한국정보전자통신기술학회논문지(jkiiect)'22-4, Vol.15 No.2
논문 22-15-02-152 http://dx.doi.org/10.17661/jkiiect.2022.15.2.152

가상환경과 DDPG 알고리즘을 이용한 자율 비행체의 소노부이 최적 배치 연구

김종인*, 한민석**

Research on Optimal Deployment of Sonobuoy for Autonomous Aerial Vehicles Using Virtual Environment and DDPG Algorithm

Jong-In Kim*, Min-Seok Han**

요 약 본 논문에서는 대잠전의 필수 요소인 소노부이를 무인항공기가 최적의 배치로 투하할 수 있게 하는 방법을 제시한다. 이를 위해 Unity 게임엔진을 통해 음향 탐지 성능 분포도를 모사한 환경을 구성하고 Unity ML-Agents를 활용해 직접 구성한 환경과 외부에서 Python으로 작성한 강화학습 알고리즘이 서로 통신을 주고받으며 학습할 수 있게 하였다. 특히, 잘못된 행동이 누적되어 학습에 영향을 미치는 경우를 방지하고 비행체가 목표지점으로 최단 시간에 비행함과 동시에 소노부이가 최대 탐지 영역을 확보하기 위해 강화학습을 도입하고. 심층 확정적 정책 그래디언트(Deep Deterministic Policy Gradient: DDPG) 알고리즘을 적용하여 소노부이의 최적 배치를 달성하였다. 학습 결과 에이전트가 해역을 비행하며 70개의 타겟 후보들 중 최적 배치를 달성하기 위한 지점들만을 통과하였고 탐지 영역을 확보한 모습을 보면 겹치는 영역 없이 최단 거리에 있는 지점을 따라 비행하였음을 알 수 있다. 이는 최적 배치의 요건인 최단 시간, 최대 탐지 영역으로 소노부이를 배치하는 자율 비행체를 구현하였음을 의미한다.

Abstract In this paper, we present a method to enable an unmanned aerial vehicle to drop the sonobuoy, an essential element of anti-submarine warfare, in an optimal deployment. To this end, an environment simulating the distribution of sound detection performance was configured through the Unity game engine, and the environment directly configured using Unity ML-Agents and the reinforcement learning algorithm written in Python from the outside communicated with each other and learned. In particular, reinforcement learning is introduced to prevent the accumulation of wrong actions and affect learning, and to secure the maximum detection area for the sonobuoy while the vehicle flies to the target point in the shortest time. The optimal placement of the sonobuoy was achieved by applying the Deep Deterministic Policy Gradient (DDPG) algorithm. As a result of the learning, the agent flew through the sea area and passed only the points to achieve the optimal placement among the 70 target candidates. This means that an autonomous aerial vehicle that deploys a sonobuoy in the shortest time and maximum detection area, which is the requirement for optimal placement, has been implemented.

Key Words : Autonomous Arial Vehicle, DDPG Algorithm, Reinforcement Learning, Sonobuoy, Unity ML-Agents, Virtual Environment

This Paper was supported by Research Fund of Republic of Korea Naval Academy in 2022.
*Department of Electronic and Control Engineering, Republic of Korea Naval Academy
**Corresponding Author : Department of Electronic and Control Engineering, Republic of Korea Naval Academy (mshan1024@navy.ac.kr)
Received March 30, 2022 Revised April 15, 2022 Accepted April 23, 2022

1. 서론

소노부이(sonobuoy)는 유·무인 항공기 등을 활용한 감시·정찰 자산 중 한 가지 형태로 다양한 유·무인 항공기 등의 플랫폼을 통해 일정 수량을 해양에 투하하여 잠수함 등 대상표적을 탐지해 대잠전(anti-submarine warfare)을 효과적으로 수행하기 위한 음향탐지용 부표이다. 항공기에서 탐지하고자 하는 해역에 다수의 소노부이를 투하하는 방식으로 여러 지역을 동시에 감시하는 대잠작전을 수행할 수 있게 한다[1].

강화학습의 방식에는 크게 학습의 보상으로 Q 함수를 학습하는 가치기반 방식과 행동 정책을 학습하는 정책기반 방식이 있다. 고전적인 가치기반의 학습 알고리즘인 TD 방식[3]은 각 에피소드 간의 분산은 작으나, 편향이 높은 문제가 있고, 정책기반의 학습에서 쓰이는 Monte-Carlo estimate 알고리즘의 경우에는 각 에피소드 간의 편향은 작으나 분산이 높은 문제가 있다. 이러한 문제점을 최소화하기 위해 가치기반 방식과 정책기반 방식을 결합한 Actor-Critic 알고리즘이 고안되었고, 이때 정책을 더 안정적으로 수렴시키기 위해 목적함수가 일정 수준 이상으로 커지는 것을 제한한 PPO[4] 알고리즘이 제안되었다. 그러나 여전히 연속적이고 방대한 상태를 가지는 자율 비행체의 소노부이 최적 배치 문제에는 수많은 local minima가 존재하여 최적의 보상 값으로 수렴시키는 데에 어려움이 존재한다.

본 논문에서는 강화학습을 통해 자율 비행체가 소노부이를 최적으로 배치하는 경로를 학습하도록 하는 방법으로 Deep Deterministic Policy Gradient(DDPG)[5] 알고리즘을 도입한다. DDPG 알고리즘은 PPO 알고리즘과는 다르게 Off-Policy 방식을 적용함으로써 타겟 정책과는 다른 행동 정책으로부터 샘플 궤적을 구할 수 있으므로 exploration 측면에서 좀 더 유리하여 local minima 문제를 일부 해결해 줄 수 있다. 소노부이를 최적으로 배치한다는 것의 의미는 비행체가 목표지점으로 최단 시간에 비행함과 동시에 소노부이가 최대 탐지 영역을 확보하는 것을 의미한다. 이를 구현하기 위해 Unity ML-agents를 이용하여 소노부이를 배치할 가상환경을 직접 생성하고 보상을 설계한다. 또한, 강화학습의 알고리즘 중 하나인 DDPG 알고리즘을 적용하여 학습을 진행함으로써 적용 가능성을 검토한다.

