Optimization of Robot Environment Interaction Based on Asynchronous Advantage Actor-Critic Algorithm

Jitang Xu[1]*, Qiang Chen[2]
School of Artificial Intelligence, Tianjin Bohai Vocational Technology College, Tianjin, 300402, China[1]
Beijing Zyhit Data Technology Co., Ltd., Beijing, 100080, China[2]

***Abstract*—With the continuous advancement of automation and intelligent technology, the application of robots in environmental interaction and delivery tasks has become increasingly important. A new noise network and advantage function were constructed. An asynchronous update mechanism was adopted to enhance the exploration ability and learning efficiency of A3C. Simulation tests were conducted on classic control tasks on the Gym platform and in complex Atari game environments. Experimental verification was conducted in actual robot grasping tasks to evaluate the proposed method's performance. The improved A3C reduced training steps by 14.4% in the "CartPole-v0" task, improved scores by 31.9% in the "BreakoutNoFrameskip-v4" game, and increased scores by 7.74% in the "PongNoFrameskip-v4" game. In actual testing, the position error was controlled at the pixel level, proving the algorithmic accuracy in delivery tasks. This study provides new technological support for the advancement of robotics technology.**

Keywords*—*A3C; robot environment interaction; intelligent technology; simulation testing

I. Introduction

With the continuous progress of automation and intelligence technology, the application of robots in environmental interaction and delivery tasks has become increasingly important [1]. However, existing robot systems face multiple challenges when performing complex tasks, including processing capabilities for high-dimensional environmental inputs, low sample efficiency, and insufficient adaptability in dynamically changing environments. These issues limit the performance and reliability of robots in practical applications [2-3]. For human-machine interaction and delivery tasks, the current methods often use a single sensor, can only ensure effectiveness in specific scenarios, or require high hardware requirements. The cost of system construction is expensive, the process is cumbersome, and requires a lot of prior knowledge [4-5]. Therefore, this study proposes a robot environment interaction optimization using the Asynchronous Advantage Actor-Critic (A3C) algorithm to enhance the exploration ability and adaptability of the method. The research motivation includes improving task execution efficiency, enhancing environmental adaptability, reducing system costs, ensuring human-machine interaction safety, and achieving precise control, in order to provide theoretical and technical support for the practical application of robots in industrial, service, and life scenarios. The innovation of this research lies in introducing new noise network design and advantage functions, as well as asynchronous update mechanisms. These improvements can better handle Gaussian noise and are expected to achieve more accurate robot decision-making and control in complex environments.

The study conducted in-depth discussions on the performance optimization of robots in environmental interaction and delivery tasks, and the entire structure is divided into five sections. Firstly, Section II provides an overview of existing robot interaction technologies, clarifying their challenges in adapting to complex environments and improving efficiency. Section III provides a detailed introduction to the system optimization design based on the A3C algorithm, including the construction of new noise networks, advantage functions, and asynchronous update mechanisms, which provide solutions to existing challenges. Section IV analyzes the performance of the proposed optimization method through simulation testing and experimental verification of actual robot tasks. The results show that the improved A3C algorithm has significant advantages in reducing training steps and improving control accuracy. Finally, Section V summarizes the entire article, emphasizing the contribution of research results to promoting the development of robotics technology, and proposing future research directions. Overall, this paper directly addresses the efficiency and accuracy issues in robot interaction. Through system optimization design and strict experimental verification, it provides practical and feasible solutions, laying a foundation for further research in the fields of automation and intelligence.

II. Related Works

A3C can enhance the learning and decision-making abilities of intelligent agents in complex environments. Tuli et al. proposed a real-time scheduler based on A3C and R2N2, which supported multi-agent decentralized learning and adaptively adjusted hyperparameters. Compared to existing algorithms, this method achieved significant improvements in energy consumption, response time, service protocol, and cost [6]. Labao et al. proposed an A3C-GS algorithm that enhanced exploration ability and reduced dependence on entropy loss through asynchronous gradient sharing. This algorithm achieved policy diversity in the short-term and ensured convergence to the optimal strategy in the long term [7]. To solve the maneuver decision-making problem of Unmanned Combat Aircraft Vehicle (UCAV) in air combat, Fan et al. proposed an autonomous maneuver decision-making means.

By establishing a UCAV flight model and maneuver library, a dual layer reward mechanism was designed. A model was constructed using A3C. This method was validated for its effectiveness and feasibility in three air combat scenarios [8]. For the insufficient processing capacity of the block chain miner in the mobile system, Du et al. used mobile edge computing to unload tasks. A3C combined with prospect theory was adopted to achieve resource pricing and allocation [9]. Ye et al. aimed to address the intelligent event triggered localization control of networked unmanned marine vehicle systems in the face of mixed attacks. The reduction of communication load was achieved by establishing a T-S fuzzy system model with random switching, introducing A3C learning event triggering method, and designing a controller using Lyapunov stability theory. Finally, the proposed control strategy's effectiveness was verified through a networked UMV system instance [10].

With the advancement of artificial intelligence, the environmental interaction ability of robots is significantly improved. The growth of the e-commerce business leads to an increase in delivery costs and increased customer demand. Regarding this, Benarbia et al. explored existing solutions and concepts for unmanned aerial vehicle delivery systems in logistics design and modeling. Drone delivery systems showed potential in reducing costs and shortening delivery times [11]. In urban environments, Yu et al. put forward a two-stage truck robot system to solve the last mile delivery problem. Large-scale instances were processed by establishing a mixed model, considering time, goods, and energy, and developing an adaptive large neighborhood search algorithm [12]. Ostermeier et al. proposed a routing optimization algorithm that combined neighborhood search and cost priority rules to optimize the cost of truck and robot systems in last mile delivery. Meanwhile, two robot scheduling strategies were evaluated. This method was validated through numerical experiments to significantly reduce delivery costs [13]. Bakach et al. investigated the optimal deployment quantity and operating cost of robots in a dual layer distribution network. Meanwhile, a mixed integer programming model considering robot battery limitations was constructed based on the p-midpoint problem. Compared to truck delivery, robot delivery saved up to 70% to 90% in cost [14]. Byrd et al. focused on the service quality of food delivery robots in food delivery services. The gap between consumer expectations and robots' actual performance was analyzed through investigation and on-site observation. Consumers had lower anticipations, but there was no significant difference in actual performance [15].

