International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 914–919
DOI: https://doi.org/10.2991/ijcis.d.200615.002; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Accuracy Improvement of Autonomous Straight Take-off, Flying Forward, and Landing of a Drone with Deep Reinforcement Learning

Che-Cheng Chang[1], Jichiang Tsai[2,*], Peng-Chen Lu[3], Chuan-An Lai[1]

[1] *Department of Information Engineering and Computer Science, Feng Chia University, No. 100, Wenhua Rd., Xitun Dist., Taichung City 407, Taiwan (R.O.C.)*
[2] *Department of Electrical Engineering and Graduate Institute of Communication Engineering, National Chung Hsing University, No. 145, Xingda Rd., South Dist., Taichung City 402, Taiwan (R.O.C.)*
[3] *Graduate Institute of Communication Engineering, National Chung Hsing University, No. 145, Xingda Rd., South Dist., Taichung City 402, Taiwan (R.O.C.)*

ARTICLE INFO

Article History
Received 31 Jan 2020
Accepted 09 Jun 2020

Keywords
Drones
Deep reinforcement learning
Q-learning
Autonomous flight

ABSTRACT

Nowadays, drones are expected to be used in several engineering and safety applications both indoors and outdoors, e.g., exploration, rescue, sport, entertainment, and convenience. Among those applications, it is important to make a drone capable of flying autonomously to carry out an inspection patrol. In this paper, we present a novel method that uses ArUco markers as a reference to improve the accuracy of a drone on autonomous straight take-off, flying forward, and landing based on Deep Reinforcement Learning (DRL). More specifically, the drone first detects a specific marker with one of its onboard cameras. Then it calculates the position and orientation relative to the marker so as to adjust its actions for achieving better accuracy with a DRL method. We perform several simulation experiments with different settings, i.e., different sets of states, different sets of actions and even different DRL methods, by using the Robot Operating System (ROS) and its Gazebo simulator. Simulation results show that our proposed methods can efficiently improve the accuracy of the considered actions.

1. INTRODUCTION

The application of Unmanned Aerial Vehicles (UAVs), also known as drones, has been increasingly prevalent in recent years. The drones can be used in different applications both indoors and outdoors. However, in the course of achieving the above tasks, the Global Positioning System (GPS) may not be always precise and available due to the effects of signal attenuation and multi-path propagation [1,2]. In the literature, there have been three major classes of methods for positioning the drones. First, the sensor-fusion method relies on multiple sensors in order to gather more data for the pose estimation of the drone. On the other hand, the device-assisted method relies on the use of ground sensors for estimating the position and trajectory of the drone. The last one is the vision-based approach, which analyzes the geometric features to find the flying path. Thus, for the purpose of eliminating the disadvantage of GPS, we intend to utilize onboard sensors, e.g., cameras, LiDARs, and so on, equipped by a drone to realize the vision-based approach for performing tasks.

In this work, we use the typical Reinforcement Learning (RL) concept, Q-Learning [3,4], to control a drone so that the drone can autonomously take-off, fly forward, and land in a straight line the best it can. One remarkable advantage is without the need of human supervision, as well as allowing the drone to learn how to use high-level actions autonomously. The drone obtains the required information simply through its onboard cameras instead of other expensive sensors. Therefore, the design is not affected by the aforementioned issue of GPS and can be applied to both indoor and outdoor scenarios. More specifically, the drone uses either its front or bottom camera to detect an ArUco marker [5–7]. The marker is essential for the purpose of positioning the drone during the course of autonomous flight in our work. By taking the center of a marker as a reference, the drone can calculate the relative position and orientation information between itself and the marker. Then it will take the corresponding action based on the control strategy of some Q-learning derivatives immediately.

At last, we present a simulation study based on a simulator of the robot operating system (ROS) [8,9], called Gazebo [10], where we use its Parrot AR. Drone 2.0 module to conduct our simulation experiments. In the experiments, we investigate the effects of taking different sets of actions under different sets of states, as well as adopting distinct Deep Reinforcement Learning (DRL) methods derived from the basic Q-learning concept, namely, Deep Q Networks (DQNs), Double DQN, and Dueling DQN.

