM OF CLASSICAL HAND-TO-EYE CALIBRATION METHOD

While the classical hand-to-eye calibration method is a widely used technique for determining the transformation between a robotic end-effector and an external camera, it still has its own limitations. Here are some common problems associated with the classical calibration methods.

- **Expensive experimental equipment:**In traditional calibration method, some of equipment is very expensive, which is not suitable for situations where the motion parameters are unknown or uncontrollable.
- **Time-consuming [29]:** The classical hand-to-eye calibration method requires the collection of a large number of calibration data points. This is especially challenging

in situations where frequent re-calibration is required, such as in robotics applications where the robot may move to different positions.

- **Measurement errors [30]:** The classical hand-to-eye calibration method is sensitive to measurement errors, which can arise from various factors such as noise in the collected data or inaccuracies in the geometry of the calibration object. Even with the small errors in the calibration measurements, it leads to significant errors in the estimated transformation matrix, which result in low accuracy and stability.
- **Dependence on a calibration object [31]:** The classical methods rely on the use of calibration objects with known geometry to obtain the necessary measurements. This can be a limitation in situations where calibration objects are not available or not easily accessible.
- **Limited flexibility:** The classical approach relies on a fixed geometric relationship between the end-effector and the camera, which limit its flexibility. For example, if the camera is mounted on a moving platform, the relationship between the end-effector and the camera can vary over time, which can lead to errors in the estimated transformation matrix.

B. PERFORMANCE OF STATIC DEEP REINFORCEMENT LEARNING

In order to meet the current demand for intelligent robotic manipulators in the industry, and tackle the problems of classical hand-to-eye calibration method, we attempt to use the deep reinforcement learning method to establish an accurate conversion relationship through simulation training and simplify the calibration process.

In deep reinforcement learning, one of important factors to reflect the efficiency of algorithms, which is the reward function. Generally, reward is the numerical value, which is the feedback on the action of agent in the previous steps or the whole process. We name this reward setting mechanism as static reward function. In practice, many mature deep reinforcement learning algorithms have successfully used this particular reward setting method to guide the agent activities in the environment, such as AlphaGo [32], Atari Gaming [13]. These environments treat reward as the signal to justify whether the agent has accomplished the task or not, and we named this reward setting method as static reward mechanism. For our calibration tasks, as the piercing tool attached in the end-effector of robotic manipulator, the robotic end pose remain relative quiescence. Therefore, when designing the reward function, we only used the distance between the end of the piercing tool and the target point (TTD) to define it, and we did not set the extra reward function for robotic end pose. So, the reward can be written as follow,

$$TTD = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}, \tag{2}$$

$$R = \begin{cases} m, & TTD>d \\ n, & TTD \leq d, \end{cases} \tag{3}$$

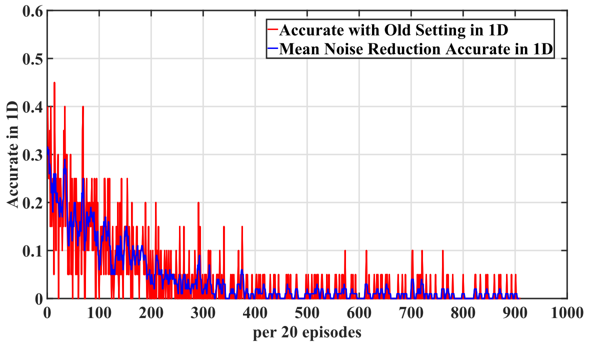

FIGURE 3. Average accurate rate synthesized by static deep reinforcement learning using reward function (3) and action function (4) for one dimensional hand-to-eye calibration problem. The red line represents the result after training without smoothing, and the blue line represents the result after smoothing using mean noise reduction technique.

where (x_1, y_1, z_1) represents the coordinate of current needle position, (x_2, y_2, z_2) represents the coordinate of target position, and R is short for the reward signal. If distance TTD less than or equal to d, it means that the agent successfully reached the target, and the reward is n. Otherwise, the reward remains m during simulation. Moreover, for our calibration task, as a piercing tool attached to the end effector of the robot manipulator, the robot end attitude remains relatively static, and the tip of the piercing tool remains perpendicular to the plane. Therefore, we do not set additional reward functions for the robot's end pose.

In addition to reward function, action function is another factors to ensure agents is moving towards target point in this experiment, and similar to static reward function, generally action function also can be seen as static action function, which is formulated as

$$A = A_c + \mathcal{N}(0, 1) * \omega, \tag{4}$$

where A_c was determined by probability distribution, which outputted by actor neural network, which will be discussed in later section. Besides, $\mathcal{N}(0, 1)$ is the normal distribution with 0 mean and 1 standard deviation, adding normally distributed noise to the action allows the agent to explore the environment more during the simulation, and ω is a factor takes values in [0,1] to ensure that output of normal noise is within $[-0.1, 0.1]$.

During the simulation, we set d to 5mm as the condition for judging the success of the task, and the reward is 2 when the TTD is less than 5mm, otherwise it remains 0. It turns out that (as shown in Fig. 3) the agent cannot even achieve the desired results in a one-dimensional target space. We realize that using this static rewards and action functions is not sufficient to justify the agent's behavior in the environment. Therefore, our paper focuses on modifying the action function and reward function to improve the average training accuracy, and we will explain the details in the later section.

IV. PROPOSED ALGORITHM ARCHITECTURE FOR CALIBRATION

In this section, we illustrate the architecture of proposed calibration algorithm through four different part, ranging from data sampling method in calibration to the parameter choices of our algorithms.

A. DATA SAMPLING METHOD

As the sample policy gradient algorithm takes too long to collect data, and the previously sampled data are not applicable once the policy parameters are updated, we work on an algorithm that can utilize the collected data to update the network. In addition, the algorithm requires preprocessing of the collected data to correct errors caused by different data distributions. We also need to add certain constraints so that the preprocessed data will not cause excessive variance when updating the policy. Meanwhile, we also need the sampled and updated networks to be in the same network because this can improve the training speed.

