Autonomous Flight Sımulation of Unmanned Aerial Vehicles with Deep-Q-Network

Jitae Yun, Su-Kyung Yoon, and Shin-Dug Kim
Department of Computer Science, Yonsei University, Seoul, Korea
{ jty11, sk.yoon, sdkim}@yonsei.ac.kr

ABSTRACT
This paper proposes an autonomous flight sımulation system for small unmanned aerial vehicles (UAVs). These autonomous UAVs can learn path-finding from the start to the destination without colliding with any obstacles in the simulation environment. A system called Deep-Q-Network (DQN) can learn control policy using deep reinforcement learning model and UAVs can learn the necessary policies to arrive at a specific point. By employing the DQN for autonomous flight simulation, the proposed simulator can eliminate inefficiencies using reinforcement learning and provide high accuracy and fast learning time.

Keywords: Unmanned aerial vehicles; Autonomous flight; Deep Q-network; Simulation.

1. INTRODUCTION

In recent years, as the use of unmanned aerial vehicles (UAVs) has increased in various fields such as for industrial, disaster, leisure, and military purposes, UAVs with autonomous flight capability, which means that the UAVs recognize and judge themselves under unexpected situations, have drawn considerable attention. In particular, path-finding from the start to the end and collision avoidance are critical for autonomous flight systems. For such autonomous flights, Dijkstra's algorithm [8], genetic algorithms [9], among others, have been mainly used; however, recently, techniques that recognize any obstacles and find paths have been suggested using deep learning [2] and reinforcement learning [3]. Espetially, reinforcement learning is to learn optimal policy to maximize the expected sum of future reward. For this, the agent recieves a state from environment and takes an action to maximize the expected cumulative reward. But when using reinforcement learning, it is difficult to apply it directly to visual percept, such as UAV simulation. In this case, since the raw image pixels on the screen are used as state, an enormous number of action-state pairs to explore are existed [7]. In addition, the larger the state space, the worse the problem becomes. Thus, to mitigate the challenges that reinforcement learning faces, this paper proposes an autonomous flight sımulation system based on deep reinforcement learning combined reinforcement learning and deep learning [1]. In this approach, the Deep Q-Network (DQN) which is the first deep reinforcement learning method proposed by Google DeepMind, extracts the feature points from the high-dimensional input pixel data, and then learns optimal policy for any obstacles recognization and path- finding in UAV simulation.

2. METHOD

The proposed system basically used the Q-learning algorithm [4, 10], which is a sort of a reinforcement learning algorithm, for autonomous flight. Figure 1 shows the basic cycle of Q-learing. When the recived state is s_t ($s_t \in S$, S is a set of states) at time t, by performaing action a ($a \in A$, A is a set of actions) the agent transitions to state s_{t+1}, after that, the agent recived a reward r_t. The basic idea of Q-learning is to determine an optimal policy through rewards obtained from the interaction between an agent's action and an environment's state [5]. In autonomous flight simulation using Q-learning, state information denotes the position change of UAVs in the current screen image, and action denotes the direction of movement of UAVs. As shown in Eq 1, the Q-learning algorithm consists of $Q(s_t, a_t)$,

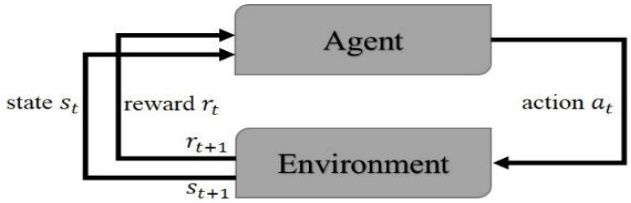

Figure 1. Basic cycle of reinforcement learning

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

the learning rate α, reward r_t, and d discount factor γ. In here, $\max Q(s_{t+1}, a)$ is maximum value of Q for all possible actions in the next state, the agent learns the way to obtain the highest Q-value. However, since Q-learning uses a Q-table to store each state and action's Q-value, if an enormous number of action-state pairs to explore are existed, the table will be very big and it causes computational overhead and requires large memory space. Furthermore, Q-learning faces sequences of highly correlated states [1]. To address this, DQN, which is a combination of a convolution neural network and Q-learning [1], is employed. Through the neural network of DQN, the input data are preprocessed and Q-value actions are selected simply. In the our approach, the feature points from the high-dimensional input pixel data are extracted, and the preprocessed data are used as input state in reinforcement learning.