In summary, many experts have conducted research on the interaction between A3C and robots. However, there are still shortcomings in sample efficiency and the ability to handle high-dimensional inputs. Therefore, this study proposes an A3C-based optimization of robot environment interaction to address the efficiency and accuracy of robot delivery interaction in complex environments. It is hoped to help improve the autonomy, flexibility, and interaction efficiency of robots in practical applications.

III. System Optimization Design for Robot Delivery Interaction based on A3C

The first section analyzes the importance of integrating advanced environmental perception technology. Meanwhile, it analyzes how to determine robot's basic coordinate framework and the end effector's spatial relationship through a homogeneous transformation matrix of forward kinematics. The second section elaborates on the improvement of the algorithm, including the design of new noise networks and the application of advantage functions.

A. Design of Human-Machine Transmission System and Environmental Perception Technology in Robot Delivery Interaction

In the robot environment interaction, the design of human-machine transmission system is crucial, which influences the delivery interaction between robots and humans. The human-machine transmission system needs to integrate advanced environmental perception technology to improve the cognitive accuracy of robots in complex environments [16]. The spatial relationship between the basic coordinate framework of a robot and the local coordinate system of its end effector (such as a gripper) is usually determined through the homogeneous transformation matrix of forward kinematics. Fig. 1 shows the connection among the claw coordinate system {Os}, the object coordinate system {Oo}, and the camera coordinate system {Oc}.

In Fig. 1, the claw coordinate system's origin position is set at 276 millimeters in the seventh coordinate system's positive Z-axis direction. The X, Y, and Z axes are parallel and aligned with the corresponding axis of the seventh coordinate system. Oo is based on the object's geometric center as its origin. The Z-axis is the normal line on the object's front surface and is away from the robot. The Y-axis conforms to the object's long side and extends upwards. The X-axis follows the right-hand rule. To locate an object in the world coordinate system, it is first necessary to extract the object's coordinate information from Oc. These four corner coordinates of the rotation box provided by the network output can be used to calculate the center's pixel coordinates of a rotation box. Fig. 2 shows the pixel coordinates of the center point as Cx, Cy, and rotation angle θ.

Due to the known pixel coordinates of point A, calling the Kinect V2 camera API can obtain the depth value Z_C of point A. Based on the calibrated inner and outer reference matrices of the camera, the representation ${}^{W}\mathrm{p}_A=(x_w, y_w, z_w)$ of point A in the world coordinate system can be obtained. For the pose, the object in the camera coordinate system is represented by Eq. (1).

$$
\begin{aligned}
{}^{C}_{O}R &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} R(Z_C, -\theta) \\
&= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c\theta & s\theta & 0 \\ -s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ s\theta & -c\theta & 0 \\ c\theta & s\theta & 0 \end{bmatrix}
\end{aligned} \tag{1}
$$

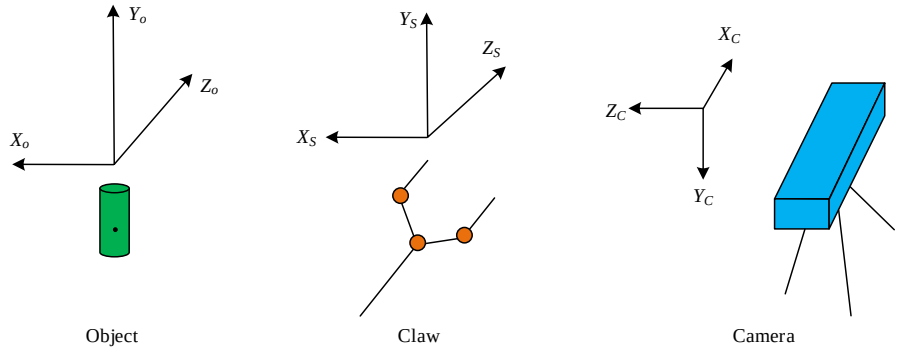


Fig. 1. Coordinate system relative relationship diagram.

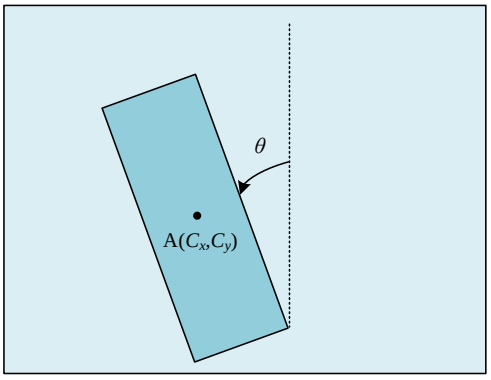


Fig. 2. Angle definition diagram.

Eq. (1) represents rotating $-\theta$ around the Z_C axis. The extrinsic matrix for camera calibration is represented by Eq. (2).

$$ {}^{W}_{C}T = \begin{bmatrix} {}^{W}_{C}R & {}^{W}\mathrm{p}_C \\ 0 & 1 \end{bmatrix} \tag{2} $$

The pose of an object in the world coordinate system is represented by Eq. (3).

$$ {}^{W}_{O}R = {}^{W}_{C}R \cdot {}^{C}_{O}R \tag{3} $$

When the claw grasps an object, their postures are the same, i.e. ${}^{W}_{G}R = {}^{W}_{O}R$. Due to the fact that the attitude input method of the machine control API is RPY, it is necessary to perform RPY inverse solution to obtain the RPY of the attitude represented as (U,V,W). For position, when the gripper grasps an object, the object's geometric center is 120mm in the gripper coordinate system's Z direction. Therefore, the robot tracking point D's position coordinates in the gripper coordinate system are represented by Eq. (4).

$$ {}^{G}\mathrm{p}_D = (0,0,120) \tag{4} $$

It can be obtained that D is in the world coordinate system, represented by Eq. (5).

