Deep Reinforcement Learning for Drone Delivery in Unity

김진우

지도교수 : 유선철 교수님

목 차

1. Introduction

전 세계적으로 배달, 배송 등 물류 산업에서 로봇 활용의 중요성이 점차 증가하고 있다. 대표적인 온라인 쇼핑 회사인 아마존은 KIVA(키바) 로봇을 사용하여 운영 비용을 20% 절감하였다.[1] 한국에서도 배달의 민족, ROBOTIS 등 로봇 관련 기업에서 배달 로봇, 서빙 로봇을 개발하여 시범 서비스를 제공하고 있다.[2] 또한, 최근에는 코로나로 인해 비대면 서비스를 원하는 고객이 많아지면서 드론 배달 관련 산업군이 확장하고 있다. 그 예로, 도미노피자에서는 2021년에 국내 최초로 드론 배달 서비스 '도미 에어'를 개발하여 드론 배달 서비스 상용화 실시한다고 한다.[3]

그림 1. 드론 택배 관련 통계 자료

드론이 물건을 안전하게 배달하려면 장애물의 위치를 인식하고 회피하는 경로 계획이 필수적이다. 본 연구에서는 Unity Drone Challenge에서 제공하는 시뮬레이터 환경을 사용하여 드론의 경로를 계획하는 강화학습 알고리즘을 구현하였다.

---

1 정원영, '아마존, 키바 로봇 도입으로 운영 비용 20% 절감', 2016.06.16.

(http://www.irobotnews.com/news/articleView.html?idxno=7844)

2 안준호, '이제야… 배달 로봇도 보도 통행 허용', 2019.12.19.

(https://biz.chosun.com/site/data/html_dir/2019/12/19/2019121900067.html)

3 백민정, '"따끈따끈 도미노피자 날아온다" 드론 배달 딱 7분, 주문 폭주', 2021.09.19.

(https://www.joongang.co.kr/article/25008388)

Unity Drone Challenge에서 제공하는 환경은 다음과 같다. 가운데 물류 창고에서 시작하여, Drone이 배송할 집들이 10개 있으며 한 episode마다 10개의 집 중 3개가 임의로 지정된다. 그리고 건물, 나무, 차량 등 정적 장애물들과 동적 장애물인 새가 있다.

그림 2. Unity Drone Delivery 환경

Unity에서 제공하는 강화학습 라이브러리, ML-Agents를 활용하여 continuous action space에 대한 SAC 알고리즘을 구현하였고, python으로 JNN, PPO 알고리즘을 작성하였다. 그리고 알고리즘을 Unity Drone Delivery 환경에 적용하여 드론이 더 빠르게, 많은 배송지로 배송하게 하는 경로 계획을 구현하였다.[4]

[4] https://github.com/kjwoo31/Unity_drone

2. Related Works

1) Reinforcement Learning in Drone

기존 연구에서는 AirSim 환경에서 드론의 obstacle avoidance, delivery를 강화학습으로 구현하였다.

Kersandt, Muñoz, Barrado (2018)[5]는 DQN, DDQN, Dueling DDQN으로 여러 개의 기둥이 있는 환경에서 drone의 obstacle avoidance를 달성하였다. Depth image, 드론과 목적지의 위치를 나타낸 map image를 합치고 그 이미지를 CNN에 통과시켜 얻어진 feature를 state로 사용하였다. 세 개의 알고리즘 중 DDQN이 성공률 80%로 가장 좋은 결과를 얻었다.

후속 연구로, Muñoz, Barrado, Çetin, Salami (2019)[6]는 AirSim의 Neighborhood 환경에서 DDQN, Joint Neral Network (JNN)을 적용하여 Drone Delivery를 구현하였다. 드론과 목적지의 위치를 map image로 나타내는 기존 방식 대신, depth image만 CNN으로 feature를 추출하고 위치 정보는 scalar로 처리하는 JNN으로 정확도 80%에서 100%로 성능을 높였다고 한다. 하지만 아직 학습 시간이 오래 걸린다는 한계점이 존재하여, 본 연구에서는 이 문제를 해결하기 위해 observation의 space size를 줄이고 off-policy인 DDQN 대신 on-policy인 PPO를 활용하였다.