For autonomous flight sımulation of UAVs with DQN training module, as shown in Figure 3, we implemented a 3D graphic map of an urban environment, where the main obstacles are various buildings and trees of different heights. The DQN training module applied to the UAV simulation consists of three modules as DQN agent, ale wrap and xitari, as shown in Figure 2. Xitari receives score value (distance from drones to destination) for getting reward and state information and controls action in simulator by using med module and screen module. ALE Wrap is a library wrapping the functions in Xitari and eventually DQN agent controls the UAV in Simulator by using ALE Wrap function and trains it to arrive well at the destination. And we design the simulation module interworking with DQN engine. We focused on algorithms for finding rewards so that UAVs[11] on the Simulator can be trained on the optimal path.

The UAVs in the simulation are controlled in the following manner using an action control module: stopped/before/after/left/right/up/down/clockwis/counterclockwis. The screen images are converted into pixel information to be delivered to the DQN training module as state information of UVAs in the state information management module. The action information of UAVs, the scores calculated by the training module, the state information of UAVs converted by the state information management module, and other variables to be shared between the 3D graphic map simulator and the DQN training module are stored and managed on the memory management module. As the DQN training module uses 'experience replay ' [6] for storing necessary information to perform Q-learning such as state transitions, rewards, and actions, additional memory space, which can be shared between the 3D graphic map simulator and the DQN training module, the agent is required. Based on the calculated results by DQN

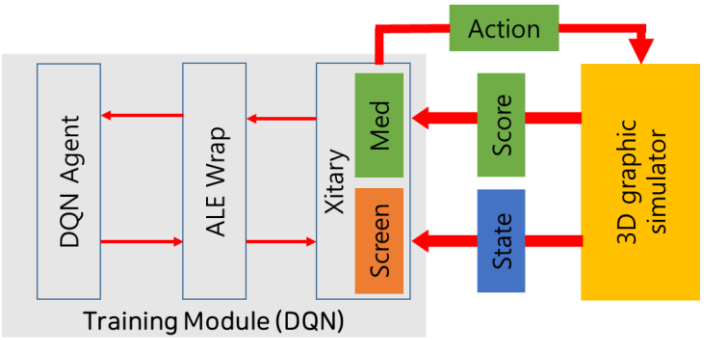

Figure 2. DQN training module and 3D graphic urban map simulator

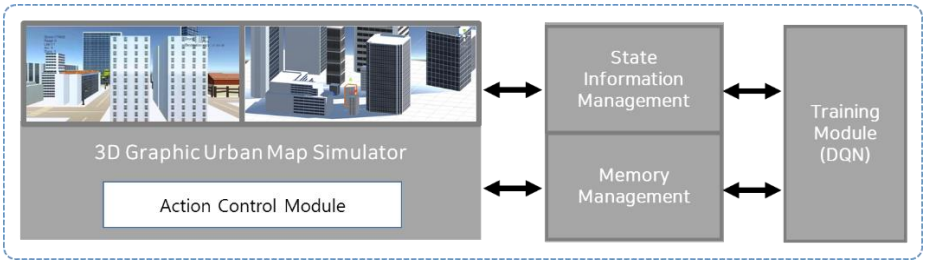

Figure 3. Proposed 3D graphic urban map simulator for UAVs

Table 1. Reward table

Reward / Status	Closer	Far-away	Collision	Destination
	1	-2	-20	+20

traning module, the action of the agent is determined. After the action, if the agent gets close to the destination, positive reward are gotten. Conversely, if an agent moves away from the destination or hits any obstacles, it gets a negative reward. Table1 shows reward algorithm we designed.

3. Results

The proposed 3D graphic urban map simulator is implemented using Unity 3D. Training through autonomous flight simulation is performed using a computer with Intel® Core™ i5-7400, 8-GB RAM, and Geforce GTX1070 GPU on the Ubuntu 14.04. The Linux Xwindow API and the MED (Memory Editor) were used for the connection between the 3D graphic urban map simulator and the DQN training module. The 3D graphic urban map simulator screen is recognized by Xwindow API, and currently displayed sreen image is input into DQN training module. And feature points are extracted through preprocessing including the neural network. Finally, those are used as the state information of Q-learning. The simulation is conducted repeatly. If the agent collides with any obstacles before it reaches the destination, it restarts at the starting point and the iteration is treated as fail.