2. 가상환경 제작

2.1 소노부이 배치 환경

본 논문에서는 대잠전의 필수 요소인 소노부이를 무인 항공기가 최적의 배치로 투하할 수 있게 하는 방법을 제시하기 위해 강화학습(reinforcement learning)을 도입하였다. 강화학습이란 기계학습의 한 영역으로 어떤 환경 내에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법이다[2]. 에이전트(agent)는 환경(environment)으로부터 보상(reward)과 상태(state)를 받아 행동(action)을 결정한다. 에이전트는 행동을 통해 환경에 영향을 주어 다음 상태로 천이(transition)되며 이 과정을 하나의 에피소드(episode)라고 한다. 강화학습 알고리즘을 통해 에피소드를 반복하면 에이전트가 특정 행동을 학습하여 자율적으로 수행할 수 있게 된다.

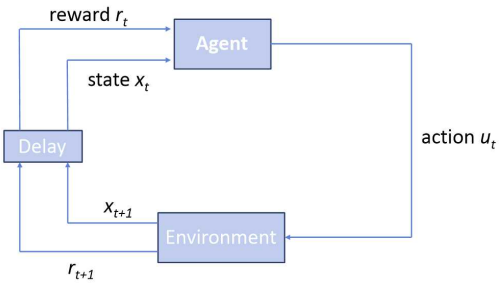

그림 1. 강화학습 모델
Fig. 1. Reinforcement-Learning Model

강화학습을 하기 위해서는 학습을 진행할 환경과 학습을 할 방법인 강화학습 알고리즘이 필요

하다. 기존 방식들의 경우, Unity 게임엔진을 통해 학습에 필요한 가상환경을 비교적 쉽게 구성할 수는 있으나 Unity 내부에서 직접 강화학습 알고리즘을 구성하여 학습하기에는 어려움이 많았다. 이를 해결하기 위해 개발된 툴이 바로 Unity Machine Learning Agents (ML -Agents)다[6]. Unity ML-Agents는 게임엔진으로 직접 구성한 환경과 외부에서 Python으로 작성한 강화학습 알고리즘이 서로 통신을 주고받으며 학습할 수 있게 해주어 기존의 강화학습 알고리즘 적용문제를 해결하였다.

그림2는 Unity ML-Agents를 이용한 통신 방법을 나타낸 것으로 강화학습 알고리즘이 에이전트가 취할 행동을 환경에 전달하여 에이전트가 행동을 하도록 하고 이에 따라 변화된 환경을 바탕으로 강화학습 알고리즘이 상태와 보상을 전달받는다. 강화학습 알고리즘은 전달받은 보상과 상태를 바탕으로 파라미터를 학습하고 이를 바탕으로 수정된 행동을 전달하는 과정을 반복한다.

그림 2. Unity ML-Agents를 이용한 통신
Fig. 2. Communication using Unity ML-Agents

2.1.1 에이전트 비행체

에이전트는 자율 비행의 주체가 되는 강화학습의 대상으로 Unity에서 제공하는 드론을 이용하였다. 드론 비행의 상태를 에이전트에 입력하여 외부의 강화학습 알고리즘에 이를 전달하면 드론의 적절한 다음 행동을 강화학습 알고리즘이 학습한다. 드론의 비행의 상태와 행동은 표1에 정의된 벡터로 정의하였다.

드론의 비행 상태는 3개로 나눌 수 있는데 드론과 비행 지점을 연결하는 벡터, 속도 벡터, 각속도 벡터이다.

표 1. 드론 비행 상태와 행동 벡터
Table 1. State of Drone Flight and Action Vector

Vector	Axis	Description
Direction	x	longitudinal direction
	y	vertical direction
	z	lateral direction
Speed	x	longitudinal speed
	y	vertical speed
	z	lateral speed
Angular Velocity	x	longitudinal angular velocity
	y	vertical angular velocity
	z	lateral angular velocity
Action	**Axis**	**Description**
Drone	x	longitudinal move
	y	vertical move
	z	lateral move

드론과 비행 지점을 연결하는 벡터는 비행 지점이 드론과 어느 방향에 어느 정도 거리에 있는지를 인지할 수 있게 해준다. 또한 속도 벡터와 각속도 벡터를 통해 비행 지점에 드론이 어떠한 속도와 각속도를 가지고 접근할 것인지를 학습할 수 있게 해준다. 각각의 벡터는 x, y, z 성분을 가지기 때문에 총 9개의 성분을 상태로 사용한다. 그리고 드론의 환경은 3차원 공간으로 구성되기 때문에 드론이 행동할 방향은 x축, y축, z축으로 총 3개의 행동을 취할 수 있다.

2.1.2 음향 탐지 성능 분포도

음향 탐지 성능 분포도(performance distribution)란 목표 해역의 해양 환경 정보를 토대로 소나의 성능을 수치반경으로 나타낸 것으로 소노부이 및 수중 센서노드 등의 최적 배치 연구 방안 탐색에 많이 활용되어왔다[7],[8]. 그림 3의 음향 탐지 성능 분포도는 백색 부분일수록 탐지거리가 높은 지점이다. 백색 부분에 소노부이를 투하할 경우 최대 탐지 영역을 확보하여 소노부이 배치 효과를 극대화할 수 있다. 그리고 이 지점을 잇는 경로를 따라

비행할 경우 최단 시간에 비행이 가능할 것이다. 따라서 음향 탐지 성능 분포도를 통해 어느 지점에 소노부이를 배치해야 최대 탐지 영역을 확보할 수 있는지, 어느 경로로 이동하면 최단 시간에 소노부이를 배치할 수 있는지를 유추할 수 있다. 본 연구에서는 모의적으로 음향 탐지 성능 분포도를 생성하고 에이전트가 이를 인식할 수 있도록 설계하였다.

