한국시뮬레이션학회 논문지
Vol. 27, No. 4, pp. 1-8 (2018. 12)
http://doi.org/10.9709/JKSS.2018.27.4.001
ISSN 1225-5904

가상 환경에서의 강화학습을 이용한 비행궤적 시뮬레이션

이재훈† · 김태림 · 송종규 · 임현재

Flight Trajectory Simulation via Reinforcement Learning in Virtual Environment

Jae-Hoon Lee† · Tae-Rim Kim · Jong-Gyu Song · Hyun-Jae Im

ABSTRACT

The most common way to control a target point using artificial intelligence is through reinforcement learning. However, it had to process complicated calculations that were difficult to implement in order to process reinforcement learning. In this paper, the enhanced Proximal Policy Optimization (PPO) algorithm was used to simulate finding the planned flight trajectory to reach the target point in the virtual environment. In this paper, we simulated how this problem was used to find the planned flight trajectory to reach the target point in the virtual environment using the enhanced Proximal Policy Optimization(PPO) algorithm. In addition, variables such as changes in trajectory, effects of rewards, and external winds are added to determine the zero conditions of external environmental factors on flight trajectory learning, and the effects on trajectory learning performance and learning speed are compared. From this result, the simulation results have shown that the agent can find the optimal trajectory in spite of changes in the various external environments, which will be applicable to the actual vehicle.

Key words : Machine Learning, Reinforcement Learning, Flight Trajectory, Proximal Policy Optimization

요 약

인공지능을 이용하여 목표 지점까지 제어하는 가장 대표적인 방법은 강화학습이다. 하지만 그동안 강화학습을 처리하기 위해서는 구현하기 어렵고 복잡한 연산을 처리해야만 했다. 본 논문에서는 이를 개선한 Proximal Policy Optimization (PPO) 알고리즘을 이용하여 가상환경에서 목표지점에 도달하기 위한 계획된 비행궤적을 찾는 방법을 시뮬레이션 하였다. 또한 외부 환경요소가 비행궤적 학습에 미치는 영향을 알아보기 위하여 궤적의 변화, 보상 값의 영향 및 외부 바람등과 같은 변수를 추가하고 궤적 학습 성능 및 학습 속도에 미치는 영향을 비교 분석을 수행한다. 본 결과를 통하여 에이전트가 다양한 외부환경의 변화에도 계획된 궤적을 찾을 수 있다는 것을 시뮬레이션 결과에 따라 알 수 있었으며, 이는 실제 비행체에 적용할 수 있을 것이다.

주요어 : 머신러닝, 강화학습, 비행궤적, Proximal Policy Optimization

1. 서론

최근 인공지능(Artificail intelligence)은 4차 산업혁명을 주도하는 기술로서 tractica의 2017년 발표에 따르면 인공지능의 세계 시장이 2016년 약10억달러 규모에서 2025년에는 약 600억 달러 규모로 성장할 것으로 예측했다. 이런 인공지능의 성장 배경인 인공지능 기술 중 머신러닝(Machine Learning)은 학습 방식에 따라 크게 지도학습, 비지도 학습 그리고 강화학습으로 나눌 수 있다. 지도학습과 비지도 학습의 경우 주로 어떠한 물체를 구분하거나 데이터의 특성을 분석, 가공하는 데 사용되는 반면 강화학습은 주로 제어나 상호작용을 통해 최적의 동작을 학습하는 분야에서 사용된다(김성필, 2016).

Received: 15 June 2018, **Revised**: 14 September 2018, **Accepted**: 27 September 2018

† **Corresponding Author**: Jae Hoon Lee
E-mail: jaehoon.lee@lignex1.com
LIG Nex1

강화학습은 비디오 게임을 플레이하거나 알파고처럼 프로선수를 바둑에서 이기는 등 다양한 분야에서 활용되고 있는데 이중 무인 비행기(드론)를 제어하는 연구도 상당히 많이 이루어지고 있다. 드론을 제어하는데 인공지능이 사용되는 가장 큰 이유는 그 사용 환경에 있다. 드론은 주로 산불감시, 표적 추적, 인명구조 같은 미지의 환경에서 활용되는 경우가 많다. 강화학습은 스스로 탐색하면서 주변환경에 유연하게 대응할 수 있게 해주는 좋은 방법이다(Huy X 등, 2018).