In order to design a model that meets our requirements, we utilized important sampling to correct errors caused by different data distributions and employed the clipping method to constrain policy iteration (as shown in Fig. 4). We chose not to use experience pooling because it can introduce additional complexity and potential performance issues. Additionally, experience pooling can lead to issues with off-policy correction, which can be computationally expensive and require additional hyperparameters. The clipping method updates the policy based on the most recent experiences gathered during training, avoiding these issues and providing a simple and effective way to limit the policy update step and improve stability during training. To do that, we have following equations:

$$J(\theta|\theta') = E[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}A^{\theta'}(s_t, a_t)], \tag{5}$$

$$J(\theta|\theta')_{clip} = E[min(J(\theta|\theta'), g(\epsilon, A)], \tag{6}$$

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, & A \geq 0, \\ (1-\epsilon)A, & A < 0, \end{cases} \tag{7}$$

where $J(\theta', \theta)$ represents the object loss function, A is advantage function, and ϵ is the numerical number of clipping size. In equation (7), it means that when advantage function A is larger than 0, the agents should increase the probability of action selection under the probability distribution $\pi_\theta(a_t|s_t)$ while limiting the magnitude of the increase. The advantage of the clipping method is to ensure that policy $\pi_{\theta'}$ will not be far from π_θ, so when the action output by the actor network is in a good direction, the clipping method will limit its excessive updating in the good direction. When the action is not good, the clipping method will limit its excessive updates in the wrong direction. Thus, the purpose of limiting the swinging range of the robotic manipulator is too large or too small, and speeding up the learning speed can be achieved.

B. ACTOR-CRITIC MODEL FOR HAND-TO-EYE CALIBRATION

With the important sampling method to preprocessing data, the calibration algorithm can be described as follow.

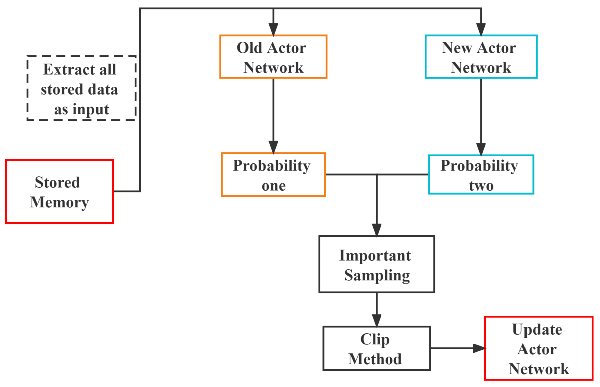

FIGURE 4. Important sampling and clipping for proposed calibration algorithm.

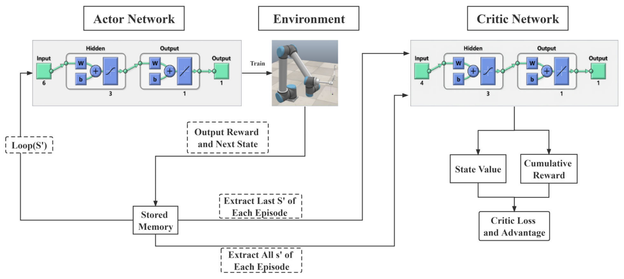
Actor Network
Environment
Critic Network
Input
Hidden
Output
Train
Loop(S')
Output Reward and Next State
Stored Memory
Extract Last S' of Each Episode
Extract All s' of Each Episode
State Value
Cumulative Reward
Critic Loss and Advantage

FIGURE 5. Algorithm Structure of Neurodynamics Adaptive Reward and Action Setting.

The proposed algorithm contains two different neural networks, and both are fully connected neural networks and also the actor-critic [33] structure. Each network plays a different role in the algorithm. One is to calculate the probability distribution of the output action, and the other network is to calculate and evaluate the value generated after the agent performs the action.

As described in Fig. 5, we use a 2-dimensional simulation as an experiment to illustrate the structure of our algorithm. The actor model contains three hidden layers, each of which has 128 neurons. The first two hidden layers contain the ReLU operation, and the last hidden layer contains the Hardwish operation. This actor network model takes the coordinates of the end-effector of the robotic manipulator and coordinates of the last action as the state and outputs the probability of action selection. For each state, the actor model infers the possible 3D coordinates of the target point and interacts with the environment. Once the environment receives the action signal, it will output the relative position of the next target point and the needle tip and form the next state together with the 3D coordinates of the end of the robotic manipulator. Therefore, in the hand-to-eye calibration task, each state is independent. Furthermore, as the traditional tip target distance (TTD) obtained by matrix transformation is replaced by the direct output of the actor-network in the deep reinforcement learning algorithm, the hand-to-eye calibration process is simplified. Moreover, the sequence of trajectories used to construct the environment is pre-collected to facilitate training of the hand-to-eye calibration model and stored separately in the form of an array, called "stored memory".

For the critic model, its structure is similar to the actor model. The difference is that it takes the last state of each episodes and all state of each episodes from the stored memory separately as the input, and output the cumulative reward and state value, where the cumulative reward and state value can be described as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{8}$$

$$V^{\pi}(s) = E_{\pi}[G_t | S_t = s], \tag{9}$$

$$Q^{\pi}(s, a) = E_{\pi}[R_t | (s, a)], \tag{10}$$

where G_t is the cumulative reward starting with time step t, $V^{\pi}(s)$ is adopted to measure the value of current state s, and Q^{π} represents the expected value that can be obtained by executing a specific action a on the current s when following the current policy π.

Then, using these information to obtain the accurate gradients to calculate the critic loss and advantage function, and guide the learning process of the actor model. In other words, the critic loss and advantage function both can be obtained by using equation (8) and (9) and advantage function can be described as:

$$A_t = G_t - V^{\pi}(s_t). \tag{11}$$

To view the actor-critic model as a whole, this model is to provide the information from the environment, including the probability distribution of action, the trajectory between end-effector of robotic manipulator and the target points, and the coordinates of the end effector corresponding to the current state of the robot arm, and the evaluation of each action selection is relatively efficient for this hand-to-eye calibration task. We will verify our assumption by running different simulation scenarios, which will be explained in later section.

