Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

# Obstacle Avoidance with Reinforcement Learning and Adaptive Resonance Theory

Lingjian Ye[1,2,3], Yimin Zhou[1,2,3,+]
*1. Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen518055, China*
*2. Guangdong Provincial Key Lab of Robotics and Intelligent System, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences*
*3. CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology*
Shenzhen, China
email: ym.zhou@siat.ac.cn

***Abstract*—The reinforcement learning (RL) of the autonomous mobile agent is one of the actual research topics. It permits mobile agents to interact constantly with their environment and to avoid obstacles. First, this paper presents an algorithm which integrates Deep Deterministic Policy Gradient (DDPG) algorithm and Fuzzy Adaptive Resonance Theory (ART) in order to improve generalization performance of RL. Then the curiosity is introduced to integrate with the first proposed algorithm to solve the problem of slow convergence caused by Fuzzy ART. Results of the simulation experiments demonstrate the effectiveness of the proposed algorithm. It shows that the algorithms perform well in both low-dimension and high-dimension state space.**

***Index Terms*—Obstacle avoidance, Fuzzy Adaptive Resonance Theory, Reinforcement Learning, Deep Deterministic Policy Gradient, curiosity**

## I. Introduction

With the development of computer science, artificial intelligence and information fusion technologies, there is huge breakthrough in the mobile agents from human participation to autonomous control. The auto-planning is the key issue for the mobile agents to perform tasks in unknown environments [1], where obstacle avoidance is the necessary step via sensor perception of the environment with certain obstacle avoidance strategies.

The traditional methods of obstacle avoidance maily includes two methods, Artificial Potential Field algorithm and Virtual Force Field algorithm [2]. Artificial Potential Field algorithm relies on the known environment and can not deal with the narrow gap between obstacles [3]. Virtual Force Field algorithm requires large amount of information storage when dealing with complex scenaries [4]. Futhermore, neural network algorithms can overcome the defects of the traditional methods and make mobile agents more adaptive and intelligent. The data obtained from the sensors can be used as the network input and the action taken by the mobile agents is used as the network output. The supervised algorithm Dronet [5] can sense and avoid obstacles by learning the action data of drivers or cyclists. Reinforcement Learning (RL) has been developed in recent years within the realm of strategy game. Notable examples include AlphaGo [6], AlphaGo Zero [7], Libratus [8], AlphaStar [9].

However, both Supervised Learning and RL algorithms require a large amount of training data. Moreover, it is difficult to migrate the strategy obtained by RL in the training to other simulation environment or real world. Thus, it limits the universality of RL. Researchers have developed a lot of methods to deal with the above problems. In order to alleviate the burden of heavy dependence on the training data, Deepmind proposed the Scheduled Auxiliary Control (SAC-X) which requires an agent to learn the basic auxiliary tasks before training in complex environments [10]. Hybrid Code Networks (HCNs) can be combined with knowledge of concrete field to reduce the requirement of the amount of the training data [11]. In order to solve the problem of poor generalization performance of RL, a hierarchical algorithm combining imitation learning with RL is proposed [12]. The Arcade Learning Environment [13], the humanoid agents thinking [14] and the Upper Confidence Stochastic Game (UCSG) algorithm [15], are continuously put forward to improve the generalization performance of RL.

Although RL can be assisted by auxiliary tasks, knowledge of certain domain, its model still has the problems of high complexity and non-universality. Therefore, the combination of the Fuzzy Adaptive Resonance Theory (ART) [16] and the Deep Deterministic Policy Gradient (DDPG) algorithm [17] (FARTDDPG) is proposed. Then an improved curiosity-driven FARTDDPG (curiosity-FARTDDPG) algorithm is brought forward to accelerate the training.

The remainder of the paper is organized as follows. Section II explains the specific flow of FARTDDPG algorithm and the modification of the curiosity- FARTDDPG algorithm. The simulation experiments are performed to testify the effectiveness of the proposed algorithms in Section III, while the discussion on the experimental results is presented in Section IV. Conclusion is given in Section V.