[*] *Corresponding author. Email: jichiangt@nchu.edu.tw*

The simulation results show that the use of different sets of DRL approaches, states, and actions will have distinct accuracy improvements of the flight results of a drone as well as convergence speeds of the corresponding learning processes. Hence, by choosing the proper DRL method and the sets of states and actions, we can obtain better efficiency, accuracy, and convergence for a drone taking off, flying forward, and landing straight. Notice that although there are some related researches in the literature [11], they only exploited the concept of double DQN to land the UAV. Yet in this work, we further give a comprehensive discussion on the effects of taking different sets of actions under different sets of states, as well as adopting distinct DRL methods.

The rest of this paper is organized as follows: In Section 2, we review the preliminary knowledge related to the necessary components for our work, i.e., Q-learning approaches, ArUco marker, ROS, and Gazebo. Then we elaborate our methods of autonomous flight in Section 3. In Section 4, we analyze the simulation results corresponding to distinct experimental settings. Lastly, we conclude this paper and discuss some possible future works in Sections 5.

2. METHODOLOGICAL FRAMEWORK

2.1. Robot Operating System

The ROS is a flexible framework for creating robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of designing complex and robust robot behavior across various platforms. Since creating robust robot software is very hard, no single individual, laboratory, or institution can hope to do it on their own [8,12,13]. Thus ROS was built to encourage collaborative robotics software development. For instance, one institution might contribute a system for producing maps, and another laboratory might use maps to navigate.

In ROS, a process is represented as a node in a graph structure and connected by edge(s) called topic(s). Hence, a node can pass messages to one another through topics, e.g., making a service call, providing a service, and so on. ROS uses peer-to-peer communication between all node processes. This decentralized architecture works well since one robot application often consists of many kinds of networked computer hardware.

2.2. Gazebo

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate robots in complex indoor and outdoor environments [10,14,15]. While similar to game engines, Gazebo can offer physics simulation at a higher degree, e.g., a suite of sensors, interfaces for both users and programs, and so on. Furthermore, since Gazebo was integrated with ROS, it has become one of the primary tools used in the ROS community.

There are many existing modulus in the Gazebo simulator for different experiments. Particularly, we use the Parrot AR. Drone 2.0 module in our simulation experiments. The Parrot AR. Drone 2.0 includes ARM Cortex A8 processor with DSP and supports Linux. Hence, it can be exploited to implement a real system easily. Moreover, the Parrot AR. Drone 2.0 is equipped with two camera modulus, front, and vertical cameras. The specification of cameras mounted on it is as follows:

- Front camera: HD(720P) sensor; 92° lens; 30FPS video.
- Vertical camera: QVGA(240×320) sensor; 64° lens; 60FPS video.

2.3. ArUco Marker

Camera pose estimation is a common and important problem in many applications requiring a precise localization in the environment, e.g., augmented reality, virtual reality, and robotics. It is one of the vision-based approaches to analyze the geometric features to position the drone and find the flying path. Although some researches like to seek natural features, fiducial markers are still attractive because they are easy to detect and achieve high speed and precision. For dealing properly with this issue, there have been many researches of different fiducial markers proposed in the literature [16]. Among those existing contributions, the design of an ArUco marker is one feasible way for our work [16,17]. Particularly, an ArUco marker is composed of many binary bits. There are five bits in every row, where the second and fourth bits serve as information bits and the other three bits are used for error detection. Moreover, each marker has its unique ID. It is able to recognize the ID of the marker by the information bits in it, and there are totally 1024 different IDs.

2.4. Q-learning

Q-learning is an algorithm commonly used in RL for the estimation side of a problem. More specifically, Q-learning provides a solution for the control part and tries to estimate the action-value to take the best possible action for the problem. It uses the state-action-reward-state tuples as experience to realize estimation [18]. Namely, it consists of an agent that will interact with an external environment represented by states. At time t, the agent will be given a reward r after taking an action a in state s. Then the agent will transfer to the next state s'. By the recursive operation, we can calculate the expected total reward in the future corresponding to each state and action, Q(s, a), called the Q-value, which is updated as follows:

$$Q_{(s,a)} \leftarrow (1-\alpha)Q_{(s,a)} + \alpha(r + \gamma \max_{a\prime} Q_{(s\prime,a\prime)}), \tag{1}$$

where $\alpha \in (0,1]$ and $\gamma \in [0,1]$ are the learning rate and discount rate, respectively. The Q-value of each state and action will be stored in a table called Q-table. The goal of RL is to find a policy for choosing a proper action at a certain state to accumulate the rewards as many as possible at successive steps.