$$ {}^{W}\mathrm{p}_D = {}^{W}_{G}R \cdot {}^{G}\mathrm{p}_D \tag{5} $$

After determining the position of an object through the camera system, the robot executes actions to grab and maintain a constant speed to the target point. A dynamic step size adjusting strategy on the ground of position error is put forward for robot EPSON with only position control, and the Y-axis control is optimized. When the system starts, it uses neural networks and cameras to calibrate and locate objects, and uses robot API to obtain the position of the gripper. Then, the error in the position difference between the object on the Y-axis and the gripper is calculated using Eq. (6).

$$ \begin{cases} e_x = x_O - x_G \\ e_y = y_O - y_G \\ e_z = z_O - z_G \end{cases} \tag{6} $$

In Eq. (6), x_O, y_O, and z_O represent the object's X, Y, and Z coordinates. x_G, y_G, and z_G represent the claw's X, Y, and Z coordinates. After calculating the position deviation, robots will execute corresponding control strategies based on this deviation. If there is a deviation in the Y-axis direction (e_x or e_z is non-zero), it indicates that the gripper and object have not been aligned. The robot will first perform precise servo motion to ensure that both are aligned along the Y-axis. Subsequently, based on e_y, the robot will enter different stepping control intervals. This study divides the control interval into five regions, labeled as Area1 to Area5. Each region corresponds to a different e_y threshold, guiding the robot to gradually approach the target object along the Y-axis. If the object's position changes, robots will pause further approach along the Y-axis. Table I shows the specific correspondence between e_y and step size S, as well as the regional division of e_x and e_z.

TABLE I. THE RELATIONSHIP BETWEEN e_y AND STEP SIZE S

Area	$\lvert e_y \rvert$ (mm)
1	<30
2	30-120
3	120-300
4	300-540
5	>540

Meanwhile, the same strategy is adopted for tracking in the X and Z planes. The claw position control strategy is represented by Eq. (7).

$$x_G^{k+1} = x_G^k + flag \times S \tag{7}$$

In Eq. (7), x_G^{k+1} represents the coordinates of the claw at the next moment. x_G^k represents the coordinates of the claw currently. S is the movement's step size. $flag$ is related to e_x. If $e_x > 0$, $flag$ is 1. If not, the value is -1. S is determined by the different areas where claws are located. The regional position is determined by the relative position e_x's $|e_x|$. The algorithm provides differentiated step sizes for the claw in various regions to optimize its motion performance. However, simply changing the step size may lead to rapid speed changes during the servo process, thereby affecting the stability and servo efficiency of the robot. Therefore, the study sets an expected range $(StepMin, StepMax)$ for every region. When $|e_x|$ is located in a certain region, if $S > StepMax$, S should decrease to get into a desired location, i.e. S, otherwise it should increase. S in three different situations is represented by Eq. (8).

$$\begin{cases} S^{next} = S^{pre} + acc, \left(S^{pre} < StepMin\right) \\ S^{next} = S^{pre} - dec, \left(S^{pre} > StepMax\right) \\ S^{next} = S^{pre}, \left(StepMin < S^{pre} < StepMax\right) \end{cases} \tag{8}$$

In Eq. (8), S^{next} and S^{pre} refer to the next and current moment's step sizes. acc and dec refer to the step increment when accelerating and decelerating. Within a specific error range, if the current step size does not reach the preset ideal range, a phased adjustment strategy will be used to gradually adjust the step size to the target range. This experiment determines whether to increase or decrease based on the current step size and $(StepMin, StepMax)$.

B. Mechanism Analysis and Optimization of A3C in Robot Delivery Interaction

A3C is a reinforcement learning strategy characterized by multi-agents interacting with the environment in parallel, improving training efficiency. This algorithm utilizes multi-step reward evaluation behavior and guides learning [17]. A3C reduces sample correlation and accelerates learning through asynchronous updates. This algorithm helps robots efficiently complete tasks in the operating environment [18]. Therefore, this study optimized the robot using A3C. Fig. 3 shows the workflow of A3C.

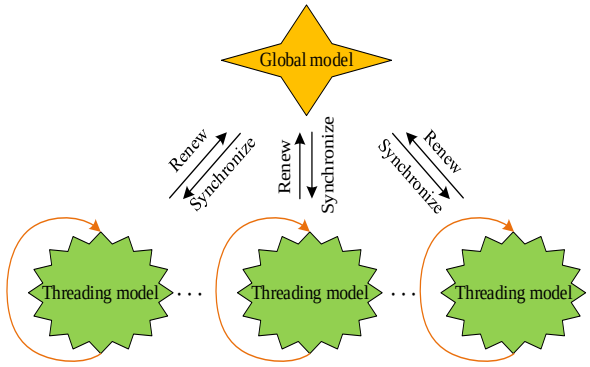


Fig. 3. A3C algorithm workflow.

In Fig. 3, the core mechanism of A3C is to run multiple sub processes in parallel. A3C adopts two network architectures, one is the global model in the main process, and the other is the local model within each sub thread [19]. A3C determines the threads in a parallel environment based on the CPU performance of the computer. Meanwhile, A3C distributes these local models to various sub threads to achieve synchronous training, allowing each agent to learn from its local environment. After the local model completes parameter updates, these updates will be merged into a global model. Meanwhile, this model's comprehensive updates will also be fed back to the local model to guide the subsequent learning process [20]. Fig. 4 is a training network model.

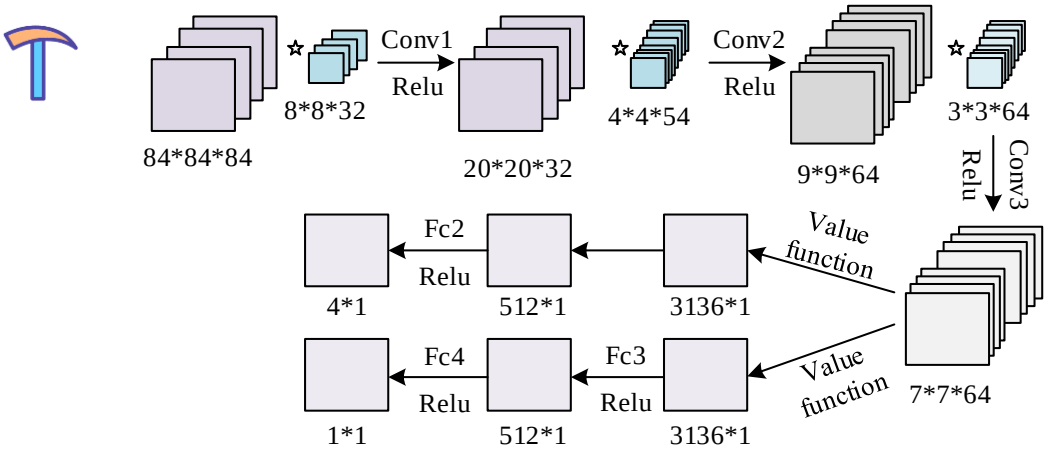


Fig. 4. Model diagram for training network.