## II. Related work

### A. Fuzzy ART

Fuzzy ART algorithm is an unsupervised learning algorithm which can cluster the data automatically without setting the

*This work is supported under the National Key Researchand Development Program of China Ref. 2018YFB1305500.
+Corresponding author.

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

number of the categories. It can be implemented in a two-layer neural network, connected by a set of adaptable weights $\omega$: the normalized, complement-coded M-dimensional input $(I \leftarrow [I, 1-I], I_i \in [0,1])$ is presented to the comparing layer, and the discovered categories are represented via the neurons of recognition layer. Fuzzy ART is controlled by the choice parameter($\alpha > 0$), learning rate($\beta \in (0,1]$) and vigilance parameter($\rho \in [0,1]$). The algorithm proceeds as follow: 1. Present input $I$ to the comparing layer and calculate the choice function $S_j$ for each recognition category $j$ :

$$S_j = \frac{\sum_{i=1}^{M} \min(I_j, \omega_{ji})}{\alpha + \sum_{i=1}^{M} \omega_{ji}} \tag{1}$$

Then, select the winning category using a winner-take-all competition:

$$J = \arg\max_{j} S_j \tag{2}$$

2. Perform a vigilance check using the match criterion $c_1$:

$$c_1 : P_J = \frac{\sum_{i=1}^{M} \min(I_j, \omega_{Ji})}{\sum_{j=1}^{M} I_j} \geq \rho \tag{3}$$

If the winning category satisfies $c_1$, then update its weights:

$$\omega_J = (1-\beta)\omega_J + \beta \sum_{i=1}^{M} \min(I_j, \omega_{Ji}) \tag{4}$$

3. If the winning category fails to match $c_1$, then reset it and repeat step 2 until a winner passes. If no existing category succeeds, then a new category is created and its weights are initialized to $I$.

### B. DDPG

DDPG is an algorithm which concurrently learns a Q-function [18] and a policy [19]. It uses off-policy data and the Bellman equation [20] to learn the Q-function, and uses the Q-function to learn the policy.

The Q-Learning side of DDPG is based on the Bellman equation. It is given by

$$\mathrm{Q(s,a)} = \mathop{\mathrm{E}}_{s' \sim D} [\mathrm{r(s,a)} + \gamma \max_{a'} \mathrm{Q(s', a')}] \tag{5}$$

where $s$ and $s'$ are the current state and next state, $a$ and $a'$ is the current action and next action that agents take, $Q(s,a)$ is a action-value function, $r(s,a)$ is the instant reward and $\gamma$ is a discount factor.

Suppose the approximator is a neural network $Q(s,a|\theta^Q)$, with the vector parameter $\theta^Q$, and that $D$ represents a set of transitions $(s,a,r,s')$. Then a mean-squared error function can be set up to explain how closely $\theta^Q$ comes to satisfy the Bellman equation:

$$\mathrm{L} = \mathop{\mathrm{E}}_{(s,a,r,s') \sim D} [(Q(s,a|\theta^Q) - (r(s,a) + \gamma \max_{a'} \mathrm{Q(s', a'|\theta^Q)}))^2] \tag{6}$$

Then it comes to the policy learning side of DDPG. A deterministic policy $\mu$, with the vector parameter $\theta^\mu$, which gives the action can be learned by maximizing the performance objective:

$$PO = \mathop{\mathrm{E}}_{(s,a,r,s') \sim D} [\mathrm{Q(s, u(s|\theta^\mu))|\theta^Q}] \tag{7}$$

## III. Methodology

### A. FARTDDPG

Fuzzy ART algorithm can be combined with DDPG algorithm to improve the generalization performance of the DDPG pre-training model. Fuzzy ART algorithm can realize stable self-increasing unsupervised clustering. As a result, the input pattern of the environmental state is transferred to be a class of patterns as the output after the processing of the Fuzzy ART. Then the output is used as the input to DDPG algorithm.

