A multi-critic deep deterministic policy gradient UAV path planning

Runjia Wu, Fangqing Gu*, Jie Huang

Guangdong University of Technology
Guangdong, China. E-mail:fqgu@gdut.edu.cn

2020 16th International Conference on Computational Intelligence and Security (CIS) | 978-1-6654-0445-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/CIS52066.2020.00010

***Abstract*—Deep Deterministic Policy Gradient is a reinforcement learning method, which is widely used in unmanned aerial vehicle (UAV) for path planning. In order to solve the environmental sensitivity in path planning, we present an improved deep deterministic policy gradient for UAV path planning. Simulation results demonstrate that the algorithm improves the convergence speed, convergence effect and stability. The UAV can learn more knowledge from the complex environment.**

***Keywords*-UAV; Reinforcement learning; Deep deterministic policy gradient.**

I. Introduction

In recent years, due to its rapid deployment and controllable mobility, UAV have been widely used in various fields, such as search and rescue [1], multi-UAV cooperation [2], formation flying [3], remote surveillance [4] and other fields. However, how to make UAV safely fly from any source to the destination without obstacles in unknown working environment becomes a hot research topic for researchers. In the face of complex and uncertain environment, many researchers have proposed solutions to UAV navigation problems. The most common method is the motion control problem in the unknown environment such as A-Star [5], artificial potential fields [6], rapidly-exploring random tree (RRT) algorithm [7] and so on [8]. However, these model-based solutions can hardly be applied to practice in the complex uncertain environment.

In order to overcome the above limitations, some researchers propose a learning-based approach. Reinforcement learning (RL) adopts correct action strategies through the interactive learning between agents and the environment. In other words, RL is independent of environment model and prior knowledge. UAV navigation was modeled as a reinforcement learning problem and validated autonomous fight in unknown environments in [9]. DeepMind innovatively combines deep learning (DL) with RL to form deep reinforcement learning (DRL). DRL transforms high-dimensional input into lower-dimensional state that is more realistic. Deep Q Network (DQN) [10], Double DQN [11] and Dueling DQN [12] have been proposed gradually along with DRL research and have a bright performance in path planning. Since DQN action space is discrete, DDPG is proposed to solve the problem of continuous control of engineering problems. Continuity of DDPG is widely used in path planning, but its convergence is often unstable in the face of a complex environment. Paper [13] solved the problem of slow convergence caused by sparse rewards by introducing the reward function of an artificial potential field. Paper [14] started with DDPG experience base combined with simulated annealing algorithm, and accelerated the learning process of DRL through multi-experience pool (MEP). Papers[13], [15] use LSTM to approximate the critical network by combining the current training observation sequence with the historical observation sequence, so that the UAV can break away from the U-shaped obstacle in large path planning. Paper [16] presented three improvements, environmental noise, delay learning and hybrid exploration techniques to improve the robustness of DDPG. Nevertheless, robustness is still a great challenges for UAV path planning.

We extend multi-critic deep deterministic policy gradient method for solving UAV path planning. Simulation results show that the proposed algorithm is effective with strong robust and adaptive capability for solving the path planning of the UAV flying destination under complex environment.

The remainder of this paper is organized as follows: In Selection II, we briefly review reinforcement learning for solving UAV path planning. Section III gives the detailed descriptions of the proposed multi-critic deep deterministic policy gradient method. Section IV provides the simulation results and analyses the empirical results. Finally, we draw a conclusion and future work in Section V.

II. Reinforcement Learning for Solving UAV Path Planning

A. UAV Motion Model

The motion model of UAV is the basis of path planning. Usually the UAV system is controlled by six degrees of freedom, representing three coordinates of the UAV position $[x, y, z]$ and controlling the three freedoms of speed change ϑ, vertical change ϑ_y and rotations around transverse axes ϑ_z. The six degree of freedom kinematics mode $[x, y, z, \vartheta, \vartheta_y, \vartheta_z]$ is used to describe the internal state of the UAV.