본 연구의 목적은 강화학습을 활용하여 이용하여 비행체를 알고리즘 없이 계획된 궤적을 통과해 목표지점에 도달하도록 하는 것이다. 이를 위하여 각각의 다른 환경변수를 포함한 시나리오를 5개를 포함한 시뮬레이션 환경을 만들었다. 시나리오는 각각 아무 변수가 없는 기본 궤적, 궤적의 중간에서 각도를 꺾은 궤적, 곡선 궤적중 직선을 추가한 궤적, 위 궤적의 직선 부분에 벗어나면 감점요소를 추가한 궤적, 마지막으로 매번 다른 방향으로 바람이 부는 궤적이다. 각각의 시나리오에서 에이전트는 목적지에 도달하는 것을 목표로 3DOF 방향으로 자유롭게 이동할 수 있게 하였다. 시나리오를 통해 나온 결과를 궤적의 변화, 보상 값의 영향, 바람의 영향으로 나누어서 비교 분석하였다.

2. 관련 연구

본 장에서는 본 연구에 사용된 알고리즘에 대하여 소개하고 시뮬레이션을 통한 강화학습의 사례를 분석한다.

2.1 시뮬레이션을 통한 강화학습 사례

시뮬레이션을 사용한 강화학습의 사례로 드론의 제어에 많이 사용되었다. 드론의 궤적 제어를 위한 연구로는 Jemin Hwangbo(2017)등은 deterministic onpolicy method를 이용한 강화학습을 통하여 Quadrotor를 궤적을 따라 이동하는 시뮬레이션을 수행하여 DDPG, TRPO -GAE 보다 학습속도가 빠르고 실험에서는 더 안정적으로 구동되었다. Huy X. Pham(2018)등은 PID+ Q-learning 알고리즘을 사용하여서 임의의 환경에서 무인항공기의 항법장치를 시뮬레이션 하여 최단 스텝으로 목적지에 도착하도록 훈련하였고, 성공적으로 비행을 학습하여 실제 무인항공기에 적용하여 동일한 결과를 얻었다. 드론의 자세를 제어하는 방법과 비행경로를 결정하여 드론의 움직임을 제어하는 부분으로 나누어진다. 드론의 자세 제어를 위한 연구로는 William Koch(2018)는 GYM FC를 사용하여 PPO, TRPO, DDPG, PID 알고리즘별로 무인비행기의 고도를 제어하는 시뮬레이션을 수행하였고, 그 결과로 PPO 알고리즘이 튜닝된 PID 제어기보다 더 높은 결과를 낼 뿐만 아니라, 학습된 에이전트(agent)는 재교육 없이도 연속적인 작업을 수행하는 것을 확인하였다. 본 논문에서는 높은 성능을 가진 PPO알고리즘을 에이전트에 적용하여 시뮬레이션을 수행하였다.

2.2 Proximal Policy Optimization(PPO) Algorithms

PPO 알고리즘은 John Schulman(2015)등이 발표한 Trust Region Policy Optimization (TRPO)를 그 뿌리로 두고 있다. TRPO는 정책 성능 제한(Policy Performance Bound) 즉 정책 업데이트(Policy update)에 제한을 두어 이전 정책과 현 정책의 차이를 KL-Divergence로 구한 다음 새로운 KL-Divergence의 크기를 δ이하로 제한하여서 결과가 급격히 변하더라도 정책까지 급하게 변하지 않게 하는 기법이다. 이를 통해 강화학습에서 학습속도를 높일 뿐만 아니라 정확성도 크게 향상 되었다. 하지만 TRPO에는 2차 근사 방법(second order gradient)이 존재하고 이는 연산이 복잡할 뿐만 아니라 구현이 어려워지는 문제를 가진다. 이러한 문제를 해결하기 위해 John Schulman(2017)등이 발표한 PPO 알고리즘에서는 정책의 변화 비율을 고정(clip)하여 제약을 두어 비율보다 높거나 낮아지는 것을 막는다. 이 방법을 통해 TRPO보다 더 빠르고, 성능적으로 우수할 뿐만 아니라 구현이 쉬운 장점들을 가지고 있다.

3. 시뮬레이션 방법

본 장에서는 본 연구에 사용된 도구 및 사용된 용어에 대하여 간략히 설명한 후 시뮬레이션 학습 주기 및 시나리오에 대하여 소개한다.

