https://doi.org/10.7840/kics.2024.49.1.165

해상 전장 환경에서의 협력적 멀티 에이전트 강화학습 적용을 위한 시뮬레이터의 설계 및 구현

김 경 범◆, 문 수 정*, 진 우 빈**, 이 혁 준°

Design and Implementation of a Simulator for the Application of Cooperative Multi-Agent Systems in Naval Battle Environments

Kyungbeom Kim◆, Sujeong Moon*, Woobeen Jin**, Hyukjoon Lee°

요 약

최근 4차 산업혁명 기술의 발전에 따라 해양 방위 산업 부문에서도 혁신적인 미래 무인체계 연구가 활발하게 이루어지고 있다. 특히 최근 들어 멀티 에이전트 군집제어 기반의 무인체계 시스템 기술에 대한 관심이 높아지고 있다. 멀티 에이전트 강화학습 기술은 멀티 에이전트 군집제어와 같은 복잡하고 동적인 특성을 갖는 문제를 해결하기 위한 효과적인 기술로 인정받고 있으며, 다양한 알고리즘의 개발과 성능분석을 위해서는 실제 멀티 에이전트 운영환경에서의 모델 학습과 테스트를 수행할 수 있도록 하는 시뮬레이터가 필수적이다. 그러나 실제와 비슷한 해상 전장 환경에서의 멀티 에이전트 알고리즘 검증을 위한 공개 SW 기반 시뮬레이터가 존재하지 않는다. 따라서 본 논문에서는 강화학습 알고리즘을 이용한 USV(Unmanned Surface Vehicle) 협력 시나리오 기반 전장 환경 시뮬레이터의 설계 및 구현 사례를 소개한다. 구현된 시뮬레이터에서 규칙 기반 알고리즘을 적용한 적 USV와의 전투 테스트를 통해 멀티 에이전트 알고리즘들의 학습 성능을 비교한다.

키워드 : 멀티 에이전트, 군집제어, 강화학습, MA-POCA, USV, 해상 무기 체계

Key Words : Multi-Agent, cluster control system, reinforcement learning, MA-POCA, USV, Naval weapon system

ABSTRACT

With the recent changes in the technology and industrial system of the 4th Industrial Revolution, research on innovative future unmanned systems continues to increase in the marine defense industry sector. There is a great interest in multi-agent cluster control systems in unmanned systems. In the case of multi-agent cluster control systems, there are usually many complex and dynamic problems, so reinforcement learning algorithms are effective to solve these problems. However, there is a deficiency in simulators for verifying multi-agent algorithms in a realistic naval battle environment. Therefore, this study proposes a simulator based on cooperative scenarios of Unmanned Surface Vehicles (USVs) utilizing the MA-POCA (Multi-Agent Posthumous Credit Assignment) reinforcement learning algorithm. Implemented simulator is utilized to compare the learning performance of various multi-agent algorithms through combat tests with simulated adversaries, specifically rule based USVs.

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 지원사업의 연구결과로 수행되었음(2017-0-00096)
◆ First Author : School of Computer and Information Engineering, Kwangwoon University, kyungbeom8@naver.com, 학생회원
° Corresponding Author : Department of Computer Engineering, Kwangwoon University, hlee@kw.ac.kr, 종신회원
* School of Electronic and Communication Engineering, Kwangwoon University, suu3479@gmail.com
** Department of Computer Engineering, Kwangwoon University, ubinjin2@naver.com, 학생회원
논문번호 : 202310-115-A-RE, Received October 27, 2023; Revised December 7, 2023; Accepted December 22, 2023

Ⅰ. 서 론

최근 해군은 미래 국방의 발전을 위해 2040년대 함대사령부를 감축하고 해양무인전력사령부를 창설해 해양 무인체계 운용을 목적으로 한 연구 및 개발을 추진하는 등 해양 무인 체계 발전에 관심을 기울이고 있다.[1]

현재, 무인체계 연구는 독립적인 무인기의 임무 수행을 지향하고 있다. 강화학습 기술을 사용한 해양 전투 싱글 에이전트 적용 연구와 같이, 특정 환경에서의 싱글 에이전트 적용 연구 결과는 이미 발표된 바 있다.[2]