Figure 4 shows the successe rates, which means the UAV reached the destination without any collision during a period of time. We obtained the success rate by measuring the number of success counts without collision during 2500 seconds. The x-axis shows the success rate, and the y-axis shows time interval. Figure 4 shows that as a time goes by, the number of times to reach the destination increases during a period of time. It means that as learning progresses, the agent learns the route to reach the destination without hitting the obstacle by using the learning information accumulated in real time. Figure 5 shows the number of failure rates after traning progress. As shown in Figure 4, the success rate does not increase significantly after learning has progressed sufficiently. We measured the failure counts during a period time. As shown in Figure 5, after training, the proposed UAVs simulator system's failure is 5~7% , on average.

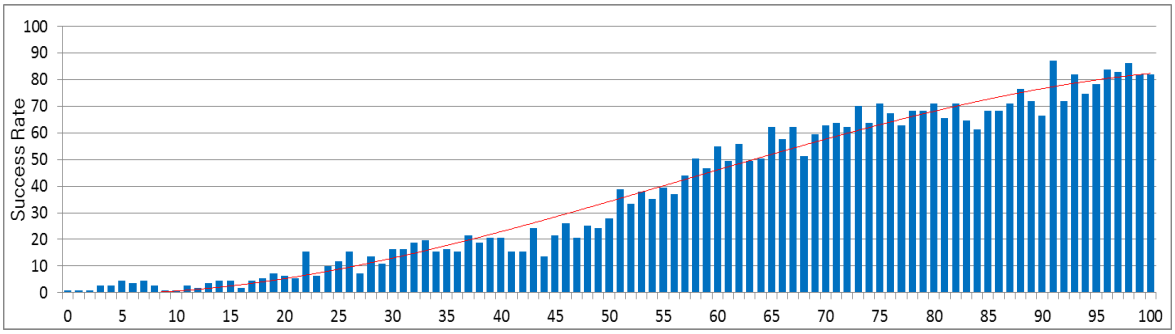

Figure 4. Success rates reaching the target point without collision

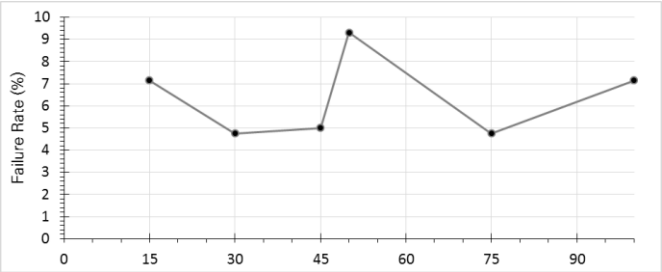

Figure 5. Failure rate after training

4. Conclusion

This paper proposes autonomous flight sımulation of UAVs with DQN. In this system, DQN is employed to eliminate facing sequences of highly correlated states and computational overhead. A 3D graphic urban map simulator is also designed for path-finding and collision avoidance. Furthermore, for integrating DQN and the 3D graphic urban map simulator, a state information management module and a memory management module are implemented. The simulation results demonstrate the proposed simulator can find the path to the destinnation through reinforcement learning and and provide high accuracy.

REFERENCES

1. Mnih, Volodymyr et.al. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 02 2015.
2. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436.
3. Sutton, Richard S., and Andrew G. Barto. Introduction to reinforcement learning. Vol. 135. Cambridge: MIT press, 1998.
4. Duff, Michael O. "Q-learning for bandit problems." Machine Learning Proceedings 1995. 1995. 209-217.
5. Zhang, Baochang, et al. "Geometric reinforcement learning for path planning of UAVs." Journal of Intelligent & Robotic Systems 77.2 (2015): 391-409.
6. Zhang, Shangtong, and Richard S. Sutton. "A Deeper Look at Experience Replay." arXiv preprint arXiv:1712.01275 (2017).
7. Ernst, Damien, Raphaël Marée, and Louis Wehenkel. "Reinforcement learning with raw image pixels as input state." Advances in Machine Vision, Image Processing, and Pattern Analysis. Springer, Berlin, Heidelberg, 2006. 446-454.
8. Johnson, Donald B. "A note on Dijkstra's shortest path algorithm." Journal of the ACM (JACM) 20.3 (1973): 385-388.
9. Hu, Yanrong, and Simon X. Yang. "A knowledge based genetic algorithm for path planning of a mobile robot." Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 5. IEEE, 2004.
10. Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8.3-4 (1992): 279-292.
11. Hongda Chen, Kuochu Chang, C.S. Agate, UAV Path Planning with Tangent-plus-Lyapunov Vector Field Guidance and Obstacle Avoidance, IEEE Transactions on Aerospace and Electronic Systems, Vol. 49, Issue2, pp. 840-856, 2013.