그림 3의 음향탐지 성능 분포도는 백색 부분일수록 탐지거리가 높은 지점이다. 백색 부분에 소노부이를 투하할 경우 최대 탐지 영역을 확보하여 소노부이 배치 효과를 극대화할 수 있다. 그리고 이 지점을 잇는 경로를 따라 비행할 경우 최단 시간에 비행이 가능할 것이다. 따라서 음향 탐지 성능 분포도를 통해 어느 지점에 소노부이를 배치해야 최대 탐지 영역을 확보할 수 있는지, 어느 경로로 이동하면 최단 시간에 소노부이를 배치할 수 있는지를 유추할 수 있다. 본 연구에서는 모의적으로 음향 탐지 성능 분포도를 생성하고 에이전트가 이를 인식할 수 있도록 설계하였다.

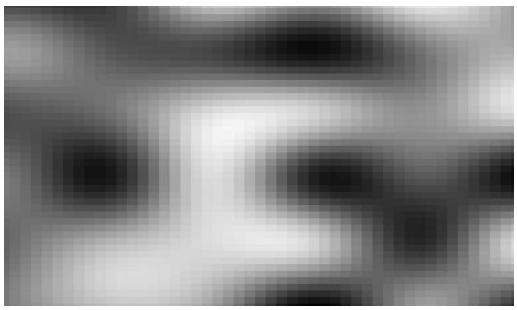

그림 3. 탐지 성능 분포도의 예
Fig. 3. Example of a detection performance distribution plot

그림 4의 왼쪽은 Unity 게임엔진을 통해 구성한 음향 탐지 성능 분포도이다. 가로, 세로 25km 길이의 가상 해역을 설정하고 임의의 70개의 지점에 탐지 반경이 1.5~2.5km가 되도록 하였다. 그림 4의 오른쪽을 보면 실제 음향 탐지 성능 분포도와는 다르게 에이전트인 비행체가 접근할 지점을 직접 구 모양으로 생성한 것을 볼 수 있다. 또한 이 지점 바로 아래 탐지 영역을 나타내는 하얀색 원을 두었다.

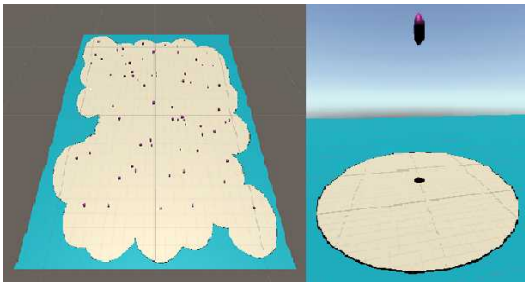

그림 4. Unity 엔진으로 만든 음향탐지 성능 분포도
Fig. 4. Acoustic detection performance distribution chart made with Unity engine

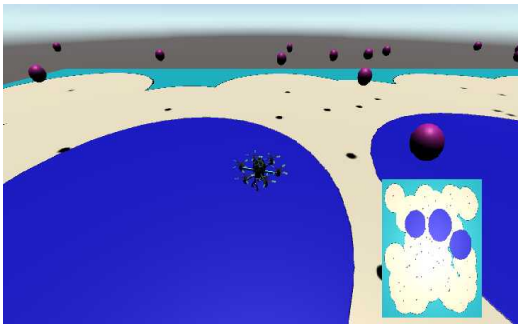

그림 5. 탐지 영역 확보
Fig. 5. Securing the detection area

에이전트가 비행 지점에 접근할 시 그림 5처럼 목표지점 아래 탐지 영역이 파란색으로 바뀌며 해당 해역에서 파란 영역만큼의 탐지 영역을 확보하였다는 것을 시각적으로 나타내었다. 또한, 오른쪽 아래에 미니맵을 두어 전체적인 시각으로 볼 수 있게 하였다.

2.2 보상 설계

강화학습에서는 일정한 스텝 주기의 에피소드마다 에이전트가 행동을 하고 이에 따라 변화된 환경이 에이전트에게 보상을 전달하게 되며 에이전트는 다시 보상을 최대화하는 방향으로 행동을 하게

된다. 따라서 에이전트의 비행 목표에 맞게 학습 방향을 지정할 수 있도록 적절한 보상을 설계하는 것이 매우 중요하다.

본 연구의 에이전트 비행체는 목표지점에 최단 시간 도달하여 최적 배치의 요건을 충족해야 한다. 목표지점에 최단 시간에 다가가는 것을 우선적으로 고려한 보상설계 기준(알고리즘)은 다음과 같다.

[1] Distance =
(목표지점 벡터 - 에이전트 위치 벡터).magnitude

[2] if (Distance 〈 1f)
보상 10 부여

[3] else if (Distance 〉 90f)
보상 -100 부여 & Done()

[4] else
보상 = 이전 거리 - 현재 거리

1번은 목표지점과 에이전트 사이의 벡터 크기, 즉 거리를 나타내는 것으로 이를 통해 목표지점과 에이전트 사이의 위치 상황을 계속해서 비교할 수 있다. 2번의 상황처럼 벡터 크기가 목표지점인 구의 반지름과 일치할 시, 즉 에이전트가 목표지점에 도달할 시 보상 10을 받음을 의미한다. 3번의 상황은 벡터 크기가 90을 넘어가 목표지점과 에이전트의 거리가 멀어지면 음의 보상인 -100을 받고 에피소드를 종료하여 다음 에피소드로 넘어감을 의미한다. 4번은 에이전트가 목표지점에 매 스텝마다 다가갈 수 있게 하기 위한 것으로 에이전트와 목표지점의 이전 거리와 현재 거리의 차를 보상으로 주어 거리 차가 점점 커지게, 즉 목표지점과 에이전트가 가까워지도록 행동하게 한다.

