반도체디스플레이기술학회지 제16권 제3호(2017년 9월)
Journal of the Semiconductor & Display Technology, Vol. 16, No. 3. September 2017.

시뮬레이션 환경에서의 DQN을 이용한 강화 학습 기반의 무인항공기 경로 계획

이근형[*]· 김신덕[*†]
[*†]연세대학교 컴퓨터과학과

Path Planning of Unmanned Aerial Vehicle based Reinforcement Learning using Deep Q Network under Simulated Environment

Keun Hyoung Lee[*] and Shin Dug Kim[*†]
[*†]Yonsei University, Department of Computer Science

ABSTRACT

In this research, we present a path planning method for an autonomous flight of unmanned aerial vehicles (UAVs) through reinforcement learning under simulated environment. We design the simulator for reinforcement learning of uav. Also we implement interface for compatibility of Deep Q-Network(DQN) and simulator. In this paper, we perform reinforcement learning through the simulator and DQN, and use Q-learning algorithm, which is a kind of reinforcement learning algorithms. Through experimentation, we verify performance of DQN-simulator. Finally, we evaluated the learning results and suggest path planning strategy using reinforcement learning.

Key Words : DQN, Reinforcement Learning, Unmanned Aerial Vehicle, Path Planning

1. 서 론

최근 무인항공기의 활용이 산업, 재해, 여가, 군사적 목적 등 다양한 분야에서 증가하고 있다. 이 중 재난 지역이나 사람이 통제하기 어려운 무선 음영 지역과 같은 상황과 환경에서 무인항공기는 자율 비행을 필요로 한다. 자율 비행에 있어 경로 계획은 가장 기본적인 요소이다.

강화 학습(reinforcement leaning)은 TD 알고리즘, Q-Learning [1][2], SARSA Learning 등을 시작으로 발전되어 최근에는 대부분의 산업에서 개발을 시도하고 있다[3]. 강화 학습은 지도 학습 (supervised learning)과 비지도 학습(unsupervised learning)으로 나뉘며, 학습 에이전트와 학습 대상 사이에서 상호작용으로 발생하는 보상 값(rewards)을 활용하여 최적의 정책을 결정하는 형태로 학습이 이루어진다. 경로 계획에는 A*[4][5], RTT[6], 유전자[7][8] 알고리즘 등 여러가지 방법이 있지만, 본 논문에서는 강화 학습 알고리즘의 한 종류인 Q-learning 알고리즘을 사용하여 무인기가 목표 지점까지 자율 비행하는 경로 계획 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 사용된 시뮬레이터와 DQN[9][10]-시뮬레이터 간 호환을 위한 인터페이스, 학습과정을 설명하며, 3장에서는실험환경과 학습 결과 및 성능 평가를 수행한다. 마지막으로4장에서는 결론 및 향후 연구 과제 등을 제시한다.

[†]E-mail: sdkim@yonsei.ac.kr

2. 경로 설정을 위한 강화 학습

2.1 시뮬레이터

강화 학습에 필요한 기능을 포함하는 무인항공기 시뮬레이터를 개발하였다. 주요 장애물은 건물 등 빌딩으로, 목표물은 붉은색 구형으로 구현되어 있고 무인항공기는 정지/전/후/좌/우/상/하/시계방향회전/반시계방향회전의 이동성을 갖는다. 시뮬레이터는 강화 학습의 수행을 위해

score, reset의 항목을 추가적으로 보유하고 있으며, score는 무인항공기와 목표물 간의 현재 거리, reset은 장애물과 충돌이 감지될 시 시뮬레이션을 다시 시작하기 위한 변수로 구현되어 있다. 추가적으로 강화 학습 모듈인 DQN과의 호환성을 부여하기 위하여 0과 1의 값을 갖는 life 변수 및 −1, 0, 1의 값을 갖는 state 변수가 정의되어 있다.