For the sake of brevity and without loss of generally, we adopt the kinematic mode of three degrees of freedom instead of six degrees of freedom. Assume that the UAV is fixed at a horizontal altitude, so that UAVs activity is limited to the x-y plane. Ignoring the momentum impact of

978-1-6654-0445-7/20/$31.00 ©2020 IEEE
DOI 10.1109/CIS52066.2020.00010

the UAV during fight, the vector $\zeta = [x, y, \phi]$ is used to simplify the description of the position and motion of the UAV. The control vector of UAV is $a_t = [a_{v,t}, a_{\phi,t}]$, where $a_{v,t}$ is the control speed and $a_{\phi,t}$ is the control yaw angle. Therefore, the vector ζ can be expressed as:

$$\begin{cases} x_{t+1} = x_t + a_{v,t} \times \cos \phi_{t+1}, \\ y_{t+1} = y_t + a_{v,t} \times \sin \phi_{t+1}, \\ \phi_{t+1} = \phi_t + a_{\phi,t}. \end{cases} \tag{1}$$

B. Reinforcement Learning

In reinforcement learning, the agent changes its state through interaction with the environment so as to obtain returns and achieve the optimal strategy. The model is usually expressed using a five-tuple S, A, P, R, γ of a Markov Decision Process (MDP). S is a set of environmental state descriptions. A is a set of all possible actions, $P : S \times A \times S \to [0, 1]$ represents the transition probability of taking an action from S to the next S. $R = S \times A$ represents the immediate reward after the agent takes an action. $\gamma \in [0, 1]$ is the discount factor which represents the difference in importance between future rewards and present rewards. Reinforcement learning (RL) is designed to maximize future rewards, and a set of rewards can be expressed as $R_t^{\gamma} = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$. Based on the reward, RL introduce two functions, the state-valued function when an agent adopts the policy π:

$$V_{\pi}(s_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} r^l r(s_{t+l}, a_{t+l}) \mid s_t\right] \tag{2}$$

And the action-value function:

$$Q_{\pi}(s_t, a_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} r^l r(s_{t+l}, a_{t+l}) \mid s_t, a_t\right] \tag{3}$$

The value function is used to measure the advantages and disadvantages of a certain state or action-state, that is, whether it is worth for an agent to select a certain state or execute an action in a certain state. Figure 1 illustrates the control of agent under reinforcement learning model.

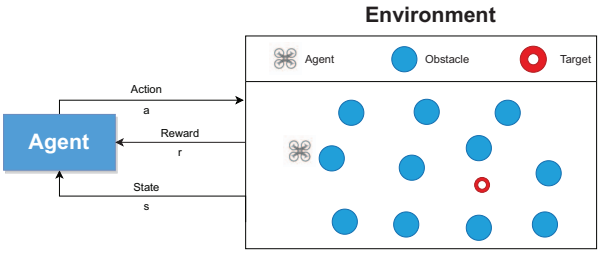


Figure 1: Control of agent under reinforcement learning model

C. State and Action Specification

The state obtained by the environment of UAV is composed of three parts, i.e., the internal state, their interactions with the environment and the location. There are six coordinates $\xi_u = [x, y, x_v, y_v, \vartheta, \psi]$ representing the information about the their internal state, where $[x, y]$ is the absolute position of the UAV, $[x_v, y_v]$ is the velocity component of the corresponding coordinate, $[\vartheta]$ is the direction of flight, and $[\psi]$ is the yaw angle. Secondly, in the real-time interaction between UAV and environment, the surrounding environment state is determined by radar, range finder and other tools. In this paper ,we use range finders to receive the environment state $\xi_f = [d_0, ..., d_N]$, where d_i is the i-th range finder mounted by the UAV. Lastly, the target position of the UAV is expressed as $\xi_T = [x_t, y_t]$. Through the combination of the three observation methods, we can get the final description of the observation space s.

$$s = [x, y, x_v, y_v, \vartheta, \psi, d_0, \cdots, d_N, x_t, y_t]$$

The control of UAV is complicated in the actual situation, which requires multiple commands to achieve the motion of the UAV. In our work, we appropriately selected the UAV's speed and roll as the motion control commands, and we use action $a = [a_v, a_{\psi}]$ to denote the motion, where $a_v \in [-1, 1]$ denotes the ratio of the current speed to the maximum speed, where $a_{\psi} \in [-1, 1]$ is a steering signal.