2.3 학습의 에피소드 진행

Unity를 통해 생성한 환경과 보상 설계를 바탕으로 여러 번의 에피소드 진행을 통해 에이전트를 학습시키는 알고리즘은 다음과 같다.

[1] 가로, 세로 25km 해역 내에 탐지 반경이 1.5~2.5km가 되는 목표지점 후보 70개를 임의의 지점에 배치한다.

[2] 에이전트를 해역 외부의 임의의 지점에 위치시키고 위치를 저장한다.

[3] 에이전트와 가장 가까운 목표지점을 탐색하고 에이전트를 이동시킨다.

[4] 에이전트가 첫 번째 목표지점에 도달할 경우 보상을 부여하고 그 지점의 반경 내에 있는 다른 목표지점들을 모두 제거한다.

[5] 다음 목표지점의 탐지 반경을 고려하여 목표지점을 탐색하고 에이전트를 이동시킨다.

[6] 에이전트가 다음 목표지점에 도달할 경우 보상을 부여하고 그 지점의 반경 내에 있는 다른 목표지점들을 모두 제거한다.

[7] 5번, 6번을 반복한다.

[8] 목표지점이 5개가 되었을 경우 에피소드를 종료하고 1번부터 다시 새로운 에피소드를 진행한다.

에이전트는 해역의 외부의 임의의 지점에 생성되어 내부의 목표지점에 도달하면서 탐지 영역을 늘려나간다. 3번에서 에이전트를 이동시킬 때 가장 중요한 점은 최단 거리로 이동을 해야 한다는 것이다. 이를 위해 에이전트와 가장 가까운 첫 번째 목표지점을 탐색하여 이동하였다. 그리고 5번부터 목표지점을 탐색할 때는 최단 거리와 더불어 최대 탐지 영역에 대한 고려도 함께 수행되어야 한다. 4번과 6번에서 목표지점에 도달할 경우 그 지점 반경 내에 있는 다른 목표지점들을 제거하는 의미는 그림 6의 상황 때문이다. 최대 탐지 영역을 확보한다는 것은 탐지 영역 간에 서로 겹치는 부분이 없이 많은 영역을 확보함을 의미한다. 그림6과 같이 탐지 영역 간에 서로 겹치는 부분이 존재하면 겹치는 영역만큼 손해를 보게 된다.

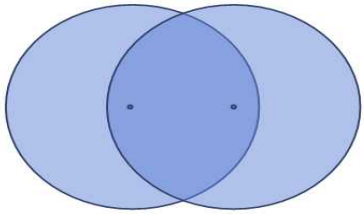

그림 6. 탐지 영역이 겹치는 경우
Fig. 6. When detection areas overlap

최대 탐지 영역을 확보한다는 것은 탐지 영역 간에 서로 겹치는 부분이 없이 많은 영역을 확보함을 의미한다. 그림6과 같이 탐지 영역 간에 서로 겹치는 부분이 존재하면 겹치는 영역만큼 손해를 보게 된다. 따라서 에이전트가 첫 번째 목표지점에 도달함과 동시에 탐지 영역을 손해보지 않으면서 최단거리로 이동할 수 있는 두 번째 목표지점으로 이동하게 해야 최적 배치를 달성할 수 있다. 다음 두 번째 목표지점을 선택하는 방법은 다음과 같다.

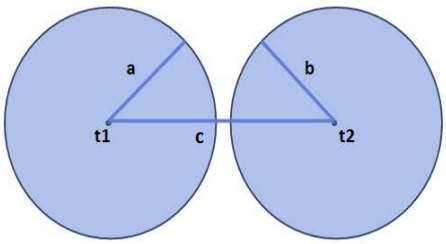

그림 7. 최대 탐지 영역, 최단 거리 목표지점 도출
Fig. 7. Deduction of maximum detection area and shortest distance target point

첫 번째 목표지점 t1의 반경 a와 두 번째 목표지점 t2의 반경 b의 합이 t1과 t2 사이의 거리인 c보다 작거나 같으면 탐지 영역의 손해를 보지 않을 수 있다. 이러한 나머지 두 번째 지점의 후보들 중에서 c의 값이 가장 작은 지점을 고르면 최단 거리로 이동할 수 있다.

따라서 에피소드의 5번부터는 이러한 알고리즘을 따라야 최적 배치를 달성할 수 있다.

3. 강화학습 알고리즘

3.1 정책기반 심층 강화학습

강화학습의 목적은 에이전트가 보상을 극대화하는 방향으로 학습을 진행하는 것이다. 보상을 나타내기 위해서 시간 스텝 t 이후 미래에 얻을 수 있는 보상 $r(x_t, u_t)$와 감가율 γ의 총합인 반환 값 G_t를 식 (1)로 나타낼 수 있다. 이 반환 값을 정책 π_θ로 생성되는 궤적 τ의 확률밀도함수인 $P_\theta(\tau)$의 기댓값으로 나타내면 식 (2)의 목적함수 $J(\theta)$로 나타낼 수 있다.