The most crucial part of this model is to propose a structure with independent and decomposed Gaussian noise. Firstly, improvements are made to a noise parameter σ's weight and bias in the noise network's Fully Connected Layer (FCL). Independent Gaussian noise refers to the addition of random noise parameters ε and σ obeying a normal distributing for each weight w in a FCL. Decomposing Gaussian noise is to minimize the random samples, retaining two random Gaussian vectors ε_i and ε_j. Then each weight in FCL is added to σ, and the layer's random matrix derives from vectors ε_i and ε_j's outer product. Finally, each weight w and bias b in FCL are added to a weight σ_w of σ obeying a Kaiming normal distributing and a bias σ_b of σ obeying a normal distributing, respectively.

ε and σ are stored in a FCL and trained using backpropagation. This noise layer's output has the same calculation means as the linear layer's weights of training standard. Meanwhile, a weight ε_w of ε obeying a Kaiming normal distributing and a bias ε_b of ε obeying a normal distributing are used to update the noise tensor parameter. This can avoid keeping noise constant when being trained, enhancing training data's richness and diversity. The Kaiming normal distribution is a normal distribution that follows $N(0,\theta)$. The variance θ is represented by Eq. (9).

$$\theta = \sqrt{\frac{2}{(1+c^2)\times f}} \tag{9}$$

In Eq. (9), c represents that in the subsequent activation layer, the negative slope is used to maintain the stability of the weight variance in forward propagation. f represents the selected position of the model. Running a sampled sample can obtain an unbiased estimate of $q_\pi(s,a)$. $v_\pi(s,a)$ can be calculated through neural networks. Based on the sampled samples, $q_\pi(s,a)$ and $v_\pi(s,a)$'s error is computed, and two value functions' difference is calculated. Therefore, this function is defined as the mean squared error, represented by equation (10).

$$V_{loss} = \frac{1}{n}\sum_{i=1}^{n}\left(q_\pi(s,a) - v_\pi(s)\right)^2 \tag{10}$$

This study is defined by averaging the cumulative rewards of all initial states. Then, based on the strategy gradient theory, the gradient of cumulative rewards is calculated and optimized for maximum value. Finally, by averaging the expected values of a small batch of samples, the estimation of the expected values is completed. The specific loss function is represented by equation (11).

$$P_{loss} = -\frac{1}{n}\sum_{i=1}^{n}\left(A_t \log \pi(a \mid s);\theta\right) \tag{11}$$

In equation (11), A_t represents an advantage function, which means an advantage of action a relative to the average in state s. From the perspective of numerical relationships, it can be understood as the degree of deviation of a random variable from its mean. By standardizing the state behavior value function to a value function's basic level, this method helps improve learning efficiency, reduce variance, and prevent overfitting caused by excessive variance. Fig. 5 shows the average gradient parallelization network.

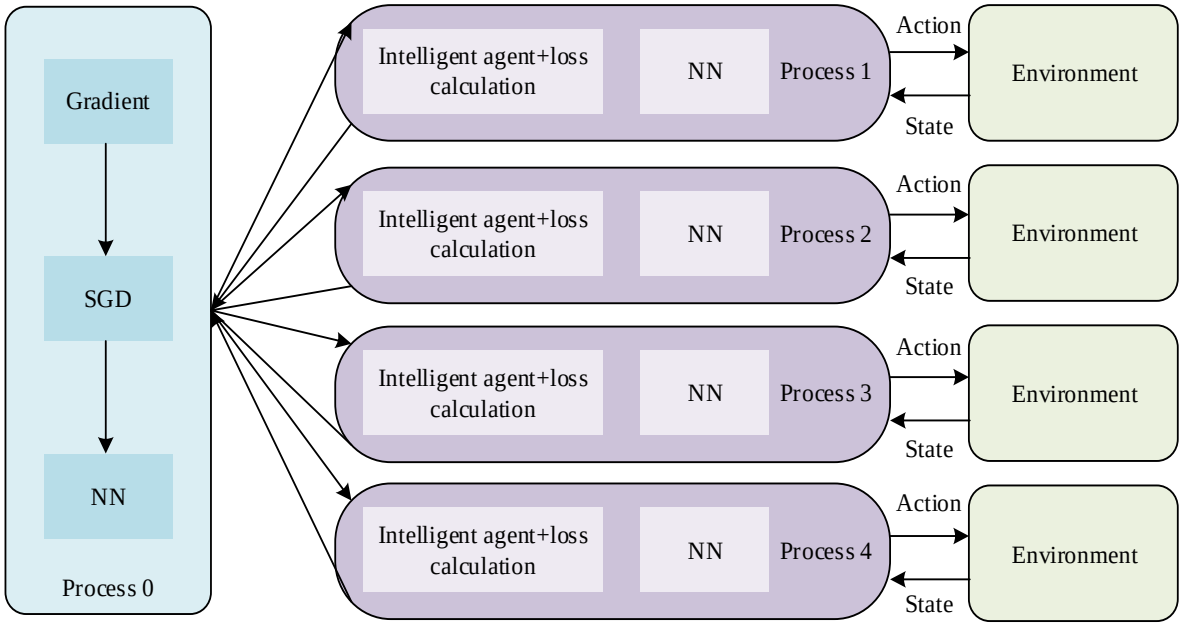


Fig. 5. Structure of average gradient parallelization network.