D. Reward Function

We give up the traditional simple sparse reward model (only when the agent reach the destination can we get the reward) and use a non-sparse reward method. Rewards are set to provide guidance for model learning. Obviously, non-sparse rewards provide more navigation domain knowledge than sparse rewards, and this change does not change the policy invariance of rewards.

The non-sparse reward consist of four constructions:

$$r = r_A + r_B + r_C + r_D \tag{4}$$

Where $r_A = \tau(d_{pre} - d_{cur})$, $d_{pre} - d_{cur}$ represents the change in distance between the current position and the destination. When $d_{pre} > d_{cur}$, r_A is a reward that is related to speed, guiding the UAV to reach the destination quickly. $r_B = -r_{step}$ is a constant penalty advance to the UAV reaches its destination with a minimum number of steps. The UAV should be encouraged to complete its missions as quickly as possible and punished after each transition. $r_C = -A\Delta\varphi + r_{free}$ represents a reward for flying without obstacles. It encourages UAV to shorten its range but at the same time ensure it can explore more space. The UAV should be able to fly toward its targets as soon as possible with penalties for deviations, but it should be encouraged to move towards free space if there are obstacles in the direction of flight. r_D prevents the UAV from getting too

close to the obstacle, and d_{obs} is the minimum distance between the UAV and the obstacle. The UAV should actively avoid obstacles. If it gets close to obstacles, it will be punished greatly, so as to ensure that the UAV can stay away from obstacles. Where τ, A, B, C is a constant which is to control the size of the reward.

III. MULTI-CRITIC DDPG

A. DDPG

Deep Deterministic Policy Gradient (DDPG) adopts the network framework of actor- critic reinforcement learning. The framework of DDPG is illustrated in figure 2. The critic can judge the value of the action based on the actor, and the actor can modify the probability of the action based on the value of the critic. The critic learns the state-action value by minimizing Time-difference (TD) errors:

$$L=\left[r(s_t,a_t)+\gamma Q'\left(s_{t+1},a_{t+1}\mid\theta^{Q'}\right)-Q\left(s_t,a_t\mid\theta^{Q}\right)\right]^2. \quad (5)$$

In DDPG method, the convolution neural networks Q network and μ network are used to approximate the action-value function and state-value function respectively. θ_Q and θ_μ are the parameters of the neural networks. Obviously, we know that the samples obtained by the agent in RL are highly correlated. The researcher uses the reply buffer to address this problem. The correlation of samples is broken by storing experiences (s_t, a_t, r_t, s_{t+1}), and then random samples are taken from experience reply when the network trains.

DDPG is derived from the deterministic policy gradient theorem for MDP. The deterministic policy gradient exists for MDP with continuous action space by this theorem. When the variance of probability policy approaches zero, it is deterministic policy $a_t = \mu(s_t|\theta_\mu)$, i.e.

$$\begin{aligned}\nabla_\theta J(\mu_\theta) &= \int_S \rho^\mu(s)\nabla_\theta\mu_\theta(s)\nabla_a Q^\mu(s,a)\Big|_{a=\mu_\theta(s)} ds \\ &= E_{s\sim\rho^\mu}\left[\nabla_\theta\mu_\theta(s)\nabla Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]\end{aligned} \quad (6)$$

B. MCDDPG

Although DDPG is widely used in UAV path planning because it can well solve the continuous motion space. However, it has poor stability and convergence speed because the actor learning ability depends on the judgment. A multi-critic deep deterministic policy gradient (MCDDPG) is presented in [17]. It uses the average of the values of K critics to approximate instead of the action-value function.