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r(x_k, u_k) \tag{1}$$

$$J(\theta) = E_{\tau \sim P_\theta(\tau)}[\sum_{t=0}^{T} \gamma^t r(x_t, u_t)] \tag{2}$$

정책(policy)이란 누적된 보상을 가장 많이 획득하기 위해 각 상태에서 어떤 행동을 취할 것인가를 나타내는 조건부 확률 밀도 함수로 $\pi_\theta(u_t|x_t)$로 나타낸다. 따라서 반환 값의 기댓값인 목적함수 $J(\theta)$를 최대로 만드는 정책 $\pi_\theta(u_t|x_t)$을 구하는 것이 정책기반 강화학습의 목적이고 정책이 θ로 파라미터화 되었다면 최적의 θ를 계산하는 것과 동일한 의미이다. 정책이 θ로 파라미터화 될 때 심층 신경망(deep neural network)을 통해 결정되며 이를 정책기반 심층 강화학습이라고 한다. 목적함수 $J(\theta)$를 최대로 만드는 θ를 구하기 위해 $J(\theta)$를 θ로 미분한 $\nabla_\theta J(\theta)$를 로그-정책 그래디언트로 정의하여 식 (3)으로 나타낸다. 목적함수 그래디언트를 통해 경사상승법(gradient ascent)으로 정책의 파라미터 θ를 업데이트하는 과정 (4)를 정책 그래디언트(policy gradient)라고 한다.

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T} (A[\nabla_\theta \log \pi_\theta(u_t \mid x_t) B]) \tag{3}$$

$$A = E_{\tau \sim P_\theta(\tau), u_t \sim \pi_\theta(u_t \mid x_t)}, \quad B = Q^{\pi_\theta}(x_t, u_t)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{4}$$

식 (3)에서 $Q^{\pi_\theta}(x_t, u_t)$는 행동 가치(action-

value) 함수로 어떤 상태변수 x_t에서 행동 u_t를 선택하고 이후 정책 π_θ에 의해 행동이 가해졌을 때 기대할 수 있는 미래의 반환 값을 의미한다. 행동 가치 함수는 정책이 실현되는 시간 스텝 t 에서의 기댓값이기 때문에 목적함수 그래디언트 $\nabla_\theta J(\theta)$를 계산할 때 에피소드가 끝날 때까지 기다릴 필요가 없는 장점이 있다.

3.2 DDPG 알고리즘

DDPG 알고리즘은 액터(actor) 신경망이 행동을 계산하고 크리틱(critic) 신경망이 행동 가치를 계산하여 행동을 개선하는 알고리즘이다[9]. 액터 신경망은 에이전트가 어떻게 행동해야 하는지를 알려주는 신경망으로 정책 신경망이라고 부르기도 한다. 그리고 크리틱 신경망은 액터 신경망을 통해 도출된 행동을 평가하는 역할을 하고 가치 신경망이라고 부르기도 한다.

DDPG와는 다르게 가치기반 심층 강화학습을 토대로 하는 Deep Q-Network(DQN)[3] 알고리즘은 각 뉴런의 출력이 하나의 행동에 대한 행동 가치를 나타내기 때문에 이산적 행동의 문제를 적용하기에 적합하다[10]. 반면 액터-크리틱 구조는 액터 신경망의 확률밀도함수인 정책 근사 함수를 통해 연속적인 행동을 나타내는데 용이하므로 연속적 행동의 문제에 적용하기 좋다.

그러나 드론의 제어와 같은 연속공간의 행동변수를 이용할 경우 정책 신경망의 출력인 $\pi_\theta(u_t|x_t)$를 임의의 확률밀도함수로 두면 신경망에 무한개의 뉴런 수가 필요하므로 실제로 적용하기 어렵다. 따라서 DDPG에서는 정책 신경망이 연속공간 행동변수일 경우 확률밀도함수 $\pi_\theta(u_t|x_t)$를 계산하는 대신 행동 그 자체를 계산한다. 이를 통해 주어진 상태변수 값에서 행동변수의 값을 심층(deep) 신경망을 통해 확정적(deterministic)으로 계산할 수 있다. 그렇게 되면 출력 층 뉴런 개수는 더 이상 무한하지 않으므로 실제로 적용을 할 수 있게 된다. 따라서 본 논문에서는 DDPG 알고리즘을 적용하여 강화학습을 수행하였다.

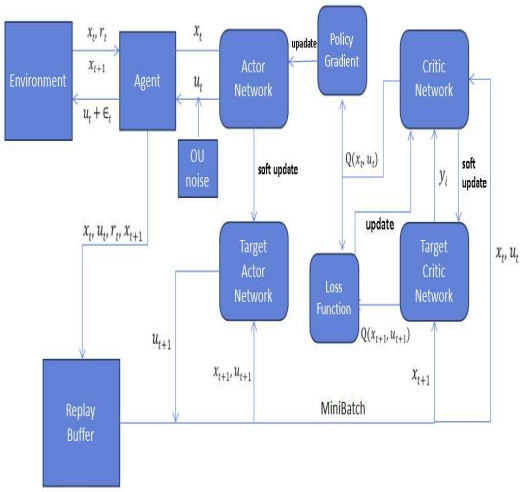

그림 8. DDPG 알고리즘 구조
Fig. 8. Structure of DDPG Algorithm

DDPG의 특징은 3가지가 있다. 먼저 학습할 때 사용하는 궤적 데이터가 시간적으로 상관되어 그래디언트가 편향되는 것을 방지하기 위해 경험 리플레이(experience replay)방식을 사용한다. 이는 에이전트의 경험을 학습에 바로 사용하지 않고 그림 8처럼 리플레이 버퍼에 저장해 두었다가 버퍼에서 샘플을 무작위로 N개 추출하는 방식이다. 다음으로 타겟 액터 신경망과 타겟 크리틱 신경망을 생성하는 것이다. 크리틱 신경망이 업데이트될 때 식 (5)의 손실함수를 이용하는데 이때 식 (6)의 시간차 타겟(TD-Target)이 영향을 받아 변한다.

$$L = \frac{1}{2N}\sum_{i=1}^{N}(y_i - Q_\varnothing(x_i, u_i))^2 \tag{5}$$

$$y_i = r_i + \gamma Q_\varnothing(x_{i+1}, \pi_\theta(x_{i+1})) \tag{6}$$

이렇게 되면 타겟이 계속 바뀌어 안정적인 학습이 불가능하다. 따라서 타겟 액터 신경망과 타겟 크리틱 신경망을 따로 두어 본 신경망의 파라미터를 천천히 따라가도록 한다.