2.5. Deep Q Network

The term "Deep" in the "Deep Q Networks" refers to the use of "Deep Convolutional Neural Networks" (DCNNs) in the DQNs [18]. DCNNs are inspired by the way that human's visual cortex area of the brain understands the images received by the eyes. While discussing about the state-formulations of a Q-learning approach, for image inputs, the state should be abstracted humanly due to the efficiency of implementation. Namely, the agent should be intelligent enough to make sense of these visual states. Hence, we can enable the Q-learning agent to simplify the state of raw image pixels by utilizing DCNNs.

2.6. Double DQN

Generally, the state-space and state-size may be extremely large, and it may take a lot of time for the agent to learn sufficient information about the environment and then determine which state/action may lead to the best reward. Consequently, the agent of a DQN may get stuck to exploiting the explored and estimated state-action combinations that have relatively higher values but not the highest ones because DQN is known to sometimes learn unrealistically high action values due to its *max* operation. This may lead to the overestimation of Q-values for some state-action combinations leading to suboptimal training. Hence, the Double DQN algorithm is proposed to solve this issue by selecting the action on the basis of a network, called the Online Q Network, but using the value corresponding to this particular state-action from another network, called the Target Q Network [18]. The equation is shown as follows:

$$Q_{(s,a)} \leftarrow r(s,a) + \gamma Q(s', \underset{a\prime}{\operatorname{argmax}} Q'_{(s\prime,a\prime)}), \tag{2}$$

where Q' is for action selection and Q for action evaluation. Therefore, it helps us reduce the overestimation of Q-value.

2.7. Dueling DQN

The DQN and Double DQN models are both sequential architectures. Particularly, all the neurons in a layer are connected only to the neurons in one layer before and one layer after their own layer. Namely, there are no branches or loops existed in these models. In the Dueling DQN model, which is a nonsequential architecture, the model layers will branch into two different subnetworks, i.e., estimation and advantage networks. Each has its own connected layer and output layers. Particularly, by splitting the network into two channels, Dueling DQN can learn which states are valuable without learning the effect of each action at each state. Such a skill is useful for some states where their actions will not affect the environment. Namely, it is not necessary to compute the value of each action, and thus helps us accelerate the training procedure. Furthermore, by decoupling the estimation of two channels, it can help us find much more reliable Q-values [18]. Dueling DQN can be represented mathematically as below:

$$Q_{(s,a;\theta,\alpha,\beta)} \leftarrow V_{(s;\theta,\beta)} + (A_{(s,a;\theta,\alpha)} - \max_{a\prime} A_{(s,a\prime;\theta,\alpha)}), \tag{3}$$

where "A" represents the advantage value, " θ" the parameter vector of the convolutional layer, " α" the parameter vector of the advantage network, and " β" the parameter vector of the state-value function.

As a result, we can summarize that Q-learning can be considered as the fundamental concept of all existing DRL approaches, which utilizes a table to store the quantified effects of all sets of states and actions. However, since the number of states in a real-world problem is too much to be defined, Q-learning is almost infeasible in real-world problems. Then DQN made up the above deficiency by combining the neural network and Q-learning, which can abstract the states for image inputs humanly. Hence, the Q-learning concept is feasible now due to the introduction of DQN. However, DQN will cause the over-fitting problem and thus reduce the learning performance. So Double DQN was then proposed to deal properly with this issue. Double DQN remains the same as DQN except the updating function. On the other hand, during the learning procedure, some states are pointless such that computing resources are wasted upon trying to find the best actions to take for these states. Thus Dueling DQN was introduced to divide the Q-network into two channels to separately compute the quantified effects of states and actions to avoid the aforementioned problem. Namely, the quantified effects of states can be evaluated regardless of the effects of subsequent actions.

3. OUR METHODS

In this section, we start to introduce our work and methods in detail. First, our work intends to make a drone take-off stably, fly forward straight, and land on the target platform precisely. Particularly, in our working scenario, for the purpose of positioning the drone during the period of autonomous flight, there are totally three ArUco markers used for reference. By taking the center of a marker as a reference, the drone can obtain the relative position and orientation information between itself and the marker. Then it will take the corresponding action according to the obtained position and orientation information and the control strategy of a Q-learning derivative immediately. The properties of these three ArUco markers are shown below:

- The first marker is placed under the AR. Drone to indicate the place where the drone takes off;
- The second one is put on a front wall for the drone to head on;
- The last marker is on the ground as the landing target.