FARTDDPG algorithm can be implemented in the deep neural network, which contains three sub-networks: Fuzzy ART sub-network, Actor sub-network and Critic sub-network. The Actor sub-network is applied with the aid of target network to make the training more stable [21], where the vector parameter $\theta^{Q'}$ is depending on $\theta^Q$. Therefore, Actor sub-network can be divided into a online network ($\theta^Q$) and a target network ($\theta^{Q'}$) , so does Critic sub-network. Then the Eq. (6) can be rewritten as:

$$L = \mathop{\mathrm{E}}_{(s,a,r,s') \sim D} [(Q(s,a|\theta^Q) - (r(s,a) + \gamma Q'(s_i', \mu(s_j'|\theta^{\mu'})|\theta^{Q'})))^2] \tag{8}$$

where $\theta^{\mu'}$ is the vector parameter of the target policy network.

It should be mentioned that the convolution layers are not included in the network due to absent of image data.

The connection among the sub-networks in FARTDDPG and the data flow are demonstrated in Fig. 1. The pseudocode of the FARTDDPG algorithm in dealing with obstacle avoidance is shown in Algorithm 1.

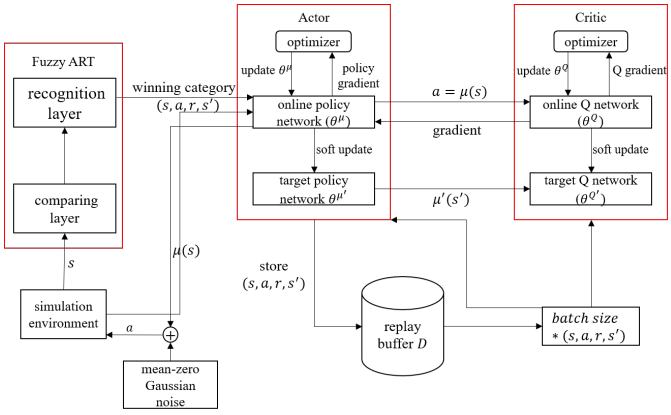


Fig. 1. The connection among the sub-networks in FARTDDPG and the data flow

### B. curiosity-FARTDDPG

Both machines and organisms can benefit from exploratory activities to develop their cognitive abilities. However, there are internal motivations in organisms, such as human's curiosity or competitiveness towards the unknown, which can promote organisms to gain more knowledge faster [22]. In order to solve the problem of slow convergence of FARTDDPG algorithm, following this biological mechanism, curiosity is introduced to integrate with the proposed FARTDDPG algorithm. The curiosity is measured by the difference between the input data I and the winning patterns class $\omega_J$, then it is applied to the design of reward function.

**Algorithm 1** FARTDDPG procedure

Initial Fuzzy ART adaptive weights $\omega$, Q-function parameters $\theta^Q$, policy parameters $\theta^\mu$, empty replay buffer $D$
Initial the target parameters equal to the online parameters $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^{\mu'}$
Initial episode count $e_1 \leftarrow 0$, max global step to train $e_2$
**for** $i$ = 1 to $e_2$ **do**
  **if** input state $I$ is not terminal state **then**
    Obtain the winning pattern class $\omega_J$ and assign it to current state $s = \omega_J$ by Eq. (1), (2), (3), (4)
    Select action $a = \mu(s|\theta^\mu) + N^{MZG}$, where $N^{MZG}$ represents the mean-zero Gaussian noise
    Execute $a$ in the environment
    Obtain the next state $s'$, reward $r$
    Store the transitions $(s, a, r, s')$ in replay buffer $D$
    Randomly sample a mini batch of transitions from $D$
    Compute target $y_i = r_i + \gamma Q'(s_i', \mu'(s_i'|\theta^{\mu'})|\theta^{Q'})$ where $i$ represents the index of transitions
    Update Q-function by one step of gradient descent for the loss given by Eq. (8)
    Update policy by one step of gradient ascent for performance objective given by Eq. (7)
    Soft update target networks with

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    where $\tau$ is the parameter of soft-update, usually takes the value 0.001
    Set $I = s'$
  **else**
    End this episode $e_1 \leftarrow e_1 + 1$ and reset the environment
  **end if**
**end for**

It does not make a big difference between the procedure of FARTDDPG algorithm and curiosity- FARTDDPG algorithm. Only the modifications are explained here.