$$Q_{avg}(s,a\mid\theta)=\frac{1}{K}\sum_{i=1}^{K}Q_i(s,a\mid\theta_i). \quad (7)$$

Where $\theta_i \in \theta$ is the parameter of the i-th critic. The average of all critics can diminish the impact of the overestimation problem caused by the individual critic. Combined with the average of critic, we further rewrote the TD error according to equation (5). Thus, the average TD error is:

$$\begin{aligned}L_{avg} = \Big[&r(s_t,a_t)+\gamma Q'_{\text{avg}}\left(s_{t+1},a_{t+1}\mid\theta^{Q'}\right)\\ &-Q_{\text{avg}}\left(s_t,a_t\mid\theta^{Q}\right)\Big]^2.\end{aligned} \quad (8)$$

Where the Q'_{avg} is the average of the target critic networks. Using the same TD error update can cause critics to lose diversity, while a separate update can make a big difference. Therefore, different from DDPG critic network, the local error and global error must be considered when calculating the loss of critic network. According to equation (8), the loss function for i-th critic is defined as

$$L_{mc}=\alpha L_{avg}+\beta L+\eta\left(Q_i(s_t,a_t|\theta_i)-Q_{\text{avg}}(s_t,a_t|\theta)\right). \quad (9)$$

Where α, β and η represent the weight factors.

Algorithm 1 The Proposed MCDDPG Method

Initialize the parameters θ_i^Q of critic networks $Q_i(s,a|\theta_i^Q)$, $i = 1,\cdots,K$ and θ^μ of the actor network $\mu(s|\theta^\mu)$.
Initialize the K target critic networks $Q_i^{'}(s,a|\theta_i^{Q^{'}})$ and actor target network $\mu^{'}(s|\theta^{\mu^{'}})$ with parameters $\theta_i^{Q^{'}} \leftarrow \theta_i^Q$ and $\theta^{\mu^{'}} \leftarrow \theta^\mu$, separately.
Initialize the reply buffer R, parameters α, β, η, and τ.
for $episode = 1 : M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Reset environment and initialize observation state s.
  for $t = 1 : T$ **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise.
    Obtain the reward r_t and observe new state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in R.
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R.
    Update the K critic networks by minimizing loss function in Equation (9).
    Update the actor network with policy gradient.
    Update the parameters of target networks with updating rate τ.
  end for
end for

IV. EXPERIMENTS

In this section, we verify the convergence and effectiveness of the proposed algorithm through simulation results.

A. Experimental platform setting

We built a random environment of different complexity. The terrain of each environment is a rectangular area of 500×600. Some cylindrical obstacles are generated by