Here, the learning procedure is described, and the experimental scenario is shown in Figure 1. First, the drone is placed on the first marker whose ID is 5 before it takes off, where is the initial position for every flight. Then the front camera of the drone will capture the second marker with ID 0 on the front wall after it takes off from the initial position to a specific height. As soon as the drone reaches the height and finds marker 0, it will start to fly toward this marker until its bottom camera captures the third maker whose ID is 10 on the landing platform. Then the drone descends vertically and finally lands on the landing platform based on the detection of its bottom camera.

The take-off, flying forward, and landing procedures of the drone are all trained with the proposed methods based on the Q-learning concept. Figure 2 shows the states and rewards adopted by our method. We consider two different sets of states. Part (a) in Figure 2 shows that the image captured by the camera is divided into a 3 by 3 grid with 10 different states ($s_0 \sim s_9$); whereas part (b) in Figure 2 shows that the image is partitioned into a 5 by 5 grid with 26 states ($s_0 \sim s_{25}$). The red number on the figure indicates the corresponding reward of each state. On the other hand, we consider two different sets of actions for the control strategies based on the Q-learning approaches in the experiments:

- Four-action set: up, down, left, and right.
- Eight-action set: up, down, left, right, top-left, top-right, bottom-left, and bottom-right.

In the last part of this section, we use a block diagram to explain the flowchart among ROS, Gazebo and our methods, which is shown in Figure 3. First, node *ar_track_alvar* gets images from node *ardrone* through topic */ardrone/bottom/image_raw* and topic */ardrone/front/image_raw* for detecting the ArUco markers. According to the information from *ar_track_alvar* and the information related to flight status of the drone from *ardrone*, node *DQN* can obtain the corresponding position and orientation information between the ArUco markers and the drone. Then it sends the reactive action instruction based on the control strategy of the DQN approach to node */cmd_vel* immediately. Accordingly, node *Gazebo* will operate the drone and then update all information of the drone and environment. The updated information will also be sent to *ardrone*. Similarly, for the applications of Double DQN/Dueling DQN, we just need to replace node *DoubleDQN/DuelingDQN* with *DQN*.

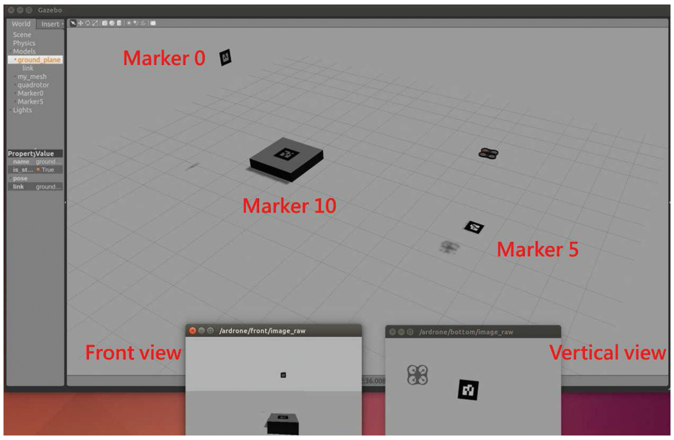

Figure 1 The experimental scenario.

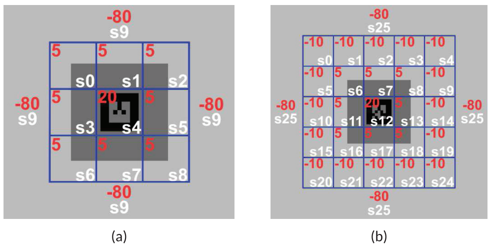

Figure 2 States and rewards adopted by our methods: (**a**) Dividing into a *3* by *3* grid with *10* different states .

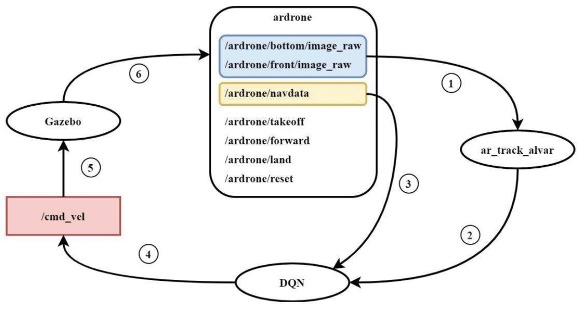

Figure 3 The flowchart of our architecture.