Define the curiosity $C$ as the Euclidean Metric of $I$ and $\omega_J$:

$$C = \sqrt{\sum_{i=1}^{M} (I_i - \omega_{Ji})^2} \tag{9}$$

Modify the reward as the weighted sum of curiosity and external rewards:

$$r = \eta r^{external} + \xi C \tag{10}$$

where $r^{external}$ represents the external reward (the instance reward $r(s, a)$ mentioned above), $\eta$ and $\xi$ are the weights of the different reward. In this paper, set $\eta = P_J$ and $\xi = 1 - P_J$

Eq. (10) can be rewritten as:

$$r = P_J r + (1 - P_J)\sqrt{\sum_{i=1}^{M} (I_i - \omega_{Ji})^2} \tag{11}$$

Then modify the step 12 in Algorithm 1:

$$y_i = P_J r + (1 - P_J)\sqrt{\sum_{i=1}^{M} (I_i - \omega_{Ji})^2} + \gamma Q'(s_i', \mu(s_j'|\theta^{\mu'})|\theta^{Q'}) \tag{12}$$

## IV. Simulation Environment

Note that the implementation of all algorithms and the simulation environment debugging are all done on Ubuntu 16.04 with python 3.5.

### A. Experimental settings

Two simulation environments, 2D simulation environment designed by ourselves and The Open Racing Car Simulator (TORCS) environment, are used in the experiment.

The 2D simulation environment for training is shown in Fig. 2 and the environment for generalization testing is shown in Fig. 3. The red block represents the car and the blue block represents obstacles. The description of its basic parameters is given by Table 1.

Table 1 The basic parameters in 2D simulation environment

| | |
|---|---|
| Action space | $a \in [-1, 1]$means turning angle $\alpha \in [-6°, 6°]$ |
| State space | Distance vector |
| Size of the form | $600px \times 600px$ |
| Upper limit of sensors | $60px$ |
| Position of sensors | Angles relative to the agent are $0°, 45°, 90°, 135°, 180°$ |
| Agent | The red rectangular object, size: $20px \times 40px$ |
| Obstacle | The blue object, size: random |
| Start point | $[40, 580]$ |

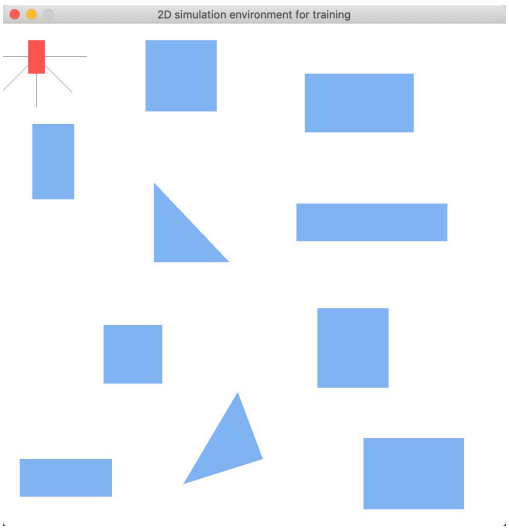


Fig. 2. 2D simulation environment for training

In TORCS environment, the definition of action space, state space and other parameters can be found in the paper about TORCS [23]. And Road Tracks-Alpine 1 is used as the