3.1 도구 및 용어

본 연구에 사용된 프로그램은 Vincent Pierre(2017)이 Unity Machine Learning Agents(ML-Agents)이며 이는 Unity에 들어가는 오픈소스 플러그인(Open-source Plug-in) 이다. ML-Agent는 Unity 엔진과 에디터가 구현하는 섬세한 물리효과뿐만 아니라 PPO 알고리즘의 구현을 포함하고 있어서 시뮬레이션 환경 및 에이전트를 설계하면 자체적으로 강화학습을 수행하며 모델을 생성해 준다.

PPO 알고리즘을 수행하기 위하여 사용된 하이퍼 파라메터(hyperparameter)의 목록, 값 및 각 값의 범위는

Table 1. Hyperparameters of Unity ML-agent

Hyper parameter	Value	Description	Typical Range
Gamma	0.995	corresponds to the discount factor for future rewards	0.8 - 0.995
Lambda	0.95	corresponds to the lambda parameter used when calculating the Generalized Advantage Estimate (GAE)	0.9 - 0.95
Buffer Size	5120	corresponds to how many experiences should be collected before we do any learning or updating of the model	2048 - 409600
Batch Size	512	the number of experiences used for one iteration of a gradient descent update	512 - 5120
Number of Epochs	3	the number of passes through the experience buffer during gradient descent	3-10
Learning Rate	0.0003	corresponds to the strength of each gradient descent update step	1e-5 - 1e-3
Time Horizon	64	corresponds to how many steps of experience to collect per-agent before adding it to the experience buffer	32 - 2048
Max Steps	1.0e5	corresponds to how many steps of the simulation are run during the training process	5e5 - 1e7
Beta	1e-3	corresponds to the strength of the entropy regularization, which makes the policy “more random.: This ensures that agents properly explore the action space during training	1e-4 - 1e-2
Epsilon	0.1	corresponds to the acceptable threshold of divergence between the old and new policies during gradient descent updating	0.1 - 0.3
Number of Layers	1	corresponds to how many hidden layers are present after the observation input, or after the CNN encoding of the visual observation	1 - 3
Hidden Units	128	correspond to how many units are in each fully connected layer of the neural network	32 - 512

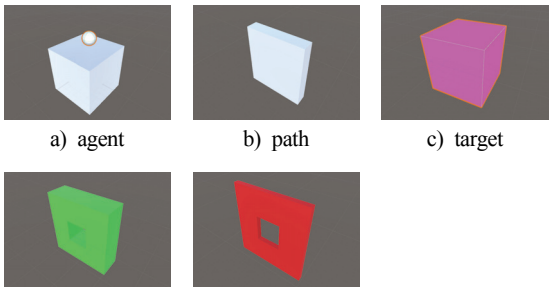


Fig. 1. configuration Objects

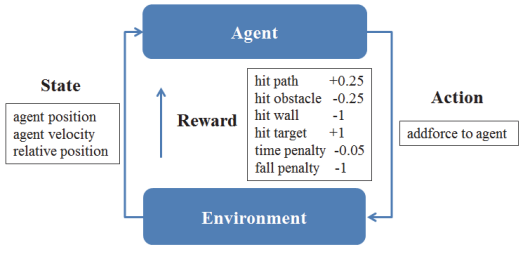


Fig. 2. Learning cycle

Table 1과 같다. 각각의 하이퍼 파라메터는 시뮬레이션 환경의 복잡도에 따라서 설정하게 된다.

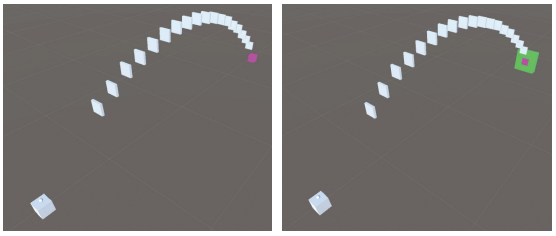
- Gamma: 보상값에 민감하도록 값을 높게설정
- Lambda: 보상값을 추정값보다 신뢰하도록 값을 높게 설정
- Buffer Size, Batch Size, Number of Epoch, Number of Layers:: 시뮬레이션의 복잡도를 고려하여 값을 낮게 설정
- Learning Rate, Max Steps: 실험을 통해 설정
- Time Horizen, Hidden Units: Buffer Size등의 크기를 고려하여 설정
- Beta: 엔트로피(Entropy)의 변화를 고려하여 설정
- Epsilon: 에이전트의 이동폭을 고려하여 낮게 설정