In the multi-process parallel training of A3C, the average gradient parallelization strategy is introduced. This can solve the high cost caused by data parallelization and the problem of gradient differences affecting convergence in gradient parallelization. This strategy first allows sub-processes to independently calculate gradients and send these gradients to the main process. In the main process, all received gradients are summed and averaged to reduce gradient differences between different sub-processes. Using this average gradient and advantage function, the main process updates the loss function and performs random gradient descent to adjust network weights. The renewed weights are broadcasted back to sub-processes, ensuring real-time policy updates. The calculation of the advantage function does not use traditional methods, but instead introduces generalized advantage estimation to estimate the advantage function. The advantage lies in the ability to effectively make bias and variance's influence on value estimation and returns balanced. The specific advantage function is represented by Eq. (12).

$$\begin{aligned}\hat{A}_t &= \hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l+1} = \delta_t + \gamma\lambda\hat{A}_{t+1}^{GAE(\gamma,\lambda)} \\ &= \delta_t + (\gamma\lambda)^l \delta_{t+l+1} = \delta_t + \gamma\lambda\hat{A}_{t+1}^{GAE(\gamma,\lambda)}\end{aligned} \quad (12)$$

In Eq. (12), λ is a hyperparameter used to balance variance and bias. When $\lambda = 0$, it is to calculate the difference between the actual value and the estimated value. $\lambda = 1$ is a Monte Carlo target value and a value estimate's difference when calculating. δ_t refers to the actual and estimated values' difference, represented by Eq. (13).

$$\delta_t = r_t + \gamma v(S_{t+1}) - v(S_t) \quad (13)$$

In the interactive learning between intelligent agents and the environment, agents continuously explore in various states to gain different actions' feedback. This enables intelligent agents to learn by practicing, while utilization is based on feedback already obtained to select the optimal action. To maintain a balance between exploration and utilization, intelligent agents should attempt new actions with a certain probability to ensure diversity in sample collection. Therefore, this study seeks to maximize cross entropy (or equivalently minimize negative entropy) to achieve equilibrium in the output distribution, represented by Eq. (14).

$$E_{loss} = -\frac{1}{n}\sum_{i=1}^{n} H(\pi(s_i)) = \frac{1}{n}\sum_{i=1}^{n}\sum_{i=1}^{n}\pi(s_i)\log\pi(s_i) \quad (14)$$

The final total loss function can be obtained in Eq. (15).

$$L = Vloss + Ploss + Eloss \quad (15)$$

The overall parameter optimization steps are as follows: Firstly, introduce independent Gaussian noise and decomposed Gaussian noise into the fully connected layer to enhance the exploratory nature of the model; Secondly, the advantage function has been redesigned to reduce variance in the learning process by standardizing state behavior values; Furthermore, for robot delivery interaction, the step size is dynamically adjusted based on position error to optimize Y-axis control; Meanwhile, set the expected step size range and adjust it based on the difference between the current step size and the expected step size; Finally, in multi process parallel training, the average gradient parallelization strategy is adopted to reduce differences and improve the convergence and stability of the algorithm by averaging the gradients of each sub process.

IV. Performance Testing and Analysis of Robot Delivery Interaction based on A3C

The proposed A3C was validated in classic control tasks and complex Atari game environments on the Gym platform. Gym, as an open-source reinforcement learning toolset, provides rich simulation scenarios such as classic control tasks, Atari games, algorithm challenges, and mujoco simulations. Fig. 6 shows a comparative testing environment.

In Fig. 6 (a), in the "CartPole-v0" task, the goal is to balance a pole on a moving car to avoid pole collapse or car deviation from the track. In Fig. 6 (b) and (c), in Atari's "PongNoFrameskip-v4" and "BreakoutNoFrameskip-v4" games, table tennis and brick playing games were simulated, respectively. The former was determined by the score, while the latter was scored based on the cumulative number of blocks broken. In these three different testing scenarios, the performance of the improved A3C was compared with that of the standard A3C. Fig. 7 is a simulation diagram of Experiment 1.

(a) "CartPole-v0"environment map (b) "PongNoFrameskip" environment map (c) "BreakoutNoFrameskip" environment map

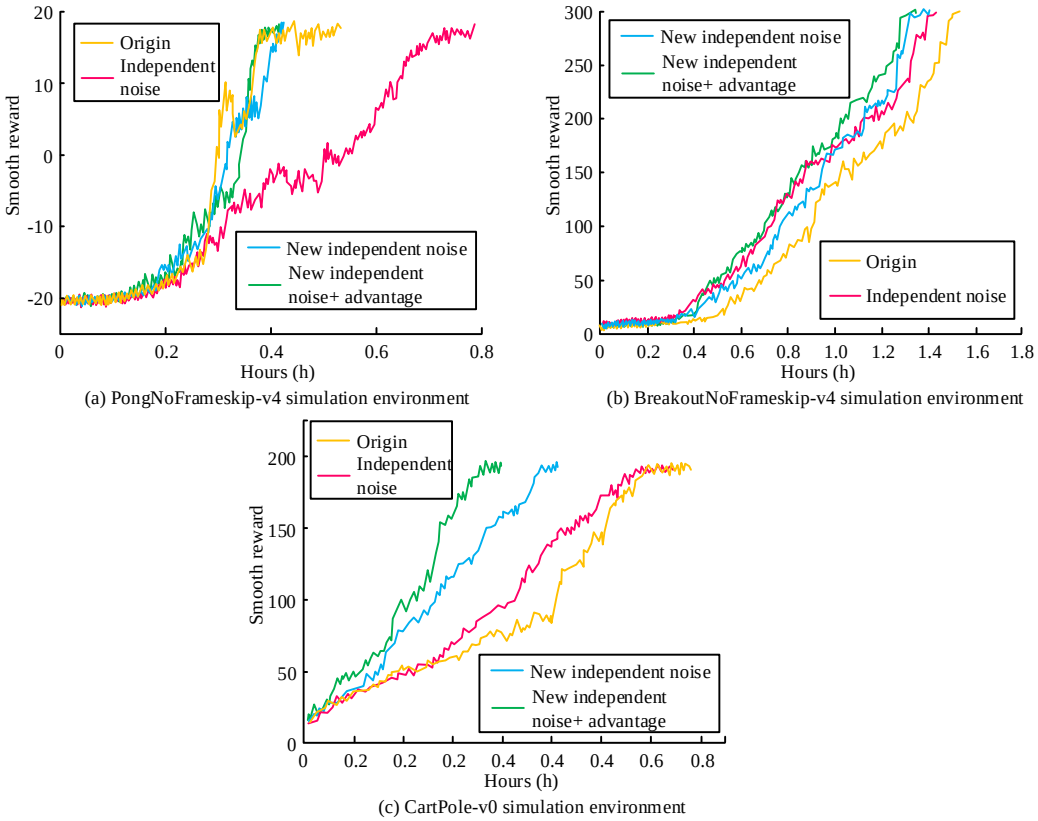
Fig. 6. Comparative test environment diagram.