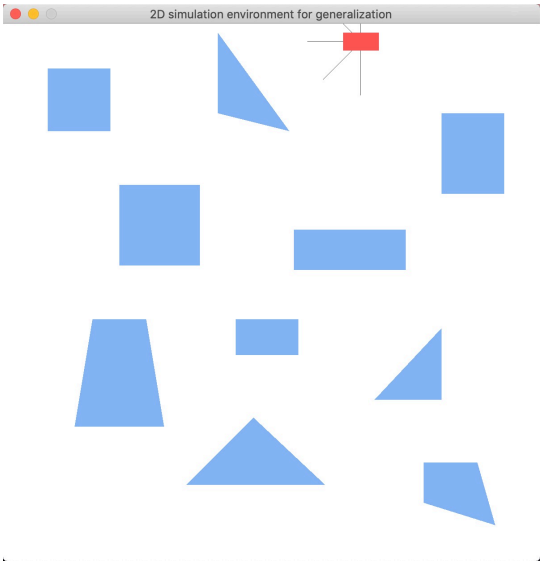


Fig. 3. 2D simulation environment for generalization testing

training dataset and Road Tracks-Ruudskogen is used as the environment for generalization testing.

### B. Definition of the reward function

Usually, different reward functions need to be defined in different simulation environments. In 2D simulation environment, for the FARTDDPG algorithm, the reward function is defined according to whether the agent collides with the obstacle or not.

$$r_{curiosity-FARTDDPG} = \begin{cases} 0 & if\, collision \\ 1 & if\, not\, collision \end{cases} \tag{13}$$

For the curiosity- FARTDDPG algorithm, Eq. (13) is modified according to Eq. (11):

$$r_{curiosity-FARTDDPG} = \begin{cases} (1-P_J)C & if\, collision \\ P_J + (1-P_J)C & if\, not\, collision \end{cases} \tag{14}$$

In TORCS environment, considering the boundaries as obstacles, the reward function is defined as follows:

$$r_{curiosity-FARTDDPG} = \begin{cases} -1, \\ if\, collision\, or\, turn\, around\, or\, 2000\, steps\, without\, 10m \\ V_x \cos\theta - V_y \sin\theta - V_x|trackPos|, \\ otherwise \end{cases} \tag{15}$$

where $V_x$ is the speed of the car along the longitudinal axis of the car, $V_y$ is speed of the car along the transverse axis of the car, $trackPos$ is distance between the car and the track axis. For the curiosity- FARTDDPG algorithm, Eq. (15) is modified according to Eq. (11):

$$r_{curiosity-FARTDDPG} = \begin{cases} -P_J + (1-P_J)C, \\ if\, collision\, or\, turn\, around\, or\, 2000\, steps\, without\, 10m \\ P_J(V_x \cos\theta - V_y \sin\theta - V_x|trackPos|) + (1-P_J)C, \\ otherwise \end{cases} \tag{16}$$

## V. Experimental Results and Discussion

In this paper, the comparative experiments of DDPG, FARTDDPG and curiosity- FARTDDPG algorithms in two simulation environments are set up. And the Indicators for comparison is given by Table 2.

Table 2 The Indicators for comparison

| | |
|---|---|
| avg_episode_reward | Average rewards for each episode of learning tasks |
| q_loss | See Eq. (8) for definition |
| policy_loss | Negative performance objective, See Eq. (7) for definition |

Note that the results of the two simulation experiments are consistent. So TORCS experiment is chosen here for analysis.

### A. Obstacle avoidance in 2D simulation environment

DDPG, FARTDDPG and curiosity-FARTDDPG algorithms are trained 200,000 steps. The q_loss and policy_loss values are recorded for each 10,000 steps during the training period. At the same time, the parameters of current model were used to test in the training environment to obtain the avg_episode_reward values. The results are shown in Fig. 4, Fig. 5 and Fig. 6.