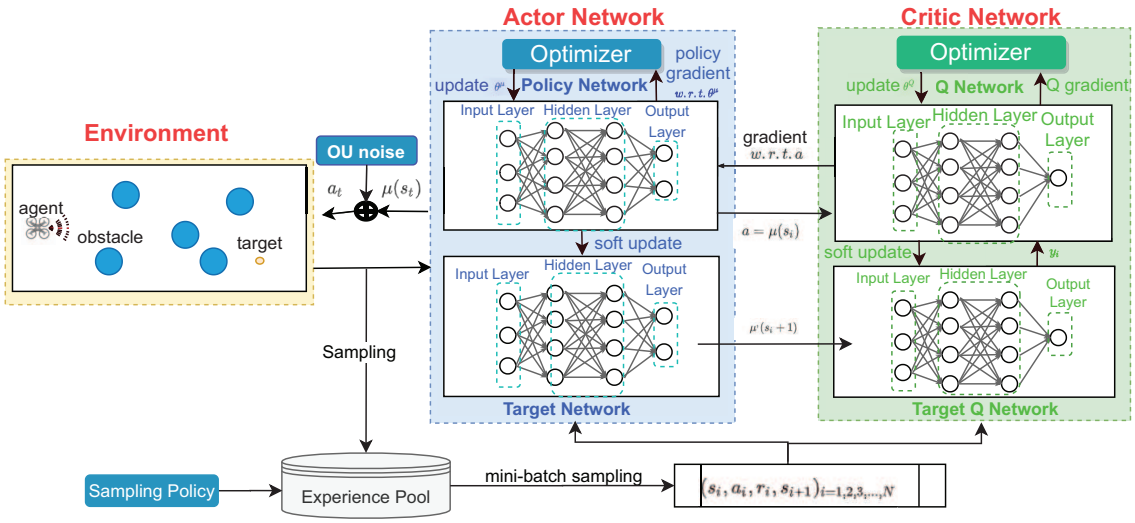
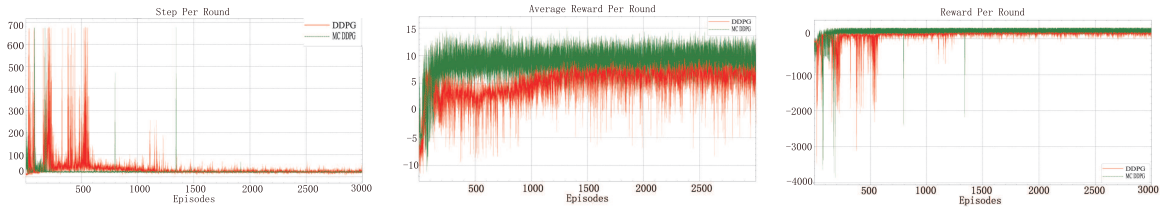


Figure 2: DDPG motion control framework.



(a) (b) (c)

Figure 3: The 3000 set of the training.

a uniform distribution $U(50, 150)$. The UAV is fixed at horizontal altitude of 100 meters and equipped with nine sensors with a detection range of 100 meters. The UAV flies from its initial position to its designated target. The maximum speed of the UAV is limited to $a_{v,max} = 50m/s$ and the maximum yaw is $a_{\psi,max} = \pi/2$.

The critic networks adopt the same network structure of $19 \times 400 \times 300 \times 1$. The observed states as inputs are normalized to 19 dimensions and the actor network composed of $19 \times 400 \times 300 \times 2$ uses 2 dimensions output action to control the UAV. The parameters α, β, η are set to $0.6, 0.4, 0.05$ in MCDDPG. UAV observation and action are normalized to $[-1, 1]$. Adam Optimizer [18] is used to learn network parameters. The learning rate of actor and the critic are set to be 10^{-4} and 10^{-3}. In addition, the discount factor is $\gamma = 0.99$, the soft update rate is $\varepsilon = 0.001$, and other hyperparameters: min-batch size $N = 256$, experience reply $R = 10000$. In addition, noise $\mathcal{N}$ with uniform distribution $U(-0.25, 0.25)$ is used to increase the exploration space. The parameter of reward is set to: $\tau = 1.2, A = 1.5, B = 32, C = 25$, and the maximum value of iteration is $T = 1000$.

B. *Performance of MCDDPG*

In order to verify the performance of MCDDPG, we compare MCDDPG with DDPG on a synthetic test problem. The parameters of DDPG are the sane as used in MCDDPG. As shown in the figure 4, 3000 sets were used to train both models. The success rate of 5-MCDDPG was 88%, while DDPG was 77%. Obviously, MCDDPG has a better success rate and convergence rate than DDPG. 1) MCDDPG uses multi critic network to gain wider access to information from the environment, enabling success rates to start rising at 200 episodes. 2) By averaging the multi-critic network, the over-estimation of the network is avoided and the convergence speed is greatly improved.