Fig. 7 (a), (b), and (c) show the relationship between different algorithm training times and the smooth reward results obtained in three experimental simulating environments: "PongNoFrameskip~V4", "BreakoutNoFrameskip-v4", and "CartPole-v0", as well as the algorithmic training procedures to reach a rewarding boundary, respectively. In Experiment 1, the performance of three A3Cs was compared, including the original A3C, Deepmind's in-depended Gaussian noise processing A3C, and an improved A3C combining novel in-depended Gaussian noise and a new dominance function. The improved A3C had a faster convergence speed, shorter training time, and more reasonable step size adjustment during the training process. This new algorithm adopted the assumption of independence when dealing with Gaussian noise, which helped to simulate and adapt to environmental noise more accurately. The design of this new advantage function further improved the algorithm's learning efficiency. Fig. 8 is a simulation of Experiment 2.

In Fig. 8 (a) (b) (c), an A3C after decomposed Gaussian noise processing, an A3C after newly decomposed Gaussian noise processing, an A3C after new dominance function processing, and the original A3C were simulated and tested in three simulating environments. Compared with Deepmind's proposed decomposed Gaussian noise A3C and the original A3C, the A3C processed with new independent Gaussian noise showed significant advantages in convergence, training time, and step size control. Similarly, when applied to "BreakoutNoFrameskip-v4" and "CartPole v0", this method performed better than the original A3C. In the PongNoFrameskip-v4, although the training time increased, this method improved convergence and step size control. Therefore, A3C, which underwent new noise processing, effectively avoided the constancy of noise during the training by explicitly updating the noise tensor.

The experiment was equipped with a six degree of freedom EPSON6 robot, a six-dimensional force sensor, gripper, and KinectV2 camera, as well as two computers for image processing and robot control. These robots were placed on the desktop with the base as the world coordinate origin. KinectV2 was located at a height of approximately 130 centimeters on the right side of the robot. The force sensor and gripper were installed at the end of the robot. This study selected black cups for network recognition testing. In Table II, the recognition results indicated that all images were accurately classified.

Table II compares the manually annotated center point position and rotation angle of the black cup with the results recognized by the robot grasping system. These data were automatically extracted from the label file, with center point coordinates in pixels and rotation angles in degrees. To ensure the accuracy of error calculation, the coordinate values were rounded to two decimal places. Although angle values were based on classification, they were also expressed as decimals. The optimized A3C accurately predicted the position and angle of objects in robot grasping tasks, exhibiting lower noise levels. Fig. 9 shows the servo curve.

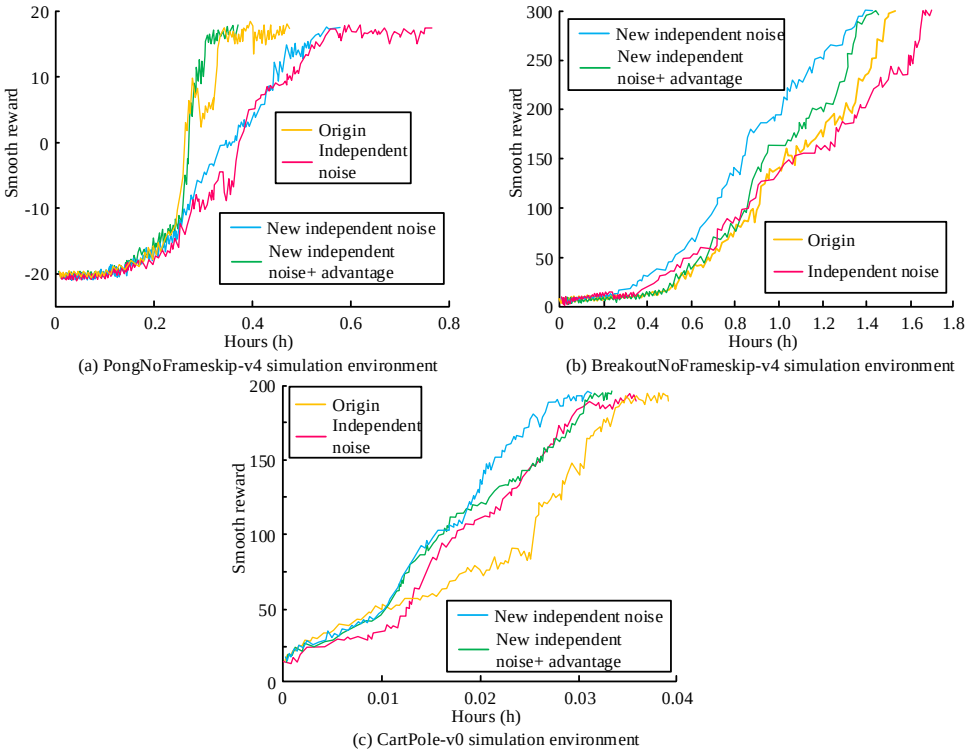


(a) PongNoFrameskip-v4 simulation environment

(b) BreakoutNoFrameskip-v4 simulation environment

(c) CartPole-v0 simulation environment

Fig. 7. Experiment 1 simulation diagram.