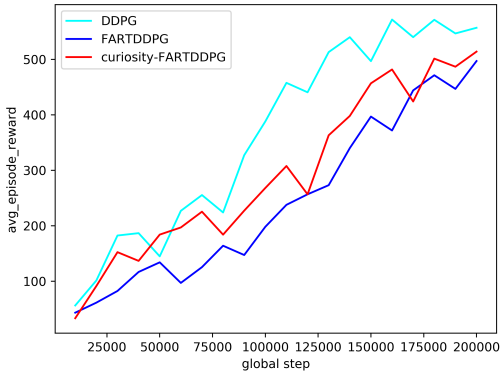


Fig. 4. avg_episode_reward in 2D simulation environment

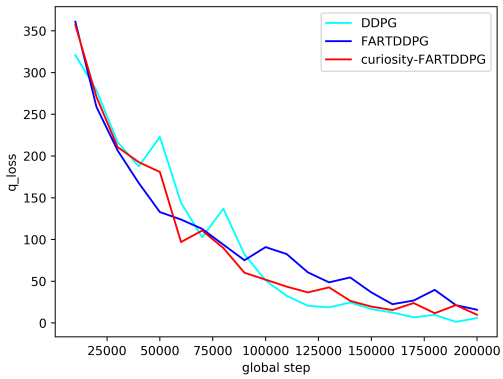


Fig. 5. q_loss in 2D simulation environment

It can be seen from Fig. 5 that the DDPG and curiosity-FARTDDPG algorithms almost have the same rate of gradient descent in q_loss.

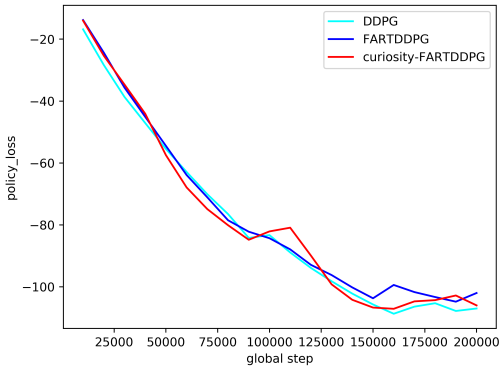


Fig. 6. policy_loss in 2D simulation environment

The convergence of the policy shown in Fig. 6 is consistent with the convergence of the Q-function shown in Fig 5.

In the generalization experiment, the avg_episode_reward is used as the indicator of generalization performance. The result is given by Table 3.

Table 3 2D Experiments on generalization of three algorithms

| Index | DDPG | FARTDDPG | Curiosity- FARTDDPG |
|---|---|---|---|
| 1 | 402.1587914 | 463.9824723 | 485.2187719 |
| 2 | 428.2318722 | 477.4526018 | 502.6054528 |
| 3 | 443.5647921 | 436.1044782 | 521.7320514 |
| 4 | 392.1581487 | 444.8412965 | 495.5421385 |
| 5 | 405.4825972 | 432.4214802 | 528.4587021 |
| 6 | 410.5682391 | 439.0021472 | 540.2870250 |
| 7 | 450.5819462 | 457.6185439 | 503.9726114 |
| 8 | 420.2124387 | 474.2588754 | 522.1586210 |
| 9 | 387.8738531 | 450.5487125 | 509.1872348 |
| 10 | 399.0856812 | 461.2487101 | 483.1678992 |
| Avg | 413.9918360 | 453.7479318 | 509.2330508 |

In the 2D generalization experiments, the curiosity-FARTDDPG algorithm obtains the maximum of the average reward of avg_episode_reward. It means that the generalization performance of the curiosity-FARTDDPG algorithm is the best among the three algorithms.

### *B. Obstacle avoidance in TORCS environment*

As the state space and action space of 2D simulation environment are low-dimensional, the TORCS environment is set up to supplement the integrity of the experiment.

The three algorithms are trained with 2.1 million steps. The q_loss and policy_loss values are recorded for each 10,000 steps. The parameters of the current model are used to test in the training environment to obtain the avg_episode_reward values for each 30,000 steps. The results are shown in Fig. 7, Fig. 8 and Fig. 9.