C. NEURODYNAMICS ADAPTIVE REWARD AND ACTION SETTING

Generally, the reward signal is 0 in most Markov Decision Process (MDP) settings. For our hand-to-eye calibration task, this traditional reward signal setting may lead to the problem of sparse reward and refers to the problem that the agent has difficulty obtaining positive rewards during exploration, resulting in failure to learn. Hence, an intuitive way to solve the sparse reward problems is reward the agents outside the reward function when the agents take a step toward the goal. In other words, we call this method reward shaping [34], initially proposed by Andrew Y. Ng in 1999. In addition, on the basis of Zeroing discretized neurodynamics method [35], [36], to make the reward shaping more adaptable to our task, and achieve the zero stability [10], [11]. we design the reward function with neurodynamics adaptive properties, which is inspired by the Euler difference methods. Based on this, we set reward function as follows:

$$R = (TTD_{current} - TTD_{previous}) * k, \tag{12}$$

where TTD$_{current}$ is the distance between needle position and target position in current step, and TTD$_{previous}$ is the distance between target position and needle position in last step. The purpose of applying this equation is to ensure that the reward function can reflect whether agents have learned the environment based on the action selection in the last steps. In other words, this modified reward function contains information about whether the robotic manipulator is moving toward the target position so that the agent can choose an action based on the results of the current state instead of making the action selected from the initial state of the agent in every step. More important in this equation is the introduction of the amplification factor k, allowing the agent to identify the target faster. Since the neurodynamics method inspired by Euler formula is zeroing stable that is determined by the root properties of its characteristic polynomial, the neurodynamics adaptive reward function induced deep reinforcement learning is zeroing stable. More details of this second simulation will be explained in Section V.

In addition to reward shaping, we designed the following action function with same neurodynamics adaptive properties:

$$A = P_{end} + A_c * \lambda, \tag{13}$$

where P_{end} is the coordinate position of the end pose of the robotic manipulator in each steps, and A_c is the action has been outputted by the current actor network, and the function of the magnification factor λ is to reduce the action by different times according to the change of the TTD distance. To do that, agents at least has sense where it should be go to reach the target. More details of this simulation will be explained in Section IV.

On the other hand, in simple tasks, a single-step neurodynamics adaptive reward setting and action setting are sufficient for the policy to converge. For the 3D hand-to-eye calibration task, this single-step neurodynamics adaptive reward may be ineffective and time-consuming, and may not meet the efficiency requirements for industrial applications. Therefore, we taken the further step based on a single-step neurodynamics adaptive reward setting and action setting called neurodynamics adaptive step-wise settings (DASS):

$$R_{dass} = \begin{cases} R * \delta_1, & TTD > a, \\ R * \delta_2, & b < TTD \leq a, \\ R * \delta_3, & c < TTD \leq b, \\ R * \delta_4, & else, \end{cases} \tag{14}$$

and

$$A_{dass} = \begin{cases} P_{end} + A_c * \phi_1, & TTD > a, \\ P_{end} + A_c * \phi_2, & b < TTD \leq a, \\ P_{end} + A_c * \phi_3, & c < TTD \leq b, \\ P_{end} + A_c * \phi_4, & else, \end{cases} \tag{15}$$

where a, b, c are the distance conditions in the calibration task, $\delta_1, \delta_2, \delta_3, \delta_4$, are the magnification parameters for reward function, and $\phi_1, \phi_2, \phi_3, \phi_4$ are the reduction parameters of the action function. Meanwhile, we formulate a rule to regulate the parameters, that is, as TTD gradually decreases, the adaptive parameter of the reward function should increase accordingly, and the adaptive parameter of the action function should decrease accordingly.

On the basis of important sampling and clipping method, actor-critic model and neurodynamics adaptive reward and action setting method mentioned in the above, an actor-critic algorithm based on the deep reinforcement learning is proposed for hand-to-eye calibration problem, which is termed as neurodynamics adaptive deep reinforcement learning algorithm.

D. STATES INFORMATION SETTING

When solving calibration tasks using deep reinforcement learning, one critical component is the state information setting. In general, the state of a robotic manipulator includes its joint angles, velocities, and positions. For calibration tasks, additional information such as the camera image, the target position, and the distance between the manipulator end effector and the target point should also be taken into account. However, adding too many variables can lead to instability, neglecting important state information can lead to poor performance. Therefore, selecting the appropriate set of state variables is crucial for achieving a balance between performance and stability in calibration tasks using deep reinforcement learning.

In the proposed algorithms, the state information setting includes the coordinate error between the target point and the actual position of the end-effector, and the coordinates of the previous action taken by the agent. This setting offers several benefits to the training process. Firstly, it is less susceptible to noise, which can cause fluctuations in the state space and result in unstable training. Secondly, by including the coordinate error between the target point and the current position of the end effector, the agent can track its progress towards the goal and adjust its actions accordingly. This allows for a more efficient and effective learning process, as the agent can make corrections based on its current state, rather than relying solely on the initial target position. Thirdly, by including the coordinates of the actions the agent performed in the previous step, the agent can learn from past experience and avoid repeating the same mistakes. Overall, this state information setting contributes to a more stable and accurate calibration process.

V. SIMULATION AND RESULT ANALYSIS

In this section, we present the results of applying deep reinforcement learning on a UR-10 robotic manipulator via the simulation platform V-rep. Our experiment has been conducted in the V-rep platform using both Remote API clients and Embedded scripts to control the robotic manipulator. Our team has chosen UR-10 manipulator with an attached piercing tool. In this experiment, the goal is to find out the coordinate transformation relationship between the robotic manipulator's end-effector and the world coordinate systems. In other words, we are trying to solve hand-to-eye calibration problems using the deep reinforcement learning method. Based on that, we set that third-party independent camera only to observe the needle tip and target point coordinates. In the V-rep platform, we were using inverse kinematics (IK) [37] to control the UR-10 robotics arm, which means that getting every world coordinate system of joint angles is not necessary. Instead, suppose the manipulator can figure out the coordinates of the needle tip of the piercing tool through a third-party independent camera. In that case, we can deduce all coordinates of each joint angle.