3.2 시뮬레이션 학습 주기

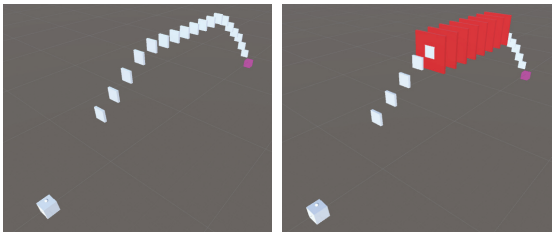
이제 시뮬레이션에 사용된 구성 개체 및 학습 주기에 대하여 설명한다. Fig. 1는 본 시뮬레이션에 사용된 구성 개체들로서 각각 에이전트(agent), 궤적(path), 표적(target),

장벽(wall), 장애물(obstacle)이 있다.Fig. 2는 본시뮬레이션에 사용된 학습 주기(Learning cycle) 로서 학습에 사용된 각각의 요소는 다음과 같다.

- Action(행동)은 3차원 공간으로 작용하는 힘으로, 방향은 3차원 공간의 모든 방향으로 랜덤하게 적용되고, 크기는 중력가속도의 2배를 적용하였다.
- State(상태)는 에이전트의 위치, 속도 그리고 표적과의 거리를 계산하여 에이전트에 전송한다.
- Reward(보상)은 궤적을 통과할 때 +0.25, 표적에 도착하면 +1를 얻는다. 반대로 장애물을 통과할 때 -0.25, 장벽에 맞거나 표적보다 아래로 떨어지는 경우는 -1의 페널티를 준다. 추가로 에피소드의 길이가 증가하는 것을 방지하기 위하여 시간지연에 따른 -0.05의 페널티를 추가하였다.



a) default trajectory b) tilted trajectory



c) parallel trajectory d) parallel trajectory advanced

Fig. 3. simulation trajectories

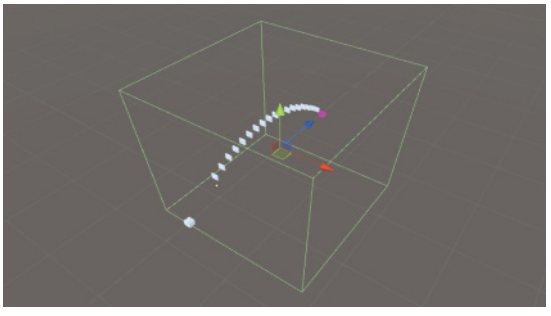


Fig. 4. wind applied trajectory

3.3 시뮬레이션 시나리오

본 시뮬레이션의 시나리오의 목적은 에이전트가 표적으로 계획된 궤적의 변화에 따른 학습 성과 및 장애요소가 학습에 미치는 영향을 알아보기 위함이며, 이를 위하여 Fig. 3, Fig. 4와 같은 5가지 시나리오를 준비하였다.

- default trajectory(궤적 1: Fig 3. a)): 모든 시나리오의 기준 궤적이다. 시나리오는 1개의 에이전트, 18개의 궤적, 1개의 표적으로 구성되어 있으며, 궤적 및 표적의 위치는 시작점에서 37°로 공기저항이 없을 때 표적까지의 경로로 설정하였다.
- qtilted trajectory(궤적 2: Fig 3. b)): 궤적의 변화에 대응하여 방향을 전환할 수 있는가를 확인하기 위해 설정된 궤적이다. 기본궤적의 중간지점부터 15° x축으로 꺾어지는 궤적으로 구성하였다. 궤적이 꺾어졌기 때문에 표적의 끝부분에 걸쳐서 표적을 통과하는 것을 막기 위하여 표적을 감싸는 장벽을 추가하였다.
- parallel trajectory(궤적 3: Fig 3. c)): 중력 및 초기 가속도가 존재하는 상황에서 지면과 수평인 궤적을 유지하는지 확인하기 위해 설정된 궤적이다. 기본궤적에서 중간부분을 곡선에서 지면에 수평이 되도록 구성하였다.
- parallel trajectory advanced(궤적 4: Fig 3. d)): 궤적 3에서 수평 구간에서 곡선으로 걸쳐서 통과하는 것을 막기 위하여 평행 유지 구간에 장애물을 추가하였다.
- wind applied trajectory(궤적 5: Fig 4.): 임의의 방향과 힘으로 바람이 부는 환경에서 궤적을 유지하는지 알아보기 위해 설정된 궤적이다. 실험이 시작될 때 마다 매번 방향 및 크기가 바뀌는 wind area를 추가하였다.