이와 다르게 멀티 에이전트 시스템은 독립적인 여러 에이전트들이 서로 상호작용하는 시스템이다. 싱글 에이전트는 에이전트 개인의 목표를 달성하려는 반면 멀티 에이전트는 에이전트 개인의 목표가 아닌 공동의 목표 달성을 우선시한다. 실제 해전 관련 임무 수행 시, 한 척의 전함이 동작하지 않고 다수의 전함이 함께 임무를 수행하는 경우가 대부분이다. 이러한 점들을 고려하여 다수의 에이전트들 간 협력을 통해 공동의 목표를 달성할 수 있는 멀티 에이전트 시스템을 활용하는 것이 유용하다.

무인 체계의 연구 및 개발에 있어서, 양질의 데이터를 수집하기에 적합한 학습 환경을 조성하는 것이 가장 중요하다. 즉, 양질의 데이터를 제공하는 시뮬레이터는 중요한 요소가 된다. 시뮬레이터에서 멀티 에이전트 강화학습의 알고리즘 증명 및 개발, 성능 테스트를 할 수 있다. 에이전트는 시행착오 방식을 통해 강화학습을 수행하기 때문에 실제 환경에서 학습하는 것이 불가능 하다. 특히 에이전트가 특수 목적으로 활용되거나 복잡한 사고를 요하는 경우, 환경으로부터 점진적으로 경험을 쌓는 학습이 중요한 요소로 작용하기 때문에 적절한 학습 환경 조성이 필수적이다. 해전에서는 지형적 특성과 복잡한 전투 상황을 고려하여 에이전트를 학습시켜야 한다. 따라서 시뮬레이터를 활용해 에이전트들이 실제 해전에서도 대처할 수 있는 수준의 학습이 진행된다면, 학습을 통해 얻은 협력과 경험으로 에이전트들이 상황을 해결하고 대처할 수 있는 기반이 된다.

다중 에이전트 학습 기능을 지원하는 대표적인 시뮬레이터로는 SMAC (StarCraft multi-agent challenge)이 있다.[3] SMAC은 실시간 전략 게임인 스타크래프트2 환경에서 협력적인 다중 에이전트 학습 문제를 해결하기 위해 연구 및 평가를 위한 표준화된 벤치 마크를 제공한다. SMAC에서는 제한적인 방향으로만 움직이는 에이전트를 사용하고, 낮은 지형과 대비되는 하이 그라운드 지형을 사용하여 몇몇 유닛만 통과할 수 있는 가상 환경을 구성한다. 이와 다르게 해양 전투 시뮬레이터는 실제 해양 환경과 유사하다는 특징을 가진다. 배의 회전과 추진력을 고려하여 실제와 비슷한 행동 공간을 사용하고, 높이 차이가 존재하는 지형 대신 섬과 같은 장애물 지형 및 조류와 같은 해양 지형을 배치하여 해양 환경과 유사하게 구성해야 한다.

본 논문은 멀티 에이전트 강화학습을 활용한 무인 전함 체계 시뮬레이터를 구현한다. 실제 상황과 유사한 환경에서 무인체계의 학습 진행을 목표로 해상 전장과 에이전트, 타겟, 지각 센서, 미사일 시스템 등을 시뮬레이터에 구현한다. 또한 해당 시뮬레이터는 다양한 전투 상황에서 무인 전함 체계 학습이 가능하고, 그 결과를 가시화하는 기능을 제공한다.

Ⅱ. 관련 연구

2.1 유니티(Unity)

유니티(Unity)는 2D, 3D 비디오 게임 및 3D 애니메이션, 가상현실 등 다양한 반응형 콘텐츠 제작 개발 환경을 제공하는 통합 저작 도구이다.[4] 유니티는 크로스 플랫폼으로써 다양한 플랫폼에서의 실행이 가능하다. 위와 같이 유니티는 접근성과 범용성이 좋아 게임 엔진[5] 뿐 아니라, 무인 운용 산업[6], 시뮬레이터 개발[7] 등 다양한 분야에서 활용되고 있다.