A. SIMULATION PLATFORM

Simulation is a very important tool for algorithm verification and can reduce the cost of learning. For this purpose, the virtual robot experimentation platform, known as V-Rep [38] is very useful. In the V-rep platform, there are various robots, including a 7-DoF manipulator, UR5, UR10, vehicles, Dobot Magician and so on, which users can choose to perform their tasks. Meanwhile, V-Rep allows the user to use various programming tools to perform the simulation simultaneously, such as Remote API clients, Add-ons, Plug-ins, Embedded scripts, and ROS nodes. For our paper, we only use Remote API clients and Embedded scripts.

The user manual describes that Remote API clients allow V-Rep to interact with an external entity. This external entity can be any hardware, and its remote function can be written as another coding language, such as Matlab, Python, or Java. On the other hand, an embedded script is a script embedded in a model, allowing users to write the central simulation command within the V-rep platform, and the coding program used is Lua [39]. This main script is the central control of the simulation. With these two powerful programming tools, we can import deep reinforcement learning via remote API clients and start our simulation using the UR10 manipulator. Unlike the traditional hand-to-eye calibration system [40], the purpose of our simulation tasks is to enable the robotic manipulator can learn how to recognize and reach the targets by itself. To achieve that, we will use deep reinforcement learning, which will be discussed later in the following section, as our approach to constructing a hand-to-eye calibration system.

B. PARAMETER CHOICE OF SIMULATION

For simulation purpose, we carefully designed parameters (as shown in Table 1) for each of them as both of them can lead the experiment into wrong direction if parameters are not been carefully reviewed. All parameters are the same in the three different experiments except for the state and action dimensions, since the specific experiment determines their dimensions.

TABLE 1. **Parameters of neurodynamics adaptive deep reinforcement learning algorithm in simulation processing.**

Hyper Parameter	Value
State dimension	2, 4, 6
Action dimension	1, 2, 3
Net dimension	2^6
Batch size	2^6
Critic learning rate	2^{-14}
Actor learning rate	2^{-14}
τ	2^{-8}
γ	0.99
Memory size	2^{12}
Max episodes	2^{20}
Max steps	2^4
Target steps	2^{12}
Repeated times	2^3
Clip ratio	0.2

C. SIMULATION SCENARIO

We have performed three experiments based on different target selections to verify that our deep reinforcement learning algorithms with modified reward and action are suitable for hand-to-eye tasks. In the simulation process, instead of recording the success rate of each episode, we calculated the average success rate of each of the 20 episodes in one or two-dimensional target selection and every 50 episodes in three-dimensional target selection and plotted it as a line graph. According to our neurodynamic adaptive step-wise settings rules, we set our reward function and action function based on equation 14 and equation 15 in simulation processing as follows:

$$R_{dass} = \begin{cases} R*10, & TTD{>}17.3, \\ R*20, & 8.66 < TTD \leq 17.3, \\ R*40, & 4.33 < TTD \leq 8.66, \\ R*80, & else, \end{cases} \tag{16}$$

and

$$A_{dass} = \begin{cases} P_{end} + A_c * 0.1 & TTD{>}17.3, \\ P_{end} + A_c * 0.05, & 8.66 < TTD \leq 17.3, \\ P_{end} + A_c * 0.025, & 4.33 < TTD \leq 8.66, \\ P_{end} + A_c * 0.0125, & else, \end{cases} \tag{17}$$

where the value of reward and agent's action movement are based on different tip target distance (TTD), in centimeters, that is, the reward should continue to grow as the tip target distance shrinks, while the agent's movement should decrease as the distance shrinks. In this way, we can not only improve the training speed of the agent, the stability of the training, but also improve the accuracy and safety.

In addition, when setting the constraints of the step-wise reward function, we refer to the maximum range of motion of the manipulator in the last meter of the three-dimensional space. In three-dimensional space, in a cube with a side length of 1m and a diagonal length of 1.73m, the most extended trajectory of the distance between the end effector and the target point is 1.73m. Therefore, we set the initial constraint of the step-wise reward function to start from 0.173m and decrease in multiples as the manipulator gets closer to the target point.

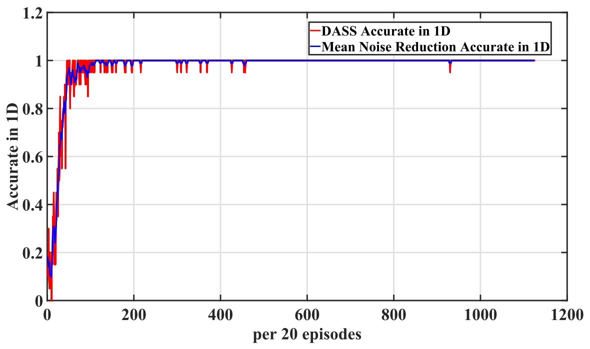

FIGURE 6. **Average accurate rate synthesized by neurodynamics adaptive deep reinforcement learning algorithm using reward function (16) and action function (15) for one dimensional hand-to-eye calibration problem.**

1) SCENARIO ONE

We start with one-dimensional target point selection, meaning only the x coordinate can be changed through entire experiments and the y, z coordinates are fixed. In other words, we set $y = 0.34412$, $z = 1.2896$, and $x \in [-0.519489, 0.580511]$. As shown in Fig. 6, this graph represents the average success rate per 20 episodes during the training simulation.The graph shows that it takes approximately 9160 episodes of training for the policy to converge. To verify this is an acceptable result, we also conducted a comparison experiment, which will be discussed in the later section. Moreover, in the graph, the red line represents the result after training without smoothing, and the blue line represents the result after smoothing using the mean noise reduction technique; the reason for adding smoothing is to reduce the volatility during training, and reducing volatility means relative smoothness. For consistency of simulation results, we performed the same procedure for the rest of the simulation.