Fig. 7 shows that the maximum of avg_episode_reward value of DDPG algorithm is the largest. It because that the clustering used by the proposed two algorithms leads to "fuzziness" when identifying the environment. This "fuzziness" means that a class of patterns corresponds to one action. However, it is that one pattern corresponds to one action in DDPG algorithm. But curiosity-FARTDDPG algorithm performs better on the general trend of the avg_episode_reward. There is a tendency for agents with curiosity to explore the unknown world. In other words, curiosity allows agents to obtain a more complete strategy of obstacle avoidance, even though the state is fuzzy. The general trend of the avg_episode_reward in the 2D experiment is different from the TORCS experiment, and the reason is the low-complexity of the state space and action space in the 2D simulation environment. As a result, a class of patterns corresponding to one action performs better in the 2D simulation environment.

It can be seen from Fig. 8 and Fig. 9 that the convergence process of the three algorithms are different in oscillation. But all the three algorithms almost converge after training. The most important is that the rate of convergence relation among the three algorithms is $FARTDDPG < curiosity - FARTDDPG < DDPG$. The "fuzziness" causes the agent to learn a strategy suitable for the current training environment that takes more steps than the DDPG algorithm. But The ability to explore with curiosity makes it faster to train a FARTDDPG model.

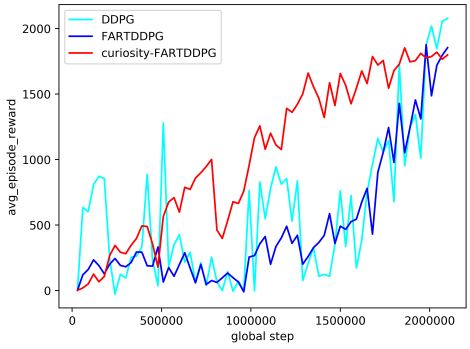


Fig. 7. avg_episode_reward in TORCS environment

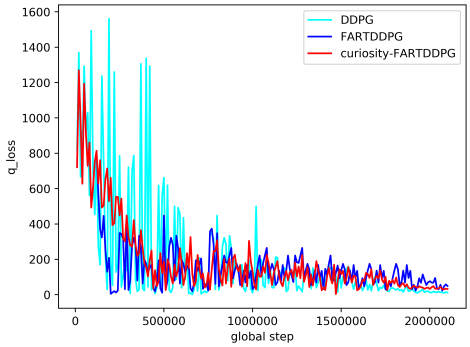


Fig. 8. q_loss in TORCS environment

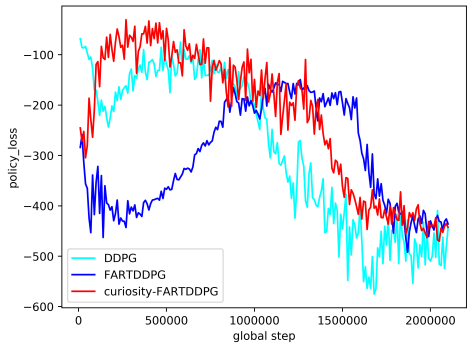


Fig. 9. policy_loss in TORCS environment

The avg_episode_reward is still used as the indicator of generalization performance. The result is shown in Table 4.

Table 4 TORCS Experiments on generalization of three algorithms

| Index | DDPG | FARTDDPG | Curiosity- FARTDDPG |
|---|---|---|---|
| 1 | 1423.4877512 | 1608.2657487 | 1615.0354857 |
| 2 | 1355.1812371 | 1675.0455745 | 1586.1158741 |
| 3 | 1379.3079422 | 1712.4673205 | 1660.7321582 |
| 4 | 1540.8125469 | 1592.2451154 | 1625.5108713 |
| 5 | 1457.2272813 | 1648.6064821 | 1691.2198741 |
| 6 | 1360.2079857 | 1642.2379872 | 1614.0560822 |
| 7 | 1430.7457861 | 1703.0189726 | 1652.6510528 |
| 8 | 1320.9582124 | 1622.1678135 | 1645.6008758 |
| 9 | 1367.8956473 | 1549.1878915 | 1669.3562472 |
| 10 | 1405.0785421 | 1683.4854633 | 1621.9223014 |
| Avg | 1404.0902932 | 1643.6728369 | 1638.2200823 |

In the TORCS generalization experiment, the average of avg_episode_reward of the proposed algorithms is almost equal, which are larger than the DDPG algorithm. Although the ”fuzziness” slows down the rate of convergence, it can improve the generalization performance outside the training environment. Even as an unknown state, it is likely to be clustered into an existing patterns class.