(a) PongNoFrameskip-v4 simulation environment

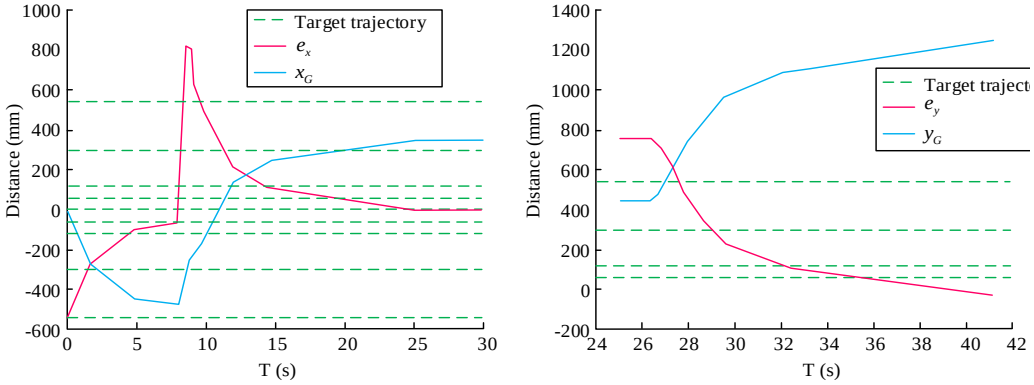
(b) BreakoutNoFrameskip-v4 simulation environment

(c) CartPole-v0 simulation environment

Fig. 8. Experiment 2 simulation diagram.

TABLE II. COMPARISON OF NETWORK OUTPUT AND REAL LOCATION RESULTS

Thing	Network Identification			Manual Annotation		
	$C_x(\mathrm{px})$	$C_y(px)$	$\theta(°)$	$C'_x(\mathrm{px})$	$C'_y(px)$	$\theta'(°)$
Black cup	573.53	246.22	-2.23	573.44	245.12	-2.24
	1087.75	922.75	34.13	1087.69	922.75	33.96
	600.75	243.25	36.61	601.14	242.96	37.10
	799.25	714.53	28.02	799.51	714.98	28.07
	641.02	760.25	-7.11	640.86	762.55	-7.11
	935.52	659.25	35.30	934.22	658.84	36.01
	893.25	722.25	-30.78	892.79	722.01	-30.09
	735.25	444.25	31.93	734.82	444.25	32.10
	696.77	494.25	-26.10	696.93	493.91	-26.60
	460.25	856.23	-28.73	461.75	855.89	-28.16



(a) X direction servo curve diagram

(b) Y direction servo curve diagram

Fig. 9. Servo curve.

In Fig. 9 (a), both e_x and the robot claw coordinates x_G were in the world coordinate system. The experiment began when t=0. At t=7.8s, the item was in Area 2. The robot's claw coordinates changed from 72mm to 800mm, from areas 2 to 5, and its claw began to change its step size. Nine seconds later, until the error turned into 0, this claw tracked an object. In Fig. 9 (b), before 25 seconds, this robot exhibited position servo in the X direction. As the Y direction was close to the object, there could not be relative motion in the Y direction until X tracked a target, so e_y remained invariable before 25 seconds. Once this target is tracked, whether the object shifted is needed to be determined. Therefore, after 25 seconds, the claw remained invariable in the Y direction. If an object's location did not change within two seconds, this claw started servo in the Y direction. e_y began to descend at 27s and entered Area1 around 38s, beginning the next phase of seeking first contact. y_G reached around 1200mm in about 38s, indicating that it comes into contact with an object. The entire curve was very fast at first. As the distance from the object got closer, the speed of the claw gradually decreased.

Fig. 10 shows the force data and claw position changes measured by the sensor. The green line represents the movement of the claw position, while the orange line reflects the changes in force perceived by the sensor. At 39 seconds, the sensor began to sense the gradually increasing force. At 41.6 seconds, if the force reading exceeded 3 Newtons, it indicated that the gripper was starting to grip. Subsequently, the system executed an impedance control strategy, using preset force feedback parameters to adjust the position of the gripper to achieve force zeroing in the F_z axis. When the force reading continued to be zero for 2 seconds, the system determined that the operator was ready to transfer an object. The claw immediately moved this object to a target location, in 2 cm/s.

Table III shows the performance comparison between the improved A3C algorithm and the standard A3C algorithm in different simulation environments. The improved A3C algorithm significantly enhances its learning and decision-making abilities in complex environments by introducing new noise network designs and advantage functions, as well as asynchronous update mechanisms. The specific numerical results show that in the "CartPole v0" task, the improved A3C algorithm score increased by 14.2%; In the game "PongNoFrameskip-v4", the score increased by 17.8%; In the game "BreakoutNoFrameskip-v4", the score increased by 13.5%.

TABLE III. COMPARISON OF OPTIMIZATION PERFORMANCE FOR ROBOT ENVIRONMENT INTERACTION

Experiment Environment	Task Description	Standard A3C Score	Improved A3C Score	Improvement Percentage
CartPole-v0	Balancing a pole on a moving cart	195.4	223.1	14.2%
PongNoFrameskip-v4	Paddle game where score determines the winner	18.5	21.7	17.8%
BreakoutNoFrameskip-v4	Brick-breaking game with score based on the number of bricks broken	430	488	13.5%

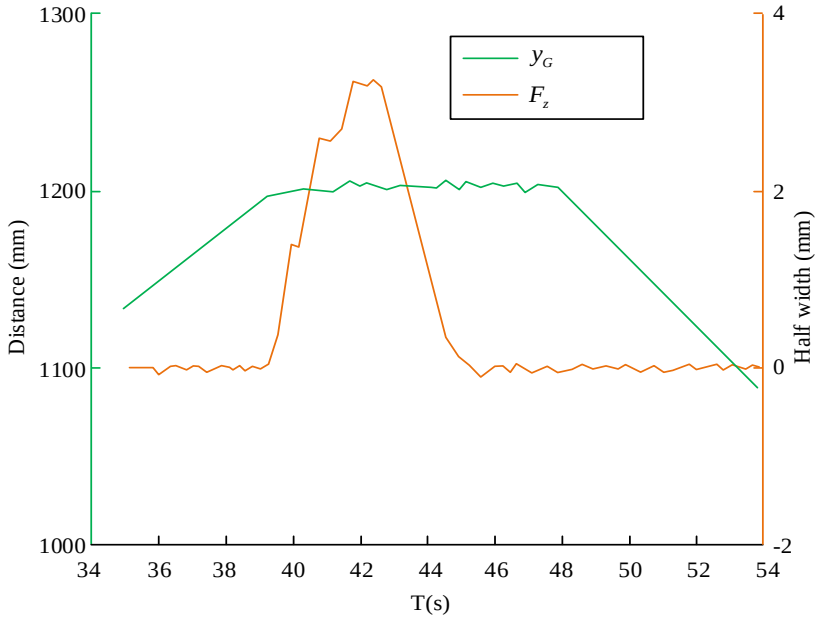


Fig. 10. F_z direction impedance control curve.