2) SCENARIO TWO

Secondly, we did slightly more complicated experiments in which the target point space is two-dimensional, where the x, and z has been sampled uniformed, where $x \in [-0.151959, -0.751959]$, $z \in [0.98946, 1.58946]$, and $y = 0.34487$. As shown in Fig. 7, it takes approximately 10000 episodes of training for the policy to converge, which the tip target distance between(TTD) needle and the target points are within 0.1cm, and the whole training process lasted about 12 hours. Similar to the scenario one, we also did the evaluation test to verify whether the new reward and action setting is more efficiency than original setting, which will be discussed in a later.

3) SCENARIO THREE

Thirdly, we performed an even more complex simulation, which the target point is three-dimensional, where

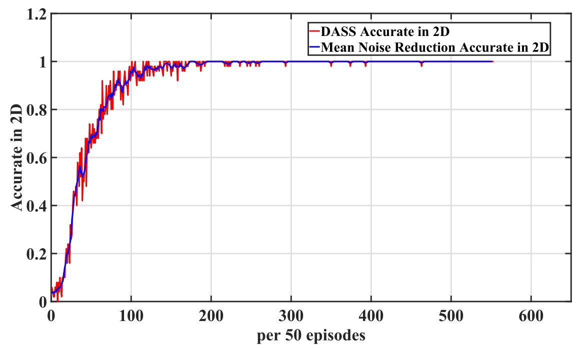

FIGURE 7. Average accurate rate synthesized by neurodynamics adaptive deep reinforcement learning algorithm using reward function (16) and action function (15) for two dimensional hand-to-eye calibration problem.

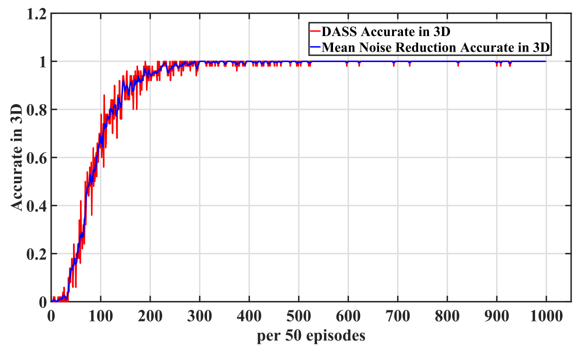

FIGURE 8. Average accurate rate synthesized by neurodynamics adaptive deep reinforcement learning algorithm using reward function (16) and action function (15) for three dimensional hand-to-eye calibration problem.

the x, y, z has been sampled uniformly, where $x \in [-0.752399, -0.151999]$, $y \in [0.59467, 0.09452]$, $z \in [1.5896, 0.98948]$. The reason we carefully designed such a target point interval is to consider the limited range of motion of the manipulator. At the same time, we consider how to make the arms of the manipulator operate without colliding with each other. The results shows in Fig. 8 told us that it takes approximately 25000 episodes of training for the algorithms to converge, and we trained continuously for 24 hours to make the robotic manipulator reach the target position accurately and maintain a stable state.

VI. COMPARISON ANALYSIS

We evaluated and compared the stability of the average accuracy and reward across different settings of action functions and reward functions in different dimensional target spaces. Additionally, we compared the results of our proposed algorithm with the DDPG and SAC algorithm to verify that our actor-critic based algorithm is more suitable for the hand-to-eye calibration task. This comparative analysis serves to provide further evidence and support for the effectiveness of our proposed algorithms. By comparing our approach to these established methods, we were able to assess its performance, evaluate its advantages, and demonstrate its superiority in addressing the calibration problem. The results

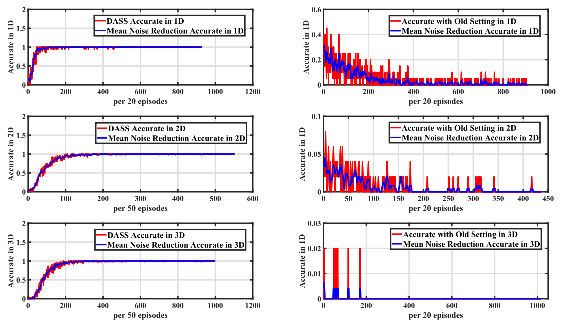

FIGURE 9. Average accurate rate synthesized by neurodynamics adaptive deep reinforcement learning algorithm using reward function (16) and action function (15) and static deep reinforcement learning algorithm using reward function (3) and action function (4) for hand-to-eye calibration problem in three different dimensional space. The top two graphs represent the average accuracy of the calibration tasks in the 1-dimensional target space, the middle two graphs represent the average accuracy of the calibration tasks in the 2-dimensional target space, and the bottom two graphs represent the average accuracy of the calibration tasks in the 3-dimensional target space.

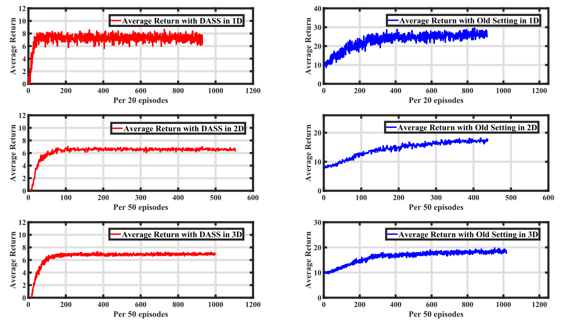

FIGURE 10. Average return compare in three different dimensional space for two different action and reward setting. The top two graphs represent the average return of the calibration tasks in the 1-dimensional target space, the middle two graphs represent the average return of the calibration tasks in the 2-dimensional target space, and the bottom two graphs represent the average return of the calibration tasks in the 3-dimensional target space.

of this comparative analysis highlight the unique contributions and advancements offered by our proposed algorithms in comparison to existing techniques.

A. AVERAGE ACCURACY AND RETURN COMPARISON FOR TWO DIFFERENT ACTION AND REWARD SETTING

As shown in Fig. 9, the average accuracy of the new action and reward function settings represented by the three left graphs is better than the original action and reward function settings represented by the three graphs on the right. Moreover, compared with the old reward and action function setting, the training process of the new setting is smooth, that is, the fluctuation between each training set is not as large as in the old setting and cannot reach the target position.