4. SIMULATION RESULTS

In this study, we employ the Gazebo simulator of ROS to implement and verify our methods. Moreover, we use its Parrot AR. Drone 2.0 module in the simulation experiments. During a flight, Gazebo will calculate the average distance between the center of the drone and that of the corresponding marker to be the accuracy degree of this flight. Our experimental settings are as follows:

- Three procedures: Drone take-off, flying forward, and landing;
- Four combinations of state and action sets: 10 states with 4 actions, 10 states with 8 actions, 26 states with 4 actions, and 26 states with 8 actions;
- Three learning approaches: DQN, Double DQN, and Dueling DQN;
- Each procedure with every combination of state and action sets based on each learning approach is executed 1000 times.

Moreover, we have posted parts of our simulation experiment videos in YouTube (https://youtu.be/BFH6rjyR-vo).

First, for the drone take-off, flying forward, and landing procedures based on DQN, Figures 4–6 separately show the average accuracy degrees with respect to corresponding markers of every 50 flights for the four combinations of state and action sets. Note that the reason for considering every 50 flights as a whole is that doing so can present the experimental results more clearly. According to the results, we can find that the larger adopted number of states is, the

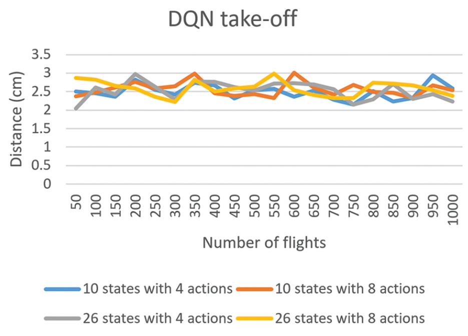

Figure 4 The accuracy degree with respect to marker 5 for drone take-off based on Deep Q Network (DQN).

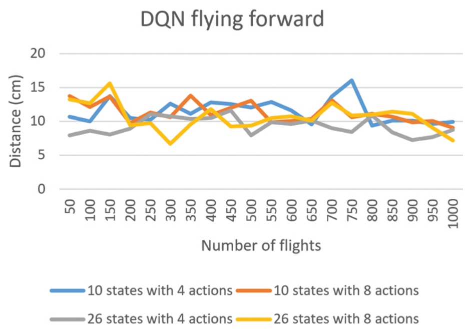

Figure 5 The accuracy degree with respect to marker 0 for drone flying forward based on Deep Q Network (DQN).

higher the accuracy degree becomes. Moreover, with more actions able to be exploited, the learning process of the drone converges more quickly. Particularly, for the drone take-off, which is the simplest one among all procedures, its control strategies with the four combinations of state and action sets based on DQN can all achieve the highest accuracy (about 2.5cm). Since the experimental results of Double DQN and Dueling DQN are similar as that of DQN, we do not present the results here for the reason of simplicity.

On the other hand, we start to consider the three procedures under the same state and action set but based on different learning approaches, i.e., DQN, Double DQN, or Dueling DQN. Some experiment results for drone take-off are first shown in Figures 7 and 8. Similarly, for such a simplest procedure, the two extreme scenarios with most and least number states and actions, i.e., 10 states with 4 actions and 26 states with 8 actions, can still achieve high accuracy (about 2.5cm). Particularly, after being trained 800 times, the accuracy degrees based on Dueling DQN converges much more stably due to the property of Dueling DQN that by decoupling the estimation of two channels, this learning technique can help us obtain much more reliable Q-value for each action.

Next, for the flying forward procedure, we consider the aforementioned two extreme state and action sets as well. According the corresponding results shown in Figures 9 and 10, we can find that among the three approaches, in spite of larger error values during the training course, Dueling DQN enjoys the best accuracy degree after being trained 800 times. Finally, since the experimental results for drone take-off and landing are similar, we do not present the results for landing here.