마지막으로 행동에 노이즈 ϵ_t를 도입하는 것이

다. 본래 행동에 무작위성이 있어 주어진 환경을 제대로 탐색해야 하지만 DDPG는 확정적 정책을 사용하기 때문에 정책이 획일적이다. 이를 해결하기 위해 행동에 Orstein-Uhlenbeck 노이즈를 도입하여 식 (7)과 같은 정책을 사용한다.

$$u_t = \pi_\theta(x_t) + \epsilon_t \tag{7}$$

4. 학습 및 학습 결과

4.1 학습

본 논문에서는 액터 신경망과 크리틱 신경망, 학습 방법과 저장을 모두 Python의 Tensorflow를 통해 구현하였으며 이 Python 코드는 Unity를 통해 구성한 환경 및 에이전트와 통신하며 학습이 이루어진다.

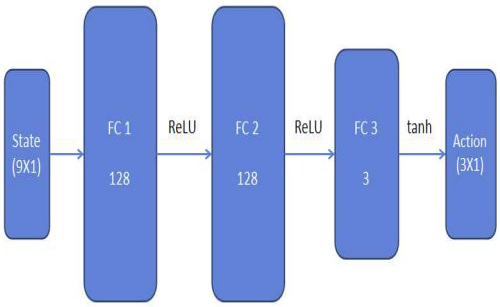

그림 9. 액터 신경망의 구조
Fig. 9. Structure of Actor Network

학습에 사용한 액터 신경망은 그림 9의 구조를 갖는다. 입력층의 상태 변수는 앞서 언급한 드론의 상태에 대한 3가지 벡터와 각각 x, y, z축을 합하여 9가지가 된다. 은닉층은 2개가 존재하는데 모두 완전 연결층으로 각각 128개의 뉴런을 갖고 활성화 함수로 ReLU를 사용하였다. 출력층은 활성화 함수 tanh를 지나 x, y, z축으로 이동하는 행동 하나를 출력하며 이는 에이전트의 움직임이 될 것이다.

에이전트의 상태를 통해 액터 신경망이 행동을 출력하면 크리틱 신경망이 이 행동에 대한 가치를 평가한다. 크리틱 신경망의 학습에 의해 출력되는 행동 가치는 액터 네트워크의 파라미터를 업데이트 하기 위한 목적함수로 사용된다. 학습에 사용한 크리틱 신경망은 그림 10의 구조를 갖는다.

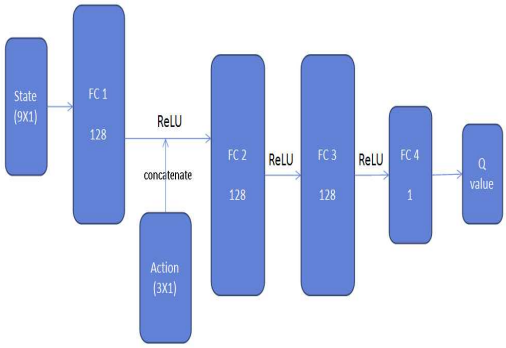

그림 10. 크리틱 신경망 구조
Fig. 10. Structure of Critic Network

에이전트의 상태 변수를 입력으로 128개의 뉴런을 가진 완전 연결층을 지난다. 이는 행동과 결합되어 다시 128개의 뉴런을 가진 완전 연결층 2개를 지난다. 완전 연결층의 활성화 함수는 모두 ReLU이며 출력층은 활성화 함수 없이 행동 가치를 출력한다. DDPG의 학습에 사용한 DDPG 알고리즘의 하이퍼 파라미터는 표 2와 같다.

표 2. DDPG 알고리즘의 하이퍼 파라미터
Table 2. Hyper Parameter of DDPG Algorithm

Parameter	Description	Value
ACTOR_LR	actor learning rate	0.0001
CRITIC_LR	critic learning rate	0.001
TAU	target update rate	0.001
DC	discount factor	0.99
Buffer Size	replay buffer size	60000
MiniBatch	batch size	128
Episode	episode length	500

4.2 학습 결과

학습이 진행되면서 300 에피소드까지는 타겟에 찾아가지 못하고 이상한 방향으로 향하며 에피소드가 빠르게 종료되는 모습을 보였으나 이후 그림 11의 상황처럼 타겟에 최초로 도달했다.

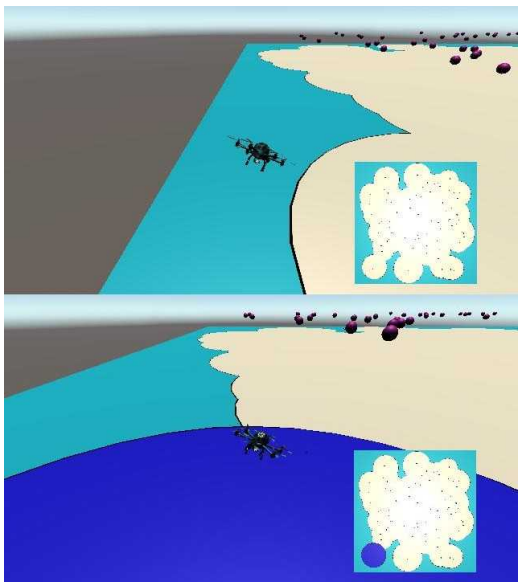

그림 11. 최초로 타겟에 도달
Fig. 11. Reach the first target

이후 보상을 높이는 방향으로 학습을 거듭하여 그림 12의 상황에서처럼 순차적으로 여러 개의 타겟에 도달하는 모습을 보여주었다.