4. 시뮬레이션 결과

본 장에서는 결과 분석, 궤적 변화에 따른 분석과 보상값의 영향분석으로 나누어 시뮬레이션 결과에 대해 분석한다.

4.1 결과 분석

Fig. 5는 4가지 시나리오에 대한 결과를 나타내며 Table 2는 결과에 사용된 통계자료에 대한 설명이다.

- cumulative_reward(보상 합), value_estimate(추정값): 성공적인 훈련과정동안 증가하는 경향을 보임

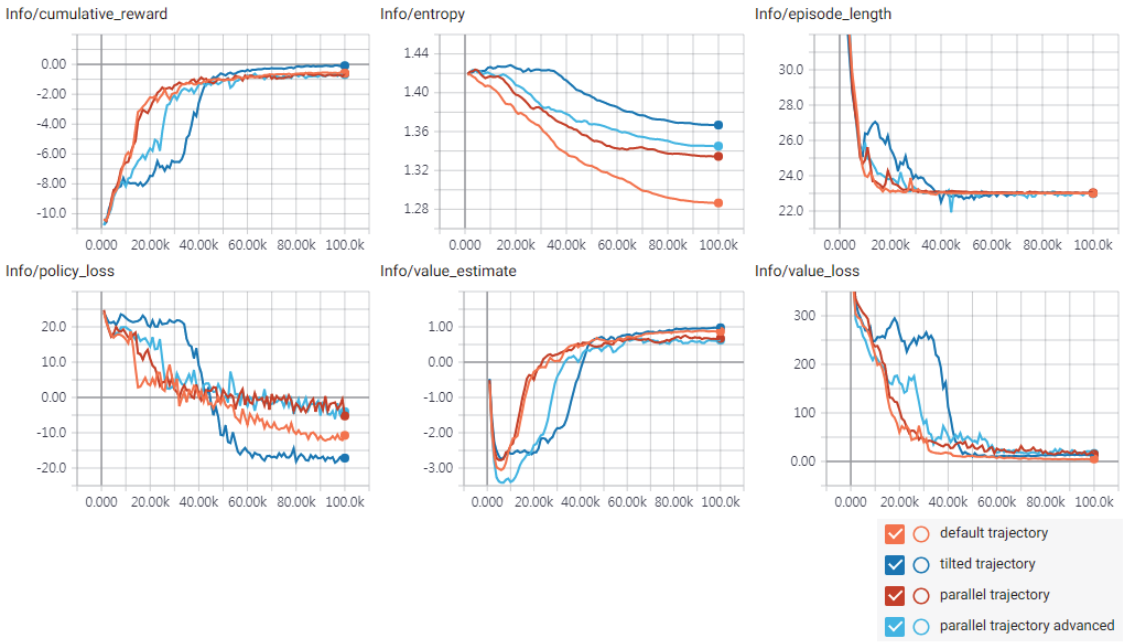


Fig. 5. Test Result Graph

- entropy(엔트로피), policy_loss(정책 손실), value - loss(손실 값): 성공적인 훈련과정동안 감소하는 경향을 보임
- episode_length(에피소드 길이): 에피소드에 걸린 시간은 시간 페널티를 최소화 할수 있도록 감소함

Table 2. Training Statistics

Training Statistic	Content
Cumulative Reward	The mean cumulative episode reward over all agents
Entropy	How random the decisions of the model are. Should slowly decrease during a successful training process
Episode Length	The mean length of each episode in the environment for all agents.
Policy Loss	The mean loss of the policy function update
Value Estimate	The mean value estimate for all states visited by the agent
Value Loss	The mean loss of the value function update