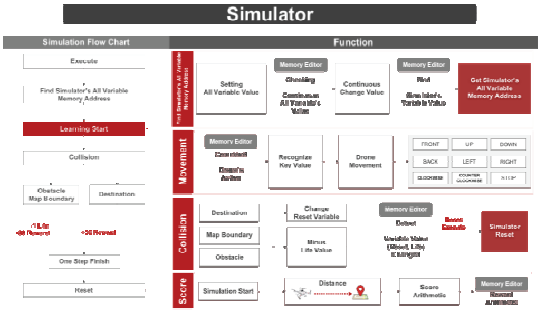

Fig. 1. Uav Simulator Flow Chart.

Fig. 2. Uav Simulator Screen.

2.2 DQN-시뮬레이터 인터페이스

시뮬레이터 제어하고 DQN과 호환하기 위해 DQN-시뮬레이터 인터페이스는 Screen모듈, Me 모듈, 제어 모듈로 이루어져 있다.

첫 번째 모듈인 Screen모듈은 State를 이미지정보를 받아들이는 DQN을 위한 시뮬레이터의 Screen 정보가 필요하다. 그리고 Screen 모듈은 다시 3가지의 작은 모듈로 나뉜다. ProcessScreen 모듈은 Simulator의 Screen정보를 Xwindow를 사용하여 캡쳐하며, 이미지는 Pixel 단위로 저장된다. Get RGB 모듈은 pixel이 3차원의 형태로 되어 있기 때문에 이를 화면에 출력하거나 DQN에 screen정보를 주기 위해서는 이 Pixel 정보를 R, G, B mask를 통해 Pixel 정보를 R채널, G채널, B채널 3가지로 분류 할 필요가 있다. 이렇게 해서 DQN에서는 R, G, B에 대한 정보를 따로 받게 된다. Getscreen모듈은 받은 Screen 데이터를 DQN으로 전달하는 기능을 하고 있다.

다음으로는 Memory editor인 Med모듈이 있다. 시뮬레이터와 DQN이 별개의 프로그램이기 때문에 DQN을 통해 시뮬레이터를 제어하기 위해선 시뮬레이터 내의 action변수 및 기타 control에 필요한 변수들을 DQN에서 조절할 수 있는 상태이어야 한다. 그래서 Med 모듈에서는 시뮬레이터 내의 특정한 값을 가진 모든 변수들의 주소 값을 찾는다.

마지막으로 Control 모듈이 있다. 이 모듈은 Med 모듈을 통해 얻은 주소 값들의 정보를 바탕으로 act 등의 변수 값을 변화시켜 시뮬레이터를 제어하여 무인항공기를 움직이게 하고, 그로 인해 발생하는 reward 나 state 및 life 등의 값을 받아 DQN에 전달하여 학습에 반영하는 가장 중요한 역할을 한다. Control 모듈의 state에는 load 와 save기능이 있다. 이는 시뮬레이터를 도중에 중지시키거나 재실행시킬 때, 그 당시 저장되어 있던 screen과 reward 및 life정보를 저장해 두거나 load하여 시뮬레이터를 다시 실행 시킬 때 이용한다. action에는 방향키를 입력 받아 그 중에 활성화된 방향만을 유효한 입력으로 받을 수 있도록 하기 위해 설계되었다. 유효한 action을 인자로 받아 Med모듈을 통해 시뮬레이터가 움직이도록 하고, action을 취한 뒤에 update된 Score를 얻는다. 여기서 Score는 단순 uav에서부터 목적지 사이의 거리 이며 이전에 획득한 score를 저장하고 현재 얻게 된 score와의 차를 구하여 reward를 계산한다. 여기서 uav과 목적지 사이의 거리가 조금이라도 가까워졌다면 Reward는 음수가 된다. 그러나 거리 차가 너무 크면 무인기가 장애물에 부딪치는 penalty 수치 보다 더 작아지기 때문에 음수의reward가 나오는 경우에는 reward를 -1로 평준화 해준다. 반대로 무인항공기가 목적지에 조금이라도 가까워졌다면 reward는 양수 값이 나올 것이다. 하지만