그림 13과 같이 에이전트가 해역을 비행하며 70개의 타겟 후보들 중 최적 배치를 달성하기 위한 지점들만 통과한 것을 확인할 수 있고 탐지 영역을 확보한 모습을 보면 겹치는 영역 없이 최단 거리에 있는 지점을 따라 비행하였음을 알 수 있다. 이는 최적 배치의 요건인 최단시간, 최대 탐지 영역으로 소노부이를 배치하는 자율 비행체를 구현하였음을 의미한다. 결국, 스텝이 모두 지나거나 이상한 방향으로 가서 에피소드가 종료되는 일 없이 그림 13의 상황처럼 5개의 타겟에 모두 도달하며 에피소드가 종료되는 모습을 볼 수 있었다.

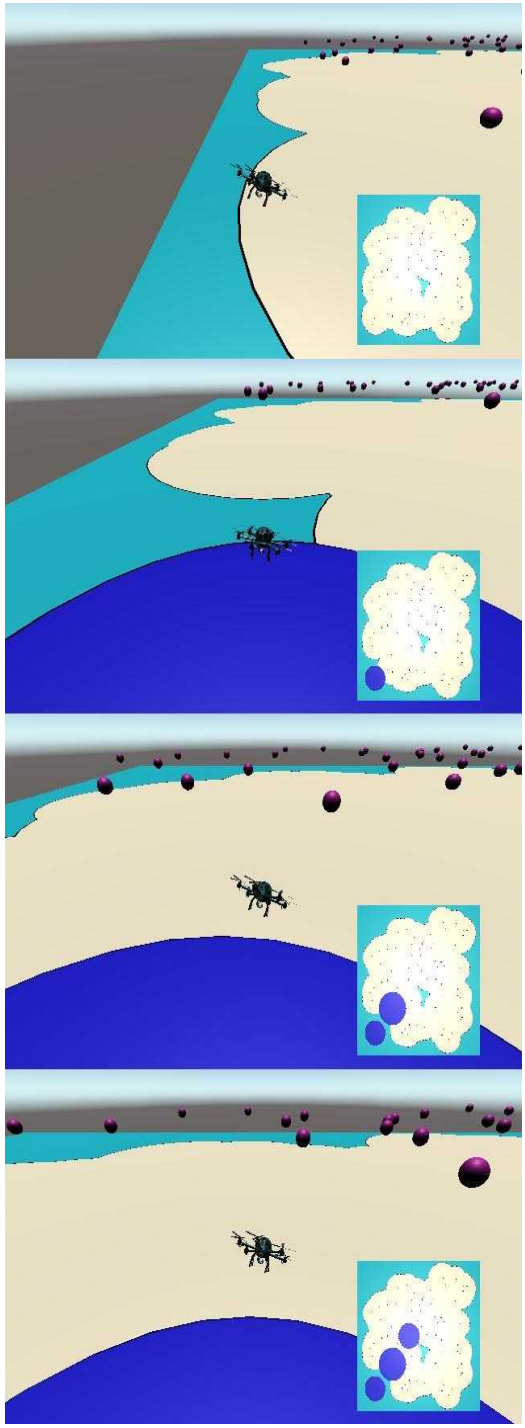

그림 12. 여러 개의 타겟에 도달
Fig. 12. Reach multiple targets

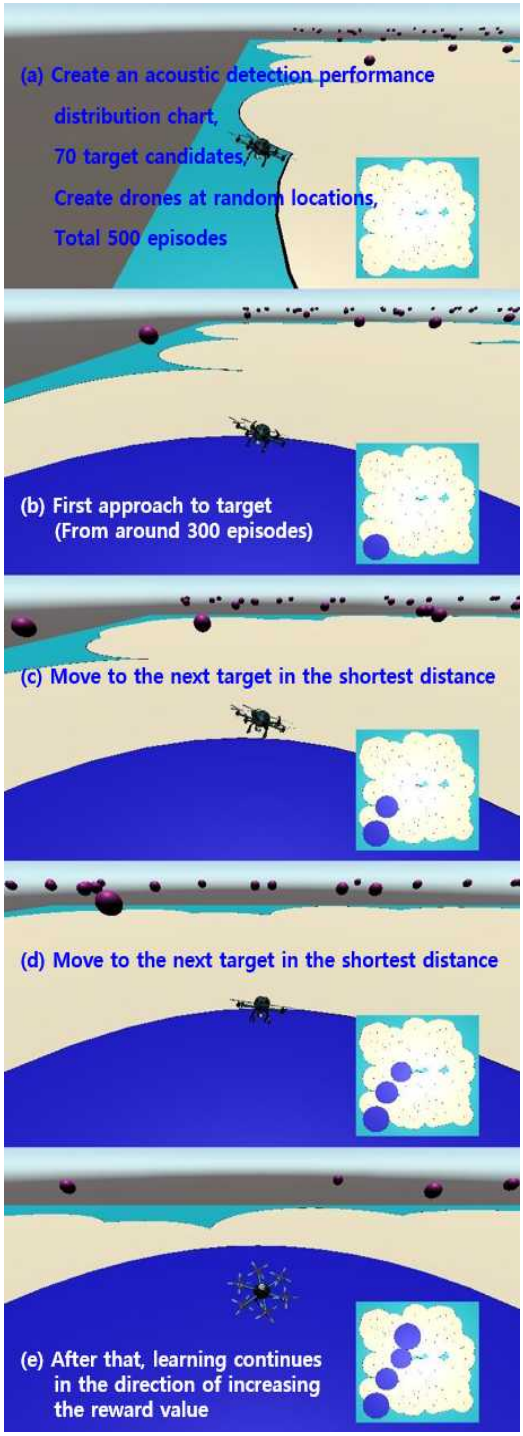

그림 13. 5개의 타겟에 도달하고 에피소드 종료
Fig. 13. Reach 5 targets and end the episode

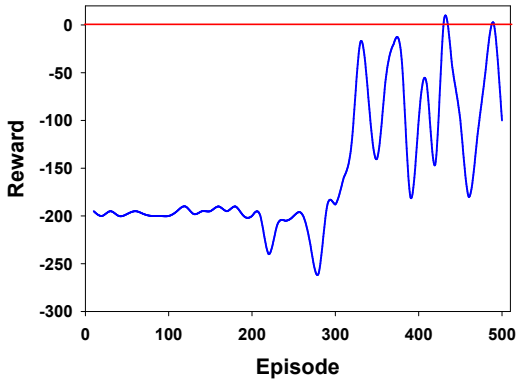

그림 14. 보상 값 변화
Fig. 14. Reward value change

그림 14의 그래프를 보면 300 에피소드까지는 보상 값이 저조하다가 이후로 높아져서 타겟을 지나치고 있다는 것을 알 수 있다. 또한 탐지 영역을 확보한 모습을 보면 겹치는 영역 없이 최단 거리에 있는 지점을 따라 비행하였음을 알 수 있다. 이는 소노부이의 최적 배치를 달성하였음을 나타낸다.

5. 결론

본 논문에서는 Unity 게임엔진을 통해 음향 탐지 성능 분포도를 모사한 환경을 구성하고 Unity ML-Agents를 통해 외부에서 Pyhton(Tensorflow)으로 작성된 DDPG 알고리즘과 통신하며 강화학습을 진행하였다. 학습의 대상인 에이전트 비행체는 드론으로 설정하였으며 보상 값을 높이는 학습 결과를 도출하기 위해 타겟으로 접근하는 적절한 보상 설계를 하였다. 학습 결과 에이전트가 해역을 비행하며 70개의 타겟 후보들 중 최적 배치를 달성하기 위한 지점들만을 통과하였다. 이는 최적 배치의 요건인 최단 시간, 최대 탐지 영역으로 소노부이를 배치하는 자율 비행체를 구현하였음을 의미한다.

향후 바다는 점점 더 많은 국가들이 뒤엉킨 복잡한 전장이 될 것이다. 바다의 소음 준위가 높아지면 적 잠수함을 탐지하기는 더욱 어려워지기 때문에 이를 보완할 기술개발이 매우 시급한 과제이다.

향후 더 발전된 환경에서 개선된 알고리즘으로 소노부이의 성능 및 배치 효과를 극대화할 수 있는 연구를 수행할 계획이다.

REFERENCES

[1] From Wikipedia, the free encyclopedia, Sonobuoy, https://en.wikipedia.org/wiki/Sonobuoy

[2] From Wikipedia, the free encyclopedia, Reinforcement learning, https://en.wikipedia.org/wiki/Reinforcement_learning

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning", NATURE, Vol. 518, No.2 pp. 529-533, 2015.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms", OpenAI, 2017.

[5] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous Control with Deep Reinforcement Learning", Google Deepmind, 2015.

[6] Vincent Pierre (2017), Unity ML-Agents http://github.com/Unity-Technologies/ml-agents