According to those figures shown above, we can learn the efficiency in convergence of the experiments. In the last part of this section, we present some important and interesting statistics of these experiments with Table 1. Specifically, since all experimental results for drone take-off and landing based on the three learning approaches are similar and have smaller error values, it is hard and meaningless to discuss them. Hence, we only consider the experimental results of flying forward procedures under two extreme scenarios with the most and least number of states and actions, respectively. In Table 1, we can observe that all the three learning approaches gets better average error values in the scenario with 26 states and 8 actions. Furthermore, Dueling DQN possesses the best improvement from

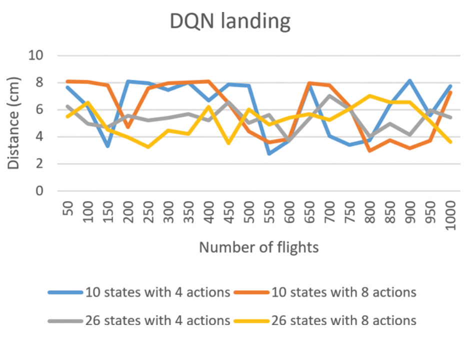

Figure 6 The accuracy degree with respect to marker 10 for drone landing based on Deep Q Network (DQN).

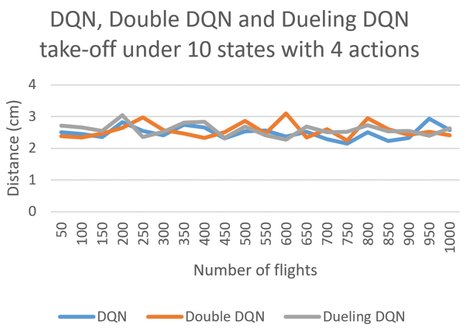

Figure 7 The accuracy degree with respect to marker 5 for drone take-off under 10 states with 4 actions.

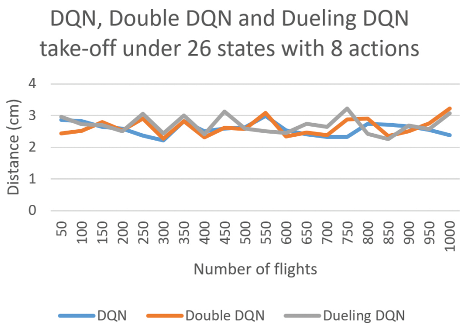

Figure 8 The accuracy degree with respect to marker 5 for drone take-off under 26 states with 8 actions.

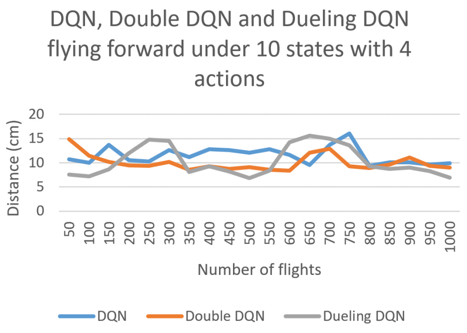

Figure 9 The accuracy degree with respect to marker 0 for drone flying forward under 10 states with 4 actions.

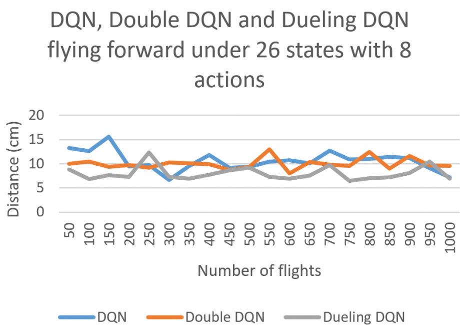

Figure 10 The accuracy degree with respect to marker 0 for drone flying forward under 26 states with 8 actions.

Table 1 The statistics of the flying forward procedure.

	10 states with 4 actions			26 states with 8 actions		
	DQN	Double DQN	Dueling DQN	DQN	Double DQN	Dueling DQN
Mean	11.45873	10.00787	10.30215	10.60394	10.00403	8.024729
Standard deviation	1.787481	1.671201	3.105719	2.047313	1.183294	1.46527

10.30215 cm to 8.024729 cm. On the other hand, upon considering the standard deviation of error values among these experiments, DQN becomes worse under the scenario with more states and actions while Double DQN and Dueling DQN still become better. Particularly, Dueling DQN achieves the best improvement as well.

5. CONCLUSIONS

In this work, we have presented several control strategies to improve the accuracy of some fundamental actions during a drone flies. The goal is to make the drone take-off, fly forward, and land as straight as possible. As far as we know, we are the first to propose DRL methods to this end. Moreover, we have demonstrated the efficacy of our methods by some simulation experiments with different settings. According to the experimental results, the learning process can achieve more accurate results and converge more quickly with more adopted states and actions.