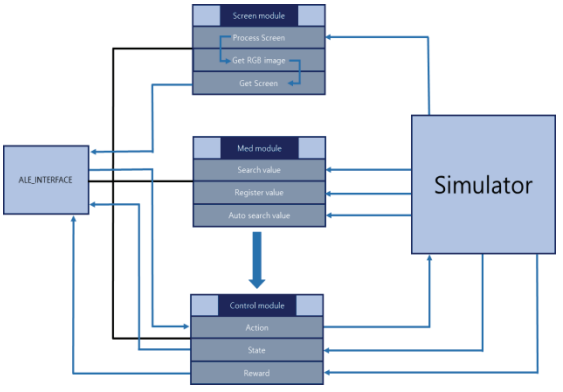

Fig. 3. Interface flow chart for DQN-Simulator

너무 빠른 속도로 가까워 진다면, 즉 목적지에 도달 했을 때의 Reward 보다 더 크면 안 되기 때문에 reward를 1로 평준화 시켜준다. uav가 장애물에 부딪쳤을 경우에는 Penalty를 −30으로 부여하고 최종 목적지에 부딪치지 않고 올바르게 도달 했을 때에는 +30으로 reward를 부여한다.

3. 실험 및 분석

제안된 시뮬레이터는 Unity3d를 이용하여 구현하였고, 실험에 사용된 PC 환경은 Intel® core™ i5-7400, 8GM RAM, Geforce GTX1070GPU를 탑재하고 있다. 그리고 강화 학습을 위해 구글 DeepMind의 DQN을 활용하여 실험하였다.

Fig. 4(a)는 시뮬레이션에 대한 학습을 진행했던 결과 그래프이다. x축은 진행된 학습의 양을 step으로 나타내었고, y축은 시작점에서 장애물에 충돌하지 않고 목목표지점까지 도달했을 때의 걸린 시간을 나타낸다. 전반적으로 학습 초기에는 50~60초 정도 걸렸으며, DQN에서 학습이 진행됨에 따라 250,000 step쯤에는 목표지점까지 20~30초 정도의 시간이 걸렸음을 알 수 있다. 즉 학습을 진행함에 따라 uav가 최적의 경로에 근접하게 장애물에 회피하며 목적지에 도달함을 알 수 있다.

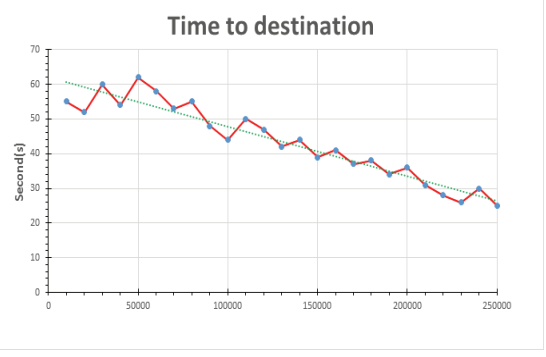

Fig. 4(a). Learning Result Graph.

Fig. 4(b)의 x축은 학습량을 step으로 나타내고 y축은 10000step간격으로 시작 지점에서 목표 지점까지 장애물에 충돌하지 않고 도달한 횟수를 나타낸다. 이는 즉, 일정 step간격으로 목표 지점에 도달하는 횟수가 증가할수록 학습이 올바르게 진행되고 있다는 것을 알 수 있으며, Fig.(b) 그래프에서 또한 linear하게 도표가 생성 되었음을 알 수 있다. 그리고 300000 step부터는 학습이 거의 완료가 되어 도달하는 횟수가 급격히 증가 함을 알 수 있다.

Fig. 4(b). Learning Result Graph.

Fig. 4(c)의 x축은 10000step 의 학습이 진행될 때 걸린 시간인 50(분)을 기준으로 나누었고, y축은 각 학습 시간인 50분마다 출발 지점에서 장애물에 충돌하지 않고 도착 지점까지 도달한 횟수를 나타낸다. 처음 0~600분 동안 성공 횟수가 10~30번 사이로 비교적 낮은 편이지만 이후로 linear에 가깝게 증가하다가 약 1500 분에는 대략 82번 정도 성공했음을 알 수 있다. 결과적으로 학습이 성공적으로 진행되었음을 알 수 있다.