From an average episode return perspective, the average episode return should remain within a specific range. In our case, once the tip target distance (TTD) is less than or equal to the pre-designed range, we consider the agent to have done its job. Therefore, the reward should remain around 7, with a maximum of 9. As can be seen from Fig. 10, the agent achieves this well in the simulation with the new reward

TABLE 2. Statistical analysis of different algorithms in terms of their mean and variance.

	Accuracy	
	Old setting	DASS setting
1D target space	0.04	**0.98**
2D target space	0.01	**0.92**
3D target space	$1.18*10^{-4}$	**0.91**

and action function setting. However, in the old setting, the average reward kept rising, which meant that the agent could not reach the goal before the end of each episode, causing the agent to generate erroneous reward stacking.

Remark: We conducted a descriptive statistical analysis on different target dimensional spaces and find that our proposed algorithm with the DASS setting outperforms the old reward and action function setting, as shown in the Table 2. It should be noted that the data used for this analysis included a training phase from scratch, so the average accuracy was close to 100%. Nonetheless, our proposed algorithm exhibits superior performance compared to older settings, demonstrating the effectiveness of our approach.

B. AVERAGE ACCURACY COMPARISON ANALYSIS WITH DDPG ALGORITHMS

In this subsection, we conducted a simulation using the deep deterministic policy gradient (DDPG) algorithm to determine whether our proposed algorithm was better suited for the hand-to-eye calibration task. We conducted a comparative evaluation of the performance of two algorithms, under identical parameter settings, reward and action function specifications. Our analysis focused on comparing the average accuracy of the algorithms across varying target dimensional spaces. The results of our simulation are presented in Fig.11, and Table 3, which shows the average accuracy of both algorithms across different target dimensional spaces. From a training perspective, our findings suggest that although the DDPG algorithm can successfully accomplish the hand-to-eye calibration task in one- and two-dimensional spaces, its training process is highly unstable. As depicted in Fig. 11, when agents were trained to accomplish the task in one-dimensional spaces, it can be observed that after the 14,000th iteration, the agent appeared to lose the trained dataset, causing the average success rate to drop to almost zero. Similarly, agents faced the same issue when training in two-dimensional spaces after the 5,000th iteration. This sudden drop in performance could be attributed to a phenomenon known as catastrophic forgetting [41], where the agent forgets previously learned information as it learns new information. One possible reason behind this phenomenon is that the agent may mistakenly believes that the robotic manipulator is trapped in a local optimum, leading to fluctuations in training and affecting its stability and efficiency.

Moreover, when using the DDPG algorithm to train the agent in the three-dimensional target space, it encounters difficulty in successfully identifying the target, leading to more severe instances of repeated training compared to the one- and two-dimensional spaces. This can be attributed to the increased complexity of the task in three-dimensional space and the greater number of potential target positions, which presents a challenge for the DDPG algorithm.

In addition, from the perspective of the efficiency of agent training, even in one or two-dimensional target space, the agent needs at least two consecutive days of training to achieve the desired results. At the same time, even if the agent trains continuously for three days in the three-dimensional target space, it cannot achieve the desired effect. Such training efficiency has no practical advantages for the robotic manipulator in practical applications. On the contrary, our proposed algorithm can successfully accomplish the task well in less than one day in a complex three-dimensional space. Although it takes almost a day for the algorithm training to complete the autonomous calibration task, we believe that future intelligent robotic manipulators should adopt this actor-critic based deep reinforcement learning algorithm. After the robotic arm is well-trained and fully aware of the environment, it can complete any calibration task in a very short time. This approach has significant advantages over traditional calibration methods, as it does not require manual intervention and can be used for various types of robotic manipulators, reducing costs and improving efficiency.

Overall, these findings highlight the limitations of the DDPG algorithm for the hand-to-eye calibration task and suggest that our proposed algorithms is more effective for this task.

C. AVERAGE RETURN COMPARISON ANALYSIS WITH DDPG ALGORITHMS

From the perspective of the average return of agent training, our proposed algorithms have shown better performance than the DDPG algorithms. As presented in Fig.12, our proposed algorithms has higher average return than DDPG algorithms across different dimensional target spaces. In particular, the analysis of the training data in different dimensional target spaces, as shown in Table 3, demonstrates that our proposed algorithms have a higher average return than the DDPG algorithms. Furthermore, our proposed algorithms has more consistent performance across trials when compared variance of average return. Therefore, our proposed algorithms is more stable performance than the DDPG algorithms, indicating that they are more effective in training the agent for the hand-to-eye calibration task.

D. AVERAGE ACCURACY COMPARISON ANALYSIS WITH SAC ALGORITHMS

To further prove the effectiveness and superiority of our proposed algorithm, we conducted an additional simulation using the soft actor critic (SAC) algorithm [42], which has been proven to be more effective in robotic field. This simulation aims to use the identical parameter settings, reward

TABLE 3. Statistical analysis of proposed algorithms and DDPG algorithms in terms of their average return, variance, and accuracy.

	Mean		Variance		Accuracy	
	proposed algorithms	DDPG	proposed algorithms	DDPG	proposed algorithms	DDPG
1D target space	**7.21**	4.36	**0.81**	0.89	**0.98**	0.95
2D target space	**6.25**	3.28	1.37	**1.13**	**0.92**	0.87
3D target space	**6.51**	3.10	**1.77**	9.60	**0.91**	0.09

TABLE 4. Statistical analysis of proposed algorithms and SAC algorithms in terms of their average return, variance, and accuracy.