특히, 유니티는 특수 목적 시뮬레이터 개발에 유용하다. ADAS 전방 카메라를 적용한 자율주행 차량 시뮬레이터 개발[8], 자율주행 배송로봇을 적용한 도시 설계 시뮬레이터 개발[9], 로봇공학 애플리케이션을 위한 실시간 유니티 3D-MatLab[7] 등등 여러 연구 목적을 달성하기 위해 유니티를 사용한다. 또한 유니티는 실시간 3D 그래픽 및 물리 시뮬레이션을 지원하여, 시뮬레이션 환경 속 오브젝트 간의 상호작용 결과를 실시간으로 확인할 수 있다.

2.2 ML-Agents 툴킷

ML-Agents는 유니티에서 개발된 오픈소스 기계학습 프레임워크로, 유니티 게임엔진과 강화학습 기술을 결합하여 AI에이전트를 개발하고 학습시킬 수 있는 도구이다. 이 프레임워크의 구조는 학습 환경과 학습 알고리즘으로 구성된다.

학습 환경은 크게 에이전트, 브레인, 아카데미로 구성된다.[10] 에이전트는 환경으로부터 관찰을 통해 환

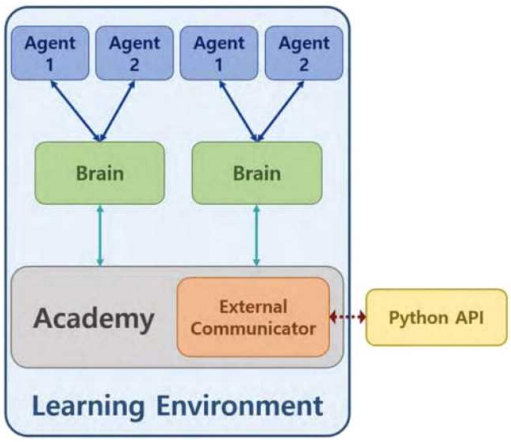

그림 1. Unity ML-Agents구조
Fig. 1. Unity ML-Agents Structure

경 정보를 가져오고 브레인으로부터 받은 행동을 수행하며 행동 수행 결과로 보상을 받는 주체자이다. 브레인은 하이퍼파라미터 값을 설정할 수 있고 같은 그룹에 속해 있는 에이전트들의 관찰값을 하나로 합치는 역할을 담당한다. 아카데미에는 외부 커뮤니케이터(External Communicator)가 존재한다. 외부 커뮤니케이터는 브레인으로부터 받은 모든 에이전트들의 관찰 및 보상을 수집하여 기계학습 알고리즘을 수행하는 외부 파이썬 프로그램에 메시지를 송신하는 역할을 수행한다. 또한, 외부 파이썬 프로그램에서 수신한 메시지를 브레인에게 전달하여 에이전트들이 환경과 상호작용이 가능하도록 학습 데이터를 수집 및 관리한다. 이렇듯 학습 환경은 에이전트가 관찰할 수 있는 상태를 제공하고, 학습 알고리즘을 통해 얻은 행동을 수행하며, 보상을 받도록 구성된다.

학습 알고리즘은 크게 2가지로 ML-Agents 내부에 정의되어 있는 알고리즘을 사용하거나, ML-Agents에서 제공하는 함수를 사용해서 직접 개발된 알고리즘을 사용할 수 있다. 학습 알고리즘의 종류로는 PPO (Proximal Policy Optimization)[11], SAC (Soft Actor Critic)[12], MA-POCA (Multi-Agent Posthumous Credit Assignment)[13]와 같은 싱글 및 멀티 에이전트 강화학습 알고리즘이 있다. 그 이외의 알고리즘들 은 ML-Agents에서 제공하는 함수를 이용하여 적용할 수 있다.