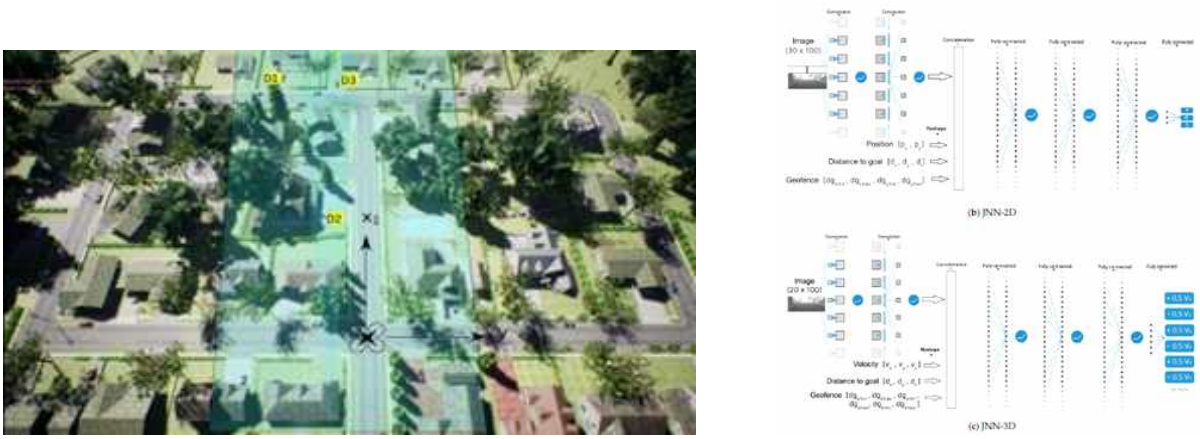

그림 3. AirSim Neighborhood 환경과 JNN 구조

[5] Kersandt, K., Muñoz, G., & Barrado, C. (2018, September). Self-training by reinforcement learning for full-autonomous drones of the future. *In 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)* (pp. 1-10). IEEE.

[6] Muñoz, G., Barrado, C., Çetin, E., & Salami, E. (2019). Deep reinforcement learning for drone delivery. *Drones*, *3*(3), 72.

2) Reinforcement Algorithm - SAC vs PPO

Baseline 알고리즘으로 ML-Agents에서 제공하는 오픈 소스 알고리즘인 Soft Actor-Critic (SAC)과 Proximal Policy Optimization (PPO) 중 하나를 선택해야 했다. 기존 연구를 참고하여 continuous action space에 적합한 SAC를 선택하였다.

Schulman, Wolski, Dhariwal, Radford, Klimov (2017)[7]가 제안한 PPO는 Policy parameter θ_{old}를 여러 번 재사용하며 갱신하는 방식이다. 대부분의 on-policy 알고리즘은 학습할 때마다 policy parameter가 변화하기 때문에 한 번 사용한 policy parameter를 재활용하기 어렵다. 하지만 PPO는 Clipping과 GAE로 policy parameter θ_{old}가 학습되는 θ와 유사한 값을 유지하며, 기존 데이터를 재사용할 수 있다. 따라서, 적은 sample로도 효율적으로 학습을 진행한다. 또한, PPO는 기존 State-of-the-art 알고리즘인 Trust Region Policy Optimization (TRPO)의 업데이트 방식을 근사하여 gradient descent만 사용하는 알고리즘이다. 구현이 쉬우며 TRPO에 가까운 성능을 보여 실용적인 강화학습 알고리즘으로 많이 쓰이고 있다.

Haarnoja, Zhou, Abbeel, Levine (2018)[8]가 제안한 SAC는 기존 Actor-Critic에 Entropy를 추가하여 확률적인 action을 출력할 수 있게 만든 알고리즘이다. SAC는 Continuous action space에서 기존 알고리즘들보다 빠른 학습 속도와 높은 성능을 보인다고 한다. 특히 환경이 복잡할수록, 큰 batch를 사용하는 on-policy 방법보다 SAC가 빠른 초기 학습 속도를 보여주었다. 이와 관련하여, Haamoja et al. (2018)[9]에서는 HalfCheetah, Walker2d 등 OpenAI Gym의 continuous locomotion tasks를 PPO보다 SAC이 잘 수행한다는 결과가 얻어졌다.