[7] S. Kim, W. Kim, J. Choi, Y. Yoon and J. Park, "Optimal Deployment of Sensor Nodes based on Peformance Surface of Acoustic Detection", Journal of the KIMST, Vol. 18, No. 5, pp. 538-547, 2015.

[8] M. Cheon, S. Kim, J. Choi, C. Choi, S. Son and J. Park, "Optimal Search Pattern of Ships based on Performance Surface", Journal of the KIMST, Vol. 20, No. 3, pp. 328-336, 2017.

[9] H.W Kim and W.C Lee, "Real-Time Path Planning for Mobile Robots Using Q-Learning", Journal of IKEEE, Vol.24, No.4, pp.71-77, 2020.

[10] J. Kim and S.R Shim, "A Case Study on the Evolutionary Development of U.S Unmanned Aerial Vehicles(UAVs)", Journal of Advances in Military Studies, Vol. 3, No. 2, pp, 17-46, 2020.

[11] Y. Cho, J. Lee and K. Lee, "CNN based Reinforcement Learning for Driving Behavior of Simulated Self-Driving Car", The transactions of The Korean Institute of Electrical Engineers, Vol. 69, No.11, pp.1740-1749, 2020.

[12] S. Park and D. Kim, "Autonomous Flying of Drone Based on PPO Reinforcement Learning Algorithm", Journal of Institute of Control, Robotics and Systems, Vol. 26, No.11, pp. 955-963, 2020.

[13] J. Lee, K. Kim, Y. Kim and J. Lee, "Singularity Avoidance Path Planning on Cooperative Task of Dual Manipulator Using DDPG Algorithm", The Journal of Korea Robotics Society, Vol.16, No.2, pp.137-146, 2021.

[14] S. Park, Reinforcement-Learning with Mathematic, https://github.com/pasus/Reinforcement-Learning-Book

[15] G. Min, M. Shin, S. Yoon, H. Lee, G. Jeong and D. Cho, Reinforcement-Learning with Tensorflow & Unity ML-Agents, https://github.com/reinforcement-learning-kr/Unity_ML_Agents

저자약력

김 종 인 (Jong-In Kim) **[정회원]**

- 2022년 3월 : 해군사관학교 전자제어 공학과 졸업 (공학사)

〈관심분야〉 소노부이, 프로그래밍, 인공신경망, 지능제어시스템 설계, ICT Convergence Platform

한 민 석 (Min-Seok Han) **[정회원]**

- 2002년 : 아주대학교 전자공학 공학사
- 2005년 : 한양대학교 전자통신컴퓨터공학 석사
- 2011년 : 한양대학교 전자통신컴퓨터공학 박사
- 2013년 ~ 2016년 : KAIST (재) 스마트IT융합시스템연구단 연구교수
- 2016년 ~ 2018년 : 오산대학교 전자과 교수
- 2019년 ~ 현재 : 해군사관학교 전자제어공학과 교수

〈관심분야〉 프로그래밍, 인공신경망, 지능제어시스템 설계, ICT Convergence Platform