Table 3. Simulation result

trajectory	step	cumulative reward	episode length	policy loss	value loss	time
default trajectory	1k	-10.04	35.97	24.10	361.39	0s
	25k	-2.02	23.07	2.40	44.24	8m55s
	50k	-1.07	23.02	0.05	11.86	20m8s
	75k	-0.64	23.00	-8.69	5.97	32m9s
	100k	-0.58	23.00	-10.74	5.14	44m0s
tilted trajectory	1k	-10.74	36.51	24.02	393.59	0s
	25k	-6.80	24.64	20.74	245.23	7m10s
	50k	-0.66	22.77	-10.77	10.06	17m16s
	75k	-0.30	23.00	-15.81	11.52	28m41s
	100k	-0.09	23.00	-17.17	13.94	40m27s
parallel trajectory	1k	-10.36	35.69	24.78	372.48	0s
	25k	-1.54	23.19	4.33	63.72	8m16s
	50k	-1.04	23.09	2.16	34.95	19m42s
	75k	-0.91	23.09	0.33	28.83	31m9s
	100k	-0.60	23.03	-5.22	14.97	42m37s
parallel trajectory advanced	1k	-10.79	36.73	23.86	378.91	0s
	25k	-4.48	23.69	10.76	167.46	7m11s
	50k	-1.85	23.00	-0.29	41.05	16m4s
	75k	-0.76	23.05	-2.61	19.27	24m36s
	100k	-0.67	22.98	-4.03	16.56	33m4s

모든 궤적의 1k의 값은 거의 동일한 값으로 시작된다. 하지만 에피소드가 증가하면서 서로 다른 패턴을 보여준다. 각 분석의 기준이 되는 궤적인 궤적 1과 궤적 3은 학습과정에서 일정하게 학습이 되며 계획된 궤적을 찾는 모습을 보여준다. 반면 대조군인 궤적 2, 궤적 4의 경우 초중반 동안 변화된 궤적과 장애물에 의해 학습이 정체되는 구간이 있지만 점차 계획된 궤적을 찾는 결과를 보여준다. 시나리오 종료 후 작성된 모델을 새 시나리오에 적용하였을 때 기존 학습된 결과대로 에이전트가 움직이는 것 또한 확인할 수 있었다.

Table 4. Wind Applied Simulation result

trajectory	step	cumulative reward	episode length	policy loss	value loss	time
wind applied trajectory	1k	-10.49	35.89	24.20	373.40	0s
	25k	-3.78	23.68	9.08	158.70	2h11m
	50k	-2.58	23.35	5.01	84.95	4h30m
	75k	-2.53	23.21	6.93	78.96	6h48m
	100k	-2.23	23.24	3.83	68.84	9h08m

4.2 궤적 변화에 따른 분석

궤적 1과 궤적2를 비교하여 궤적의 변화가 시뮬레이션에 미치는 영향을 분석한다. 궤적 1에서 에이전트는 궤적을 통과하기 위해 y축과 z축으로만 힘을 작용하게 된다. 하지만 궤적 2 에서는 궤적의 중간지점 부터 x축으로 꺾이게 되면서 3축으로 힘을 작용해야 목표된 궤적을 통과하게 된다. Fig. 5의 에피소드 0부터 7k구간은 궤적의 중간지점까지의 학습 상태를 보여준다. 해당 기간 동안 보상 합과 에피소드 길이가 거의 일치하는 것을 통해 같은 궤적을 이동하였다는 것을 알 수 있다. 이후 7k부터 33k 구간에서는 궤적 1은 이전 구간과 같이 학습이 원활하게 이루어지는데 반해 궤적 2는 7k 이후 학습상태는 정체 혹은 오히려 증가하는 모습을 보여준다. 이 결과를 통해 에이전트가 이 구간에서 모든 방향으로 이동하면서 보상 합이 증가하는 방향에 대한 데이터를 누적한다. 33k 이후 구간에서는 학습한 궤적을 반복하면서 궤적 1과 유사하게 시뮬레이션을 종료됐다.

4.3 보상 값의 영향 분석

궤적 3과 궤적 4를 통해 보상 값의 영향을 분석한다. 궤적 4에서 장애물 통과 시 -0.25의 페널티가 주어지기 때문에 에이전트가 학습 정체 구간을 갖게 된다. Fig. 5의 에피소드 0부터 7k 구간의 통계 값을 통해 거의 동일한 궤적으로 통과한다는 것을 알 수 있다. 7k부터 24k 구간에서 학습이 정체를 확인할 수 있다. 궤적에서 조금이라도 벗어나게 되면 궤적에서 얻는 보상 값만큼 페널티가 추가되게 되기 때문에 정확한 궤적을 학습하는데 더 많은 에피소드가 필요하게 된다. 에피소드 길이는 7k까지