	Average Return		Variance		Accuracy	
	proposed algorithms	SAC	proposed algorithms	SAC	proposed algorithms	SAC
1D target space	7.21	**7.41**	**0.81**	2.37	**0.98**	0.89
2D target space	**6.25**	6.07	**1.37**	2.19	**0.92**	0.10
3D target space	**6.51**	3.10	**1.77**	9.60	**0.91**	0.09

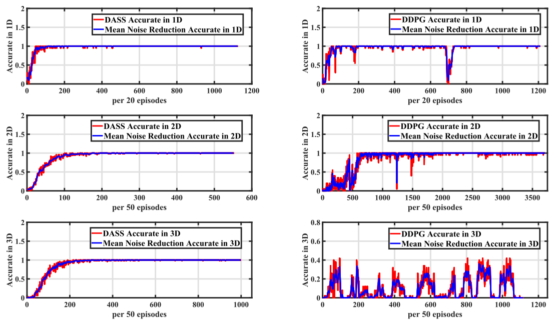

FIGURE 11. Average accuracy comparison with DDPG algotithms. The top two graphs represent the average accuracy of two algorithms in the 1-dimensional target space, the middle two graphs represent the average accuracy of two algorithms in the 2-dimensional target space, and the bottom two graphs represent the average accuracy of two algorithms in the 3-dimensional target space.

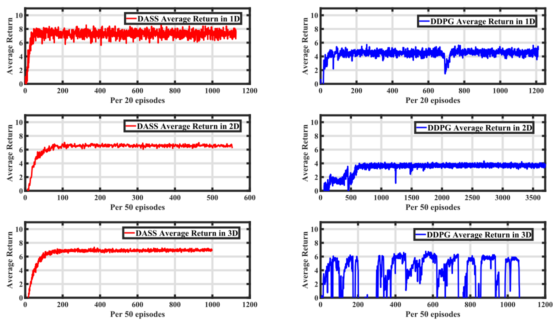

FIGURE 12. Average return comparison with DDPG algotithms. The top two graphs represent the average return of two algorithms in the 1-dimensional target space, the middle two graphs represent the average return of the two algorithms in the 2-dimensional target space, and the bottom two graphs represent the average return two algorithms in the 3-dimensional target space.

function specifications and action function specifications to compare the performance of our algorithm and SAC. Through the comparative assessment of the performance of the two algorithms, we could provide further evidences to prove the

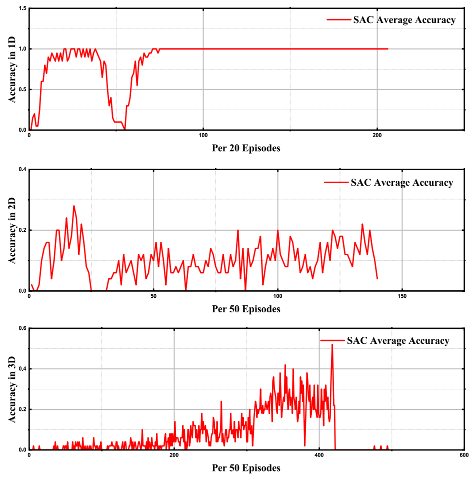

FIGURE 13. Average accuracy of SAC algorithms in three different target spaces.

feasibility and advantage of our proposed algorithm for the hand-to-eye calibration task.

As shown in Fig. 13, although SAC algorithms can quickly complete the calibration tasks in one-dimensional spaces, it has the same problems with SAC algorithms where the agent appeared to lose the trained dataset, causing the average success rate to drop to almost zero after 900th iteration. Moreover, when using the SAC algorithm to train the agent in the two- and three-dimensional target space, it encounters difficulty in successfully identifying the target, leading to more severe instances of repeated training compared to the one-dimensional spaces. This can be attributed to the increased complexity of the task in two- and three-dimensional space and the greater number of potential target positions, which presents a challenge for the SAC algorithm. In additional, as shown in Table 4, our proposed algorithms has more consistent performance across trials when compared variance of average return. Therefore, our proposed algorithms is more stable performance than the SAC algorithms, indicating that they are more effective in training the agent for the hand-to-eye calibration task.

VII. CONCLUSION

In this paper, we propose a novel method for addressing the hand-to-eye calibration problem using deep reinforcement learning. Our proposed algorithm utilizes an actor-critic framework and incorporates neurodynamics adaptive reward and action functions, which allow for better convergence, reduce the dependence on the initial value, and overcome the local convergence issues of traditional deep reinforcement learning methods. Additionally, we introduce a step-wise mechanism under the guidance of the attention mechanism, and zero stability to handle the complexity of the calibration task in challenging environments. We have conducted several simulations to demonstrate the validity of our proposed algorithm, and the results show that the agent can achieve nearly 100% accuracy after the learning phase with step-wise neurodynamics adaptive reward and action function settings. Furthermore, we have compared our proposed algorithm with DDPG and SAC algorithms through additional simulations, which further prove its effectiveness and superiority.

For future research on the intelligent robotic manipulator, the agent still has the following capabilities to improve. The training time of the hand-to-eye calibration task needs to be adapted to the increasingly developed industrial intelligent production technology. Secondly, the agent should be able to adjust the parameters in time with the sudden change of the camera position to quickly find the new relative position relationship between the robotic manipulator and the camera so that the calibration training process still can succeed in very short time.

REFERENCES

[1] R. S. Penning, J. Jung, J. A. Borgstadt, N. J. Ferrier, and M. R. Zinn, "Towards closed loop control of a continuum robotic manipulator for medical applications," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4822–4827.

[2] R. Shah and A. B. Pandey, "Concept for automated sorting robotic arm," *Proc. Manuf.*, vol. 20, pp. 400–405, Jan. 2018.

[3] E. Schoenfeld, L. Parrington, and S. Von Muehlen, "Door breaching robotic manipulator," *Proc. SPIE*, vol. 6962, Apr. 2008, Art. no. 69620S.

[4] H. Cheng, H. Chen, and B. W. Mooring, "Accuracy analysis of dynamic-wafer-handling robotic system in semiconductor manufacturing," *IEEE Trans. Ind. Electron.*, vol. 61, no. 3, pp. 1402–1410, Mar. 2014.

[5] Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," *Sci. Robot.*, vol. 2, no. 7, Jun. 2017, Art. no. eaan5074.