On the other hand, it is obvious that there is no certain learning method that can always enjoy the best efficiency on accuracy and convergence. Hence, for future works, we intend to combine more than one DRL method to integrate their learning results so as to make a better decision for all different scenarios. Moreover, the Asynchronous Advantage Actor Critic (A3C) algorithm is also an interesting topic for our future research. Particularly, we further intend to implement more complicated applications with advanced DRL methods, such as Policy Gradient methods, to make a drone able to fly completely autonomously and more precisely.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

AUTHORS' CONTRIBUTIONS

Che-Cheng Chang: Writing - original draft, Writing - review & editing, Methodology, Software. Jichiang Tsai: Writing - review & editing, Conceptualization, Methodology. Peng-Chen Lu: Software, Data curation. Chuan-An Lai: Software, Data curation.

ACKNOWLEDGMENTS

This research was supported by Ministry of Science and Technology, Taiwan, R.O.C. under grant 108-2218-E-035-009- and 108-2221-E-035-063-.

REFERENCES

[1] S. Dionisio-Ortega, L.O. Rojas-Perez, J. Martinez-Carranza, I. Cruz-Vega, A deep learning approach towards autonomous flight in forest environments, in 2018 International Conference onElectronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 2018, pp. 139–144.

[2] V. Maximov, O. Tabarovsky, Survey of accuracy improvement approaches for tightly coupled ToA/IMU personal indoor navigation system, in Proceedings of International Conference on Indoor Positioning and Indoor Navigation, France, 2013.

[3] T. Sugimoto, M. Gouko, Acquisition of hovering by actual UAV using reinforcement learning, in 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, China, 2016, pp. 148–152.

[4] R.S. Sutton, A.G. Barto, Reinforcement Learning: an Introduction, The MIT Press, Cambridge, MA, USA, 2018.

[5] M.F. Sani, G. Karimian, Automatic navigation and landing of an indoor AR, drone quadrotor using ArUco marker and inertial sensors, in 2017 International Conference on Computer and Drone Applications (IConDA), Kuching, Malaysia, 2017, pp. 102–107.

[6] P. Benavidez, J. Lambert, A. Jaimes, M. Jamshidi, Landing of an ardrone 2.0 quadcopter on a mobile base using fuzzy logic, in 2014 World Automation Congress (WAC), Waikoloa, HI, USA, 2014, pp. 803–812.

[7] L.V. Santana, A.S. Brandao, M. Sarcinelli-Filho, Navigation and cooperative control using the AR drone quadrotor, J. Intell. Robot. Syst. 84 (2016), 327–350.

[8] R. Mishra, A. Javed, ROS based service robot platform, in 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 2018, pp. 55–59.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.-Y. Ng, ROS: an open-source robot operating system, in ICRA Workshop on Open Source Software, Kobe, Japan, 2009.

[10] N. Koenig, A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 2004, pp. 2149–2154.

[11] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, A. Cangelosi, Autonomous quadrotor landing using deep reinforcement learning, arXiv:1709.03339, 2018.

[12] ROS.org, Powering the world's robots [Online], 2020. https://www.ros.org/

[13] S. Ramil, L. Roman, M.-G.A. Edgar, M. Evgeni, ROS-based multiple cameras video streaming for a teleoperation interface of a Crawler Robot, J. Robot. Netw. Artif. Life. 5 (2018), 184–189.

[14] Gazebo [Online], 2020. http://gazebosim.org/

[15] M. Josip, K. Stanko, S. Ivo, P. Vladan, Adaptive fuzzy mediation for multimodal control of mobile robots in navigation-based tasks, Int. J. Comput. Intell. Syst. 12 (2019), 1197–1211.

[16] S. Garrido-Jurado, R. Munoz-Salinas, F.J. Madrid-Cuevas, M.J. Marin-Jimenez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognit. 47 (2014), 2280–2292.

[17] F.J. Romero-Ramirez, R. Munoz-Salinas, R. Medina-Carnicer, Speeded up detection of squared fiducial markers, Image Vision Comput. 76 (2018), 38–47.

[18] M. Sewak, Deep Reinforcement Learning, Springer, Singapore, 2019.