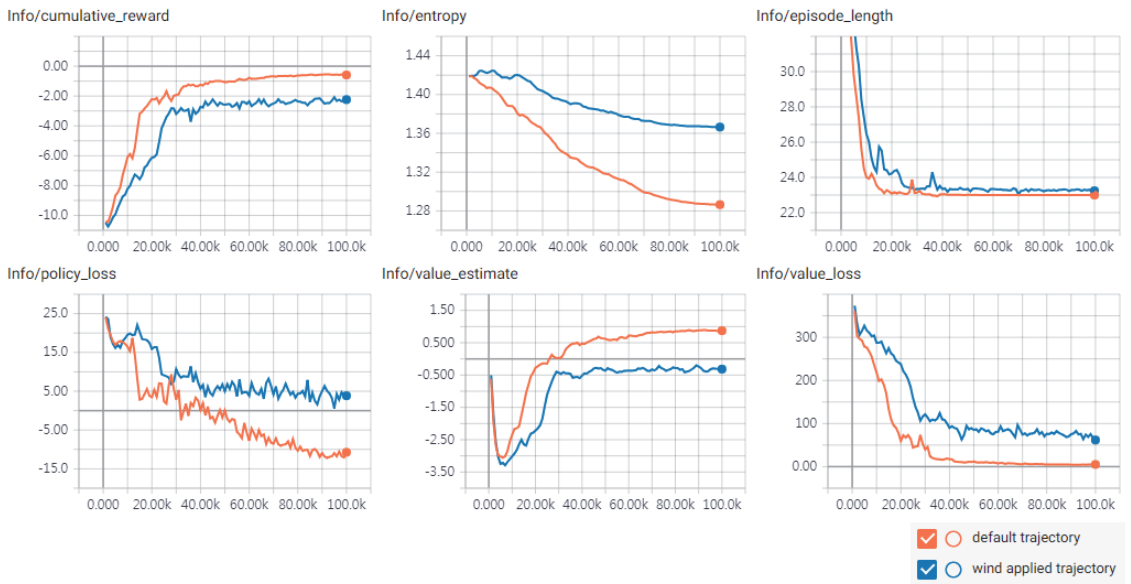


Fig. 6. Wind Applied Test Result Graph

걸린 시간은 1m 48s, 1m 47s으로 거의 동일한 시간이 걸렸지만 이후 24k에 이르러서는 8m 16s, 7m 11s로 장애물이 있는 궤적이 1분여 빨라졌다. 해당 구간의 에피소드 길이는 거의 동일하기 때문에 에이전트가 장애물을 통과하지 않기 위해 궤적의 변화를 최소화 했다는 것을 알 수 있다. 이러한 정책이 누적되어 이후 50k에서는 19m 42s, 16m 4s, 75k에서는 31m 9s, 24m 36s, 100k에서는 42m 37s, 33m 4s로 에피소드에 걸린 시간 차이는 점차 늘어나게 된다.

4.4 바람의 영향 분석

궤적 1과 궤적 5를 통해 바람의 영향을 분석한다. Fig. 4의 큐브형태의 공간에 바람을 가정하여 임의의 방향으로 임의의 힘을 작용하도록 하였다. Fig. 6의 전 구간에서 궤적 5가 상대적으로 학습 속도가 늦어지는 것을 확인할 수 있다. 하지만 보상 합 및 에피소드 길이의 최종 값의 차이를 비교하여 볼 때 에이전트는 거의 동일한 궤적을 이동한다. 이는 매번 다른 방향 다른 크기로 바람이 부는 상황에서도 계획된 궤적을 찾아 이동할 수 있다는 뜻이다. 또한 기존의 시나리오에 비해 가장 큰 차이는 시뮬레이션 수행 시간이다. 궤적 1의 경우 에피소드 1k당 약 20초 이내에 수행되었지만 궤적 5의 경우 1k당 약 6분 정도 시간이 소모되었다. 이는 에피소드 진행 도중 에이전트가 무작위 바람에 대응한 연산에 많은 시간이 소모되기 때문이다.

5. 결론

본 연구에서는 강화학습을 이용하여 비행궤적을 시뮬레이션하고 궤적의 변화, 보상값의 변화, 바람등의 변수가 에이전트의 학습에 미치는 영향에 대하여 분석하였다. 본 연구의 결과를 통하여 변수가 에이전트의 학습 속도에 영향을 주지만 결국 동일한 궤적을 습득 한다는 사실을 알 수 있었다. 시뮬레이션을 통해 생성된 모델을 새 가상환경에 적용하였을 때 에이전트는 최종 학습된 정책과 동일한 결과를 낸다. 즉 비행체와 동일한 형상 및 기동특성을 가진 에이전트로 시뮬레이션을 수행하고, 이를 통해 생성된 모델을 CPU를 통해 FPGA에 적용하거나 직접 텐서플로우를 FPGA에 적용시켜 실제 비행궤적에 적용의 될 수 있을 것이다.