## VI. Conclusion and Future Work

In this paper, two methods based on ART, RL and biological internal motivations are introduced to deal with the obstacle avoidance. In order to improve the generalization performance of DDPG algorithm, the FARTDDPG algorithm is proposed. Then in order to solve the problem of slow convergence of FARTDDPG algorithm and make agents more exploratory, curiosity is introduced to integrate with the proposed FARTDDPG algorithm. The results demonstrate that the proposed methods can effectively work in simulation environments. The generalization experiments show that the proposed algorithms perform better in the complex environments.

Future work will focus on modifying and deploying the proposed method in real agents, such as unmanned vehicle and robots.

## Acknowledgement

The author would like to thanks the research students in the Gongdong Provincial Engineering and Technology Research Center of Intelligent Unmanned System and Autonomous Environmental Perception, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences.

## References

[1] Y. Cheng, W. Zhang. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing*, 2018, 272: 63-73.

[2] L. Tai, S. Li, M. Liu. A deep-network solution towards model-less obstacle avoidance. *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2016: 2759-2764.

[3] Y. B. Chen, G. C. Luo, et al. UAV path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 2016, 47(6): 1407-1420.

[4] A. Gianibelli, I. Carlucho, et al. An obstacle avoidance system for mobile robotics based on the virtual force field method. *2018 IEEE Biennial Congress of Argentina*, 2018: 1-8.

[5] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, et al. DroNet: Learning to Fly by Driving. *IEEE Robotics and Automation Letters*, 2018, 3(2): 1088-1095.

[6] D. Silver, A. Huang, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016, 529(7587): 484-492.

[7] D. Silver, J. Schrittwieser, et al. Mastering the game of go without human knowledge. *Nature*, 2017, 550(7676): 354-359.

[8] N. Brown, T. Sandholm. Safe and nested subgame solving for imperfect-information games. *Advances in neural information processing systems*, 2017: 689-699.

[9] K. Arulkumaran, A. CULLY, J. Togelius. Alphastar: An evolutionary computation perspective. *arXiv preprint arXiv:1902.01724*, 2019.

[10] M. Riedmiller, R. Hafner, et al. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[11] J. D. Williams, K. Asadi, G. ZWEIG. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*, 2017.

[12] H. M. Le, N. Jiang, et al. Hierarchical imitation and reinforcement learning. *arXiv preprint arXiv:1803.00590*, 2018.

[13] M. G. Bellemare, G. Marc, et al. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013, 47: 253-279.

[14] B. M. Lake, T. D. Ullman, et al. Building machines that learn and think like people. *Behavioral and brain sciences*, 2017, 40: 1-72.

[15] C. Y. Wei, Y. T. Hong, C. J. LU. Online reinforcement learning in stochastic games. *Advances in Neural Information Processing Systems*, 2017: 4987-4997.

[16] G. A. Carpenter, S. Grossberg, D. B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 1991, 4(6): 759-771.

[17] S. Gu, T. Lillicrap, I. Sutskever, et al. Continuous deep q-learning with model-based acceleration. *International Conference on Machine Learning*, 2016: 2829-2838.

[18] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 1988, 3(1): 9-44.

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, et al. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 2000: 1057-1063

[20] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 1954, 60(6): 503-515.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 2015, 518(7540): 529-533.

[22] J. Ceha, N. Chhibber, J. Goh, et al. Expression of Curiosity in Social Robots: Design, Perception, and Effects on Behaviour. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. ACM*, 2019: 1-12.

[23] D. Loiacono, L. Cardamone, P. L. Lanzi. Simulated car racing championship: Competition software manual. *arXiv preprint arXiv:1304.1672*, 2013.