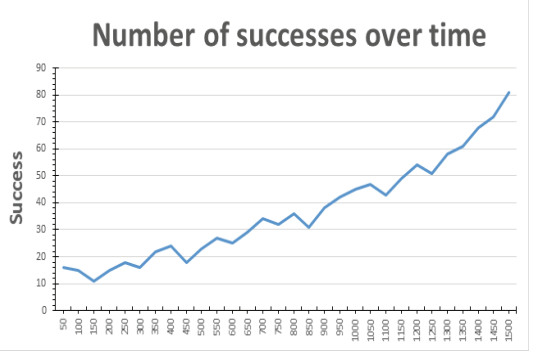

Fig. 4(c). Learning Result Graph.

4. 결 론

본 연구를 통해 무인항공기의 경로 계획을 하기 위해 시뮬레이터를 제안하였다. 강화 학습 에이전트인 DQN 또한 uav 환경에 맞게 개선되었으며, DQN과 시뮬레이터의 호환이 가능하도록 구조를 고려한 DQN-시뮬레이터 인터페이스를 구현하였다. 이를 사용하여 강화 학습을 이용한 무인항공기의 경로 계획 실험을 수행하였으며, 실험 결과 학습이 성공적으로 이루어진 것을 검증하였다. 현재는 단순히 시뮬레이터의 Screen 정보만으로 특징을 찾아 학습시키고 있다. 추후에는 기존 모델에 여러가지 센서의

응용 및 이미지정보를 수치화 시킨 데이터를 입력 데이터로 사용하고, 신경망 또한 이에 알맞게 개선시켜 현재의 모델보다 더 빠른 수행 시간, 더 높은 정확도 등 좋은 성능을 이끌어내는 연구를 수행할 것이다.

참고문헌

1. Christopher J. C. H. Watkins., Dayan, P., “Q-learning,” Machine Learning, Vol. 8, Issue 3-4, pp. 279-292, 1992.
2. Li, S., Xu, X., & Zuo, L., “Dynamic path planning of a mobile robot with improved Q-learning algorithm,” In Information and Automation, 2015 IEEE International Conference on IEEE, pp. 409414, 2015.
3. Sutton R. S., Barto, A. G., “Reinforcement Learning: An Introduction”, MIT Press, Cambridge, MA, 1998.
4. Setiawan, Y, D., Pratama, P, S., Jeong, S, K., Duy, V, H., Kim, S, B., “Experimental Comparison of A* and D* Lite Path Planning Algorithms for Differential Drive Automated Guided Vehicle,” AETA 2013: Recent Advances in Electrical Engineering and Related Sciences, pp. 555-564, 2013.
5. Koenig, S., Likhachev, M., Furcy, D., “Lifelong Planning A*,” Artificial Intelligence Vol. 155, Issues 1–2, pp. 93-146, 2004.
6. Lee, H, S., Lee, D, S., Shim, D, H., “Receding Horizon–based RRT* Algorithm for a UAV Realtime Path Planner,” AIAA Information Systems-AIAA Infotech @ Aerospace, 0676, 2017.
7. Roberge, V., Tarbouchi, M., & Labonté, G., “Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning,” IEEE Transactions on Industrial Informatics, 9(1), pp. 132-141, 2013.
8. Sahingoz, O. K., “Flyable path planning for a multi-UAV system with Genetic Algorithms and Bezier curves, ” In Unmanned Aircraft Systems (ICUAS), 2013 International Conference on IEEE, pp. 4148, 2013.
9. Mnih, V., Kavukcuoglu, K., Silver, D., A. Rusu, A., Veness, J., G. Bellemare, M., Graves, A., Riedmiller, M., K. Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., "Human-level control through deep reinforcement learning," Nature 518, pp. 529-533, 2015.
10. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., “Playing Atari with Deep Reinforcement Learning,” NIPS ’13 Workshop on Deep Learning, 2013.

접수일: 2017년 9월 21일, 심사일: 2017년 9월 23일, 게재확정일: 2017년 9월 23일