[6] K. Paes, W. Dewulf, K. V. Elst, K. Kellens, and P. Slaets, "Energy efficient trajectories for an industrial ABB robot," *Proc. CIRP*, vol. 15, pp. 105–110, Jan. 2014.

[7] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "KCT: A MATLAB toolbox for motion control of KUKA robot manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2010, pp. 4603–4608.

[8] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 16–29, Feb. 1989.

[9] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, pp. 3–24, Oct. 2007.

[10] L. Chen, L. Jin, and M. Shang, "Zero stability well predicts performance of convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 6, pp. 6268–6277.

[11] Z. Zheng and D. Zheng, "From zeroing dynamics to zeroing-gradient dynamics for solving tracking control problem of robot manipulator dynamic system with linear output or nonlinear output," *Mathematics*, vol. 11, no. 7, pp. 1–24, Mar. 2023.

[12] R. Bellman, "A Markovian decision process," *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, 1957.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12. Cambridge, MA, USA: MIT Press, Dec. 2000, pp. 1057–1063.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[17] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[19] M. Andrychowicz, F. Wolski, and A. Ray, "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[20] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[22] A. Tabb and K. M. A. Yousef, "Solving the robot-world hand-eye(s) calibration problem with iterative methods," *Mach. Vis. Appl.*, vol. 28, nos. 5–6, pp. 569–590, Aug. 2017.

[23] J. Angeles, G. Soucy, and F. P. Ferrie, "The online solution of the hand-eye problem," *IEEE Trans. Robot. Autom.*, vol. 16, no. 6, pp. 720–731, Dec. 2000.

[24] F. C. Park and B. J. Martin, "Robot sensor calibration: Solving AX=XB on the Euclidean group," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 717–721, Oct. 1994.

[25] K. Daniilidis, "Hand-eye calibration using dual quaternions," *Int. J. Robot. Res.*, vol. 18, no. 3, pp. 286–298, Mar. 1999.

[26] S. Qiu, M. Wang, and M. R. Kermani, "A new formulation for hand–eye calibrations as point-set matching," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 9, pp. 6490–6498, Sep. 2020.

[27] X. Chen, C. Wang, W. Zhang, K. Lan, and Q. Huang, "An integrated two-pose calibration method for estimating head-eye parameters of a robotic bionic eye," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1664–1672, Apr. 2020.

[28] L. Zheng, H. Wu, L. Yang, Y. Lao, Q. Lin, and R. Yang, "A novel respiratory follow-up robotic system for thoracic-abdominal puncture," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2368–2378, Mar. 2021.

[29] Z. Zhao, "Hand-eye calibration using convex optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2947–2952.

[30] R. Horaud and F. Dornaika, "Hand-eye calibration," *Int. J. Robot. Res.*, vol. 14, no. 3, pp. 195–210, Jun. 1995.

[31] M. Zhou, M. Hamad, J. Weiss, A. Eslami, K. Huang, M. Maier, C. P. Lohmann, N. Navab, A. Knoll, and M. A. Nasseri, "Towards robotic eye surgery: Marker-free, online hand-eye calibration using optical coherence tomography images," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3944–3951, Oct. 2018.

[32] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[33] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1–7.

[34] AY. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 278–287.

[35] D. Chen and Y. Zhang, "Robust zeroing neural-dynamics and its time-varying disturbances suppression model applied to mobile robot manipulators," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4385–4397, Sep. 2018.
[36] B. Qiu and Y. Zhang, "Two new discrete-time neurodynamic algorithms applied to online future matrix inversion with nonsingular or sometimes-singular coefficient," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2032–2045, Jun. 2019.
[37] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Expanding Societal Role Robot. Next Millennium*, Oct./Nov. 2001, pp. 298–303.
[38] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1321–1326.
[39] R. Ierusalimschy, L. H. de Figueiredo, and W. C. Filho, "Lua—An extensible extension language," *Softw., Pract. Exp.*, vol. 26, no. 6, pp. 635–652, Jun. 1996.
[40] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 345–358, Jun. 1989.
[41] K. James, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[42] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*.

ZHENG ZHENG received the B.S. degree in actuarial science from the University of Nebraska Lincoln (UNL), in 2017, and the M.S. degree in global logistic from Arizona State University (ASU), in 2018. He is currently pursuing the Ph.D. degree in mathematics with the South China University of Technology (SCUT). His research interests include the application of reinforcement learning methods in robotics and zeroing stability in nonlinear dynamic systems.

MENGFEI YU received the B.S. degree in information and computing science from Jiangxi Normal University, Nanchang, China, in 2020. He is currently pursuing the master's degree in computational mathematics with the South China University of Technology, Guangzhou. His research interests include robotics, reinforcement learning, and data mining.

PENGFEI GUO (Member, IEEE) received the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2015. He is currently an Associate Professor with the School of Computing Science, Zhongkai University of Agriculture and Engineering, Guangzhou. He has been a Visiting Scholar with Cardiff University. From 2008 to 2009, he was a Research Assistant with the Institute of Physics, Academia Sinica, Tapei, Taiwan. His research interests include the neural networks, computer vision, robot arm control, image quality assessment, the development of surface processing and biological/medical treatment techniques using non-thermal atmospheric pressure plasmas, the fundamental study of plasma sources, and the fabrication of micro- or nanostructured surfaces. His awards and honors include the Frew Fellowship (Australian Academy of Science), the I. I. Rabi Prize (APS), the European Frequency and Time Forum Award, the Carl Zeiss Research Award, the William F. Meggers Award, and the Adolph Lomb Medal (OSA).

DELU ZENG received the bachelor's degree in applied mathematics and the Ph.D. degree in signal and information processing from the South China University of Technology (SCUT), Guangzhou, in June 2003 and June 2010, respectively. He has been a Visiting Scholar with Columbia University, the University of Waterloo, and the University of Oulu. He is currently a Full Professor with the School of Electronic and Information Engineering, SCUT. His research interests include statistics learning, image and speech processing, computational intelligence, machine learning, fitting and approximation and their applications to communications, and industrial intelligence.