V. CONCLUSION

With the developing artificial intelligence and machine learning technologies, robots have significantly improved their ability to perform complex tasks and interact with the environment. A new noise network and advantage function were developed. An asynchronous update mechanism was implemented to improve the algorithm's learning efficiency and adaptability to dynamic environments. On the Gym platform, the improved A3C reduced training steps by 14.4% in the "CartPole-v0" simulation, significantly improving training efficiency. In the "BreakoutNoFrameskip-v4" simulation test, this algorithm increased the broken blocks by about 31.9%, while in the "PongNoFrameskip-v4" simulation, it increased by 7.74%. The actual robot grasping task results showed that this algorithm accurately predicted the position and angle of objects, exhibiting lower noise levels. Therefore, the optimized human-machine delivery interaction of A3C proceeded smoothly and safely. Although the improvement of A3C algorithm has shown performance improvement in specific testing environments, its generalization ability is still limited in diverse environments and unknown contexts. The current research mainly focuses on specific simulation tasks and limited practical applications and has not fully covered the adaptability of algorithms in different environments. In addition, although asynchronous updates and noisy network design have made progress in improving learning efficiency, the applicability of these improvements in a wider range of scenarios still needs further research. Future work will expand the testing scope of algorithms, including more complex simulation environments and more diverse practical application scenarios, to evaluate their generalization and robustness. This will help determine the algorithm's adaptability to novel environments and its potential to maintain efficient performance, thereby promoting further development and practical applications of the algorithm.

REFERENCES

[1] Gurcan F, Cagiltay N E, Cagiltay K. Mapping human–computer interaction research themes and trends from its existence to today: A topic modeling-based review of past 60 years[J]. International Journal of Human–Computer Interaction, 2021, 37(3): 267-280.

[2] Pan S. Design of intelligent robot control system based on human–computer interaction[J]. International Journal of System Assurance Engineering and Management, 2023, 14(2): 558-567.

[3] Ren F, Bao Y. A review on human-computer interaction and intelligent robots[J]. International Journal of Information Technology & Decision Making, 2020, 19(01): 5-47.

[4] Chen Z, Cheng G, Xu Z, Xu K, Shan Y, Zhang J. A3c system: one-stop automated encrypted traffic labeled sample collection, construction and correlation in multi-systems[J]. Applied Sciences, 2022, 12(22): 11731-11757.

[5] Zhu X, Qiu T, Qu W, Zhou X, Wang Y. Path planning for adaptive CSI map construction with A3C in dynamic environments[J]. IEEE Transactions on Mobile Computing, 2021, 22(5): 2925-2937.

[6] Tuli S, Ilager S, Ramamohanarao K, Buyya R. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks[J]. IEEE transactions on mobile computing, 2020, 21(3): 940-954.

[7] Labao A B, Martija M A M, Naval P C. A3C-GS: Adaptive moment gradient sharing with locks for asynchronous actor–critic agents[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 32(3): 1162-1176.

[8] Fan Z, Xu Y, Kang Y, Kang Y, Luo D. Air combat maneuver decision method based on A3C deep reinforcement learning[J]. Machines, 2022, 10(11): 1033-1051.

[9] Du J, Cheng W, Lu G. Resource pricing and allocation in MEC enabled blockchain systems: An A3C deep reinforcement learning approach[J]. IEEE Transactions on Network Science and Engineering, 2021, 9(1): 33-44.

[10] Ye Z, Zhang D, Wu Z G, Yan H. A3C-based intelligent event-triggering control of networked nonlinear unmanned marine vehicles subject to hybrid attacks[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(8): 12921-12934.

[11] Benarbia T, Kyamakya K. A literature review of drone-based package delivery logistics systems and their implementation feasibility[J]. Sustainability, 2021, 14(1): 360-375.

[12] Yu S, Puchinger J, Sun S. Van-based robot hybrid pickup and delivery routing problem[J]. European Journal of Operational Research, 2022, 298(3): 894-914.

[13] Ostermeier M, Heimfarth A, Hübner A. Cost-optimal truck-and-robot routing for last-mile delivery[J]. Networks, 2022, 79(3): 364-389.

[14] Bakach I, Campbell A M, Ehmke J F. A two-tier urban delivery network with robot-based deliveries[J]. Networks, 2021, 78(4): 461-483.

[15] Byrd K, Fan A, Her E S, Liu Y, Almanza S. Robot vs human: expectations, performances and gaps in off-premise restaurant service modes[J]. International Journal of Contemporary Hospitality Management, 2021, 33(11): 3996-4016.

[16] Koren Y, Feingold Polak R, Levy-Tzedek S. Extended interviews with stroke patients over a long-term rehabilitation using human–robot or human–computer interactions[J]. International Journal of Social Robotics, 2022, 14(8): 1893-1911.

[17] Dornelas R S, Lima D A. Correlation Filters in Machine Learning Algorithms to Select De-mographic and Individual Features for Autism Spectrum Disorder Diagnosis. Journal of Data Science and Intelligent Systems, 2023, 3(1): 7-9.

[18] Ye X, Li M, Si P, Yang R, Wang Z. Collaborative and intelligent resource optimization for computing and caching in IoV with blockchain and MEC using A3C approach[J]. IEEE Transactions on Vehicular Technology, 2022, 72(2): 1449-1463.

[19] Zou J, Hao T, Yu C, Jin H. A3C-DO: A regional resource scheduling framework based on deep reinforcement learning in edge scenario[J]. IEEE Transactions on Computers, 2020, 70(2): 228-239.

[20] Jin J S, Tsai Y L, Chang Y C, Tsai W. Low expression of A3C and PLP2 indicating a favorable prognosis in human gliomas[J]. Cellular and Molecular Biology, 2023, 69(7): 71-79.