본 연구의 확장으로 향후의 연구 주제들을 다음과 같이 제시한다. 첫째, 실제 비행체와 운용조건, 환경조건 등을 좀 더 구체화한 시뮬레이션 환경을 구축하여 시뮬레이션 하는 것이다. 시뮬레이션의 결과는 학습된 모델로 남게 되어 향후 이전 과정을 거치지 않고도 시뮬레이션 종료 시 학습결과를 불러올 수 있다. 따라서 학습된 모델을 비행체의 항법장치에 적용하게 된다면 비행체가 실제 비행을 통하여 강화학습의 결과를 축적하지 않고도 비행에 적용될 수 있을 것이다. 둘째, 학습된 모델을 다른 시나리오에 적용하는 커리큘럼 학습(Curriculum Learning) 개념을 적용하는 것이다. 바람을 적용한 시나리오에서처럼 연산에 시간이 걸리게 되면 학습속도 자체를 늦추게 된다. 이를 보완하기 위해 기본 시나리오를 학습한 모델을 복잡한 시나리오에 적용하여 초반부터 학습 속도 및 효율을 증가시키는 것이 가능 할 것이다. 셋째, 표적에 일정범위 내로 다가가 에이전트의 카메라에 목적지가 포착되었을 경우 독립적인 사물탐지(object detect)로 전환하는 것이다. 사전 학습된 목적지를 실시간으로 추적하여 표적이 움직이더라도 그에 대응하여 표적에 도달할 수 있을 것이다.

References

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, Pieter Abbeel (2015), *Trust Region Policy Optimization,* arXiv:1502.05477v5 [cs.LG].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov (2017), *Proximal Policy Optimization Algorithms,* arXiv:1707.06347v2 [cs.LG] 28 Aug 2017.

Vincent Pierre (2017), *Unity ML-Agents*, https://github.com/Unity-Technologies/ml-agents

Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick (2017), *Mask R-CNN*, arXiv:1703.06870 [cs.CV]

Jemin Hwangbo, Inkyu Sa, Roland Siegwart, Marco Hutter (2017), *Control of a Quadrotor with Reinforcement Learning,* arXiv:1707.05110v1 [cs.RO]

Huy X. Pham, Hung. M. La, David Feil-Seifer, Luan V. Nguyen (2018), *Autonomous UAV Navigation Using Reinforcement Learning,* arXiv:1801.05086v1 [cs.RO]

William Koch, Renato Mancuso, Richard West, Azer Bestavros, *Reinforcement Learning for UAV Attitude Control*, arXiv:1804.04154v1 [cs.RO]

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre,

G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *Mastering the game of go with deep neural networks and tree search,* Nature, vol. 529, no. 7587, pp. 484–489, 2016

Huy X. Pham, Hung. M. La, David Feil-Seifer, Luan V. Nguyen, *Autonomous UAV Navigation Using Reinforcement Learning,* arXiv:1801.05086v1 [cs.RO]

김성필 (2016), 딥러닝 첫걸음, 한빛미디어, 서울, pp. 17-33.

이 재 훈 (jaehoon.lee@lignex1.com)

2009 한동대학교 기계제어 공학과 학사
2011 한동대학교 기계제어 공학과 석사
2011~ 현재 LIG Nex1 유도무기 연구소 선임연구원d

관심분야 : 딥 러닝, 강화학습

김 태 림 (taerim.kim@lignex1.com)

2009 서강대 전자공학과 학사
2011 한국과학기술원 전기 및 전자공학과 석사
2011~ 현재 LIG Nex1 유도무기 연구소 선임연구원

관심분야 : 임베디드 시스템, 시스템 엔지니어링

송 종 규 (jonggyu.song@lignex1.com)

2011 한양대 전자통신공학과 학사
2013~ 현재 LIG Nex1 유도무기 연구소 선임연구원

관심분야 : 디지털 회로 설계, VHDL, 임베디드 SW

임 현 재 (hyunjae.im@lignex1.com)

2013 인하대 정보통신공학과 학사
2013~ 현재 LIG Nex1 유도무기 연구소 선임연구원

관심분야 : 실시간 시스템, 임베디드 소프트웨어, 운영체제, 유도무기