---

[7] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[8] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (pp. 1861-1870). PMLR.

[9] Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., & Levine, S. (2018). Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*.

3. Implementation

1) Baseline

가. State, Action, Reward

State로 Drone의 위치, 배달 위치 좌표, 속도, 각속도가 주어지고 Lidar와 Camera에서 얻어지는 sensor data가 주어진다. Action은 x, y, z 방향 -1~1 사이 연속적인 값을 갖는다. Reward는 물품 배송이 완료되면 100점을 추가하는 Event reward와 이전 Step에서 목표지점과 거리에서 현재 Step에서 목표지점과 거리를 뺀 Distance reward를 주게 하였다.

나. Hyperparameter

Unity ML-Agents에서 제공하는 예제 중 'Food Collector'를 학습할 때 활용한 hyperparameter를 참고하여 작성하였다. 'Food Collector'는 파란색 큐브가 위험 물체인 빨간색 큐브를 피하면서 목표물인 초록 큐브에 부딪혀야 하는 task로 장애물을 피하며 목적지로 향하는 drone delivery와 유사하다.

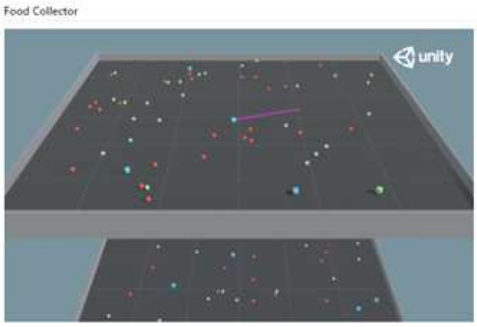

그림 4. Unity Food Collector Task

또한, 학습 결과를 확인하며 환경에 적합하게 hyperparameter를 변경하였다. Continuous action space이기 때문에 batch size, buffer size와 network의 hidden unit, layer number를 높게 설정하였다. 그리고 drone, lidar, camera에서 얻어지는 데이터가 서로 연관성이 적어 전체적으로 normalize 하여 network에 통과시켰다. 마지막으로, 보상이 드물게 발생하여 Time horizon을 길게 설정하였다. 구체적인 hyperparameter 수치는 github 링크[10]에서 확인할 수 있다.

[10] https://github.com/kjwoo31/Unity_drone/blob/main/drone_config/sac.yaml

2) JNN, PPO

Baseline에서는 택배 배달 위치 좌표 3곳을 모두 표시하였는데, 3곳 중 1곳의 좌표만 표시하게 바꾸었다. 먼저 학습 속도가 빨라지고 효율적 Cost가 가장 적은 경로를 찾고, 그 경로에 따라 다음 목적지를 설정하게 하였다. (Cost는 거리를 기준으로 하였다.) 그 외 데이터는 JNN은 Baseline에서 사용한 모든 정보 (drone, lidar, camera), PPO는 드론 정보와 수평 방향, 위아래 Raycast만 활용한다. Baseline에서 continuous action을 사용한 것과 달리, JNN과 PPO에서는 [0,0,0], [1, 0, 0], [0, 1, 0], …의 7가지 action만 사용하였다. 또한, reward는 lidar 정보를 바탕으로 장애물을 회피하면 보상을 주는 Avoidance reward를 추가하여 장애물에 부딪히지 않게 하였다. 마지막으로, 빠르게 학습하기 위해 최소한의 network size, time horizon으로 학습시킨 후 학습 결과에 따라 학습이 잘 이루어지지 않으면 점점 증가시켰다.