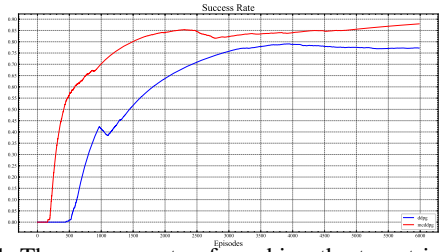


Figure 4: The success rate of reaching the target in training

For more specific verification, we trained the model three

times, intercepted 3000 episodes in 6000 sets and averaged them, selected the number of steps per episode, the average reward per episode, and the total reward per episode, as shown in Figure 4. (The green line represents MCDDPG and the red line represents DDPG). We can see that MCDDPG fluctuates much less than DDPG, and the reward is also better than DDPG. In order to prove the generalization ability of the two algorithms, we further calculated the success rate, collision rate and loss rate of agents. The results are shown in Table I.

Table I: The result of algorithms

	Learning Stage			Exploiting Stage		
	success	collision	loss	success	collision	loss
DDPG	77.2%	18.4%	4.3%	76.2%	11.5%	12.3%
MCDDPG	88.5%	9.8%	1.7%	92.1%	1.6%	6.3%

The result show that the generalization of MCDDPG is better than that of DDPG. The success rate is increased by 3.6% than 1% of DDPG. We can note that the loss rate of model exploiting is greatly improved compared with the mode training. The model training can avoid the collision of UAV with obstacles, which is useful in practical situations.

V. Conclusion and Future Work

In this paper, we extended MCDDPG for solving the path planning of UAV flying destination under complex environment. Simulation results show that there algorithm is superior to the traditional DDPG in path planning. However, some issues remain to be resolved, such as MCDDPG hyper-parameter settings, improvements to non-sparse rewards, experience reply settings and so on.

References

[1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[2] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, "UAV air combat autonomous maneuver decision based on DDPG algorithm," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, 2019, pp. 37–42.

[3] H. SiraRamłrez, R. CastroLinares, and G. PurielGil, "An active disturbance rejection approach to leader-follower controlled formation," *Asian Journal of Control*, vol. 16, no. 2, pp. 382–395, 2014.

[4] R. Stevens, F. Sadjadi, J. Braegelmann, A. Cordes, and R. Nelson, "Small unmanned aerial vehicle (UAV) real-time intelligence, surveillance and reconnaissance (ISR) using onboard pre-processing," *Proc SPIE*, vol. 6967, 2008.

[5] J. K. Howlett, T. W. Mclain, and M. A. Goodrich, "Learning real-time A* path planner for unmanned air vehicle target sensing," *Journal of Aerospace Computing Information and Communication*, vol. 3, no. 3, pp. 108–122, 2012.

[6] G. Luo, J. Yu, Y. Mei, and S. Zhang, "UAV path planning in mixed-obstacle environment via artificial potential field method improved by additional control force," *Asian Journal of Control*, vol. 17, no. 5, pp. 1600–1610, 2015.

[7] R. Kala and K. Warwick, "Planning of multiple autonomous vehicles using RRT," in *IEEE International Conference on Cybernetic Intelligent Systems*, 2011, pp. 20–25.

[8] F. J. Rubio, F. J. Valero, J. L. Suñer, and V. Mata, "Simultaneous algorithm for trajectory planning," *Asian Journal of Control*, vol. 12, no. 4, pp. 468–479, 2010.

[9] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1765–1772.

[10] V. Mnih, K. Kavukcuoglu, and D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2015, pp. 2094–2100.

[12] Z. Wang, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot, and N. D. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2016, pp. 1995–2003.

[13] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29 064–29 074, 2020.

[14] Z. Hu, W. Kaifang, X. Gao, Y. Zhai, and Q. Wang, "Deep reinforcement learning approach with multiple experience pools for UAVs autonomous motion planning in complex unknown environments," *Sensors*, vol. 20, no. 7, p. 1890, 2020.

[15] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[16] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning," *Remote Sensing*, vol. 12, no. 4, p. 640, 2020.

[17] J. Wu, R. Wang, R. Li, H. Zhang, and X. Hu, "Multi-critic DDPG method and double experience replay," in *2018 IEEE International Conference on Systems, Man, and Cybernetics*, 2018, pp. 165–171.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *The 3rd International Conference for Learning Representations*, 2015, pp. 1–15.