4. Results

1) Baseline

observation, action의 크기가 커서 물품 1개를 배송하는 local minima에 갇혀 더 배송하지 못했다.[11]

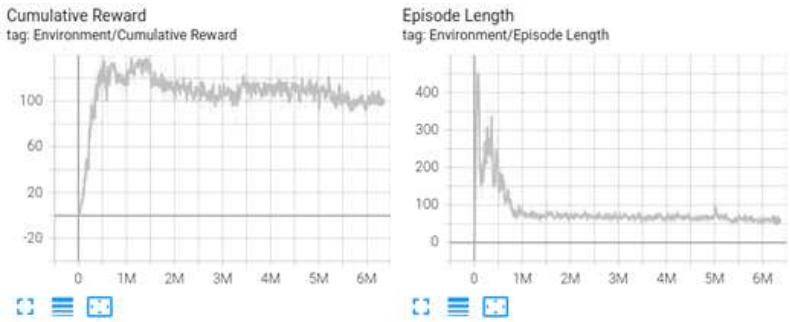

그림 5. Baseline 학습 결과 (X축: episode, Y축: reward, length)

2) Joint Neural Network (JNN)

Off-policy이며, 이미지 처리 시간이 오래 걸려 학습에 걸리는 시간이 너무 오래 걸린다. 설계 동안 계속하여 학습하였지만 한 번도 배달에 성공하지 못했다.[12]

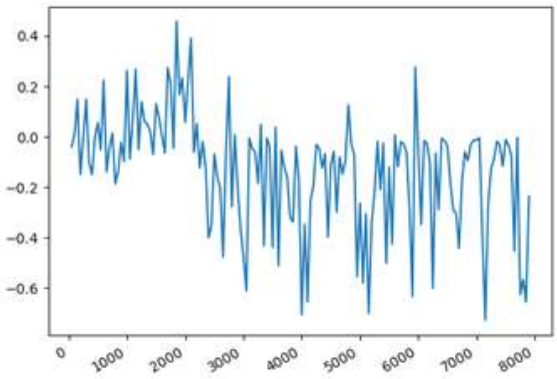

그림 6. JNN 학습 결과 (X축: episode, Y축: reward)

[11] https://youtu.be/heHW-mvRGf8

[12] https://youtu.be/AWz4s-csdDI

3) Proximal Policy Optimization (PPO)

On-policy이며, data Efficient하고 state/action space가 작아서 빠르고 효과적으로 학습하였다. Baseline에서 실패하였던 2개 이상의 물품 배송에도 성공하였고 JNN보다 빠르게 학습하였다.[13] Unity Drone Challenge에 제출하였을 때, 10번의 episode에서 23번의 배송에 성공하였다. (23/30)

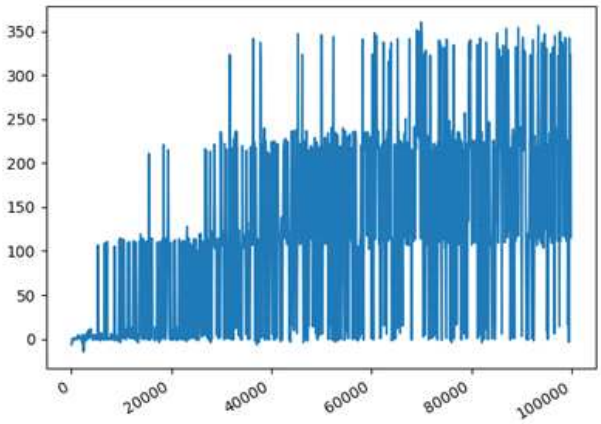

그림 7. PPO 학습 결과 (X축: episode, Y축: reward)

5. Conclusion

Unity 환경에서 학습한 결과, 적은 state와 action으로 학습한 PPO가 빠른 학습 속도, 높은 배달 성공률을 보였다. PPO를 사용한 학습 방법은 적은 episode로 학습할 수 있어 실제 드론에도 실용적으로 사용할 수 있을 것이라 기대된다.

Unity Drone Challenge의 우승자 역시 PPO 알고리즘을 사용하였다고 한다.[14] PPO의 batch size, network size를 크게 하고, time horizon을 높게, 오랫동안 학습하면 좀 더 안정적인 배송이 가능할 것이다. 또한, 드론의 움직임 제어, 안정적인 착지가 가능하면 실제 드론에도 적용할 수 있을 것이다.

[13] https://youtu.be/7VYC500BR44

[14] https://ramanuzann.notion.site/DDC-RLKorea-Unity-f4b9551314a84f5cbc4d7e316c3164dc