2.3 멀티 에이전트 강화학습 알고리즘

멀티 에이전트 강화학습 알고리즘은 싱글 에이전트 강화학습 알고리즘과 달리 다수의 에이전트가 협업

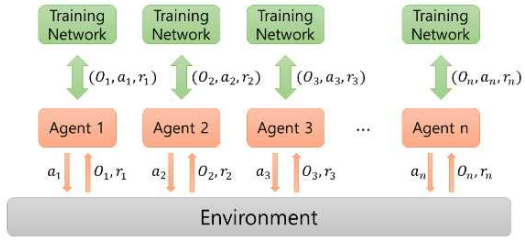

그림 2. DTDE 구조
Fig. 2. DTDE Structure

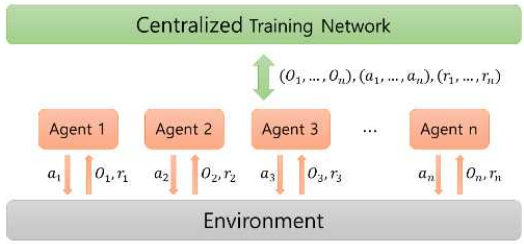

그림 3. CTDE 구조
Fig. 3. CTDE Structure

또는 경쟁하는 알고리즘을 의미한다. 멀티 에이전트 강화학습에는 두가지 프레임워크가 존재한다.

각 에이전트가 개별로 행동 가치 함수를 학습하고 분산형 실행을 하는 DTDE(Decentralized Training and Decentralized Execution) 메커니즘이 있다. 에이전트가 서로 통신할 수 없는 환경에서는 효과적이지만 부분적 관찰값을 이용하기 때문에 전체 상황에 대해 알기 어렵고 에이전트들 간의 상호작용을 고려하지 않아 협력이나 경쟁을 효과적으로 다루기 어려운 단점이 있다.[14]

네트워크 학습에는 모든 에이전트의 관찰값들을 사용하고, 실행 단계에는 개별 에이전트가 각각의 개인 관찰값을 사용하여 독립적으로 행동하는 CTDE (Centralized Training and Decentralized Execution) 메커니즘이 있다. 타 에이전트들의 행동을 이해하고 있기 때문에 비정상성 문제를 해결하기에 효과적이지만 에이전트가 서로 통신할 수 없는 환경에서는 사용하기 힘들다.

2.3.1 IPPO (Independent Proximal Policy Optimization)[15]

DTDE 메커니즘 중 하나인 IPPO는 개별 에이전트가 각각 자신의 행동 정책 및 행동 가치 함수를 학습을 하는 정책 경사 함수 기반 알고리즘이다. 이 때 각 에이전트의 행동 가치 함수의 학습으로 PPO

(Proximal Policy Optimization)[11] 알고리즘을 적용했다. PPO 알고리즘은 정책 경사 알고리즘에서 발생하는 정책의 과한 학습으로부터 일어나는 문제를 방지하고자 대리 함수를 이용한 범위 제한인 신뢰 구간을 만들어 안정적인 신경망의 학습을 가능하게 했다.

$$r_t(\theta) = \frac{(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (1)$$

$$L^a(\theta) = E_{z_t^a, u_t^a}[\min \mathrm{r_t}(\theta)\, A_t^a, clip(\mathrm{r_t}(\theta), 1-\epsilon,\ 1+\epsilon)A_t^a] \quad (2)$$

기존 TRPO (Trust Region Policy Optimization)[16] 알고리즘에서 복잡한 계산인 2차 미분 대신 1차 미분으로 근사화 시켜 계산량을 보완했다.

2.3.2 MA-POCA (Multi Agent Posthumous Credit Assignment)[13]

CTDE 메커니즘 중 하나인 MA-POCA는 에이전트 그룹의 기대 반환 값을 추정하기 위해 중앙 집중식 가치 함수를 학습하고, 그룹 구성원 간의 공헌도 할당을 위해 COMA(Counterfactual Multi-Agent Policy Gradient) 방식의 반사실적 기준선을 적용한 알고리즘이다. 멀티 에이전트 학습 중 에피소드가 끝나기도 전에 먼저 종료되는 에이전트가 존재한다. 즉, 에이전트 수가 불규칙하게 되고, 이로 인해 컴퓨팅 리소스 낭비 문제가 발생한다. 컴퓨팅 리소스 낭비와 공헌도 할당 문제를 해결하기 위해 MA-POCA 알고리즘은 어텐션 층인 RSA 블록을 사용하고 COMA 알고리즘의 반사실적 기준선을 적용한다.

$$Q_\psi(RSA\left(g_j(o_t^j), f_i(o_t^i, a_t^i)_{1\le i\le k_t}\right)(i \neq j) \quad (3)$$

*i*는 에피소드에서 종료되지 않고 활성화 되어있는 에이전트, *j*는 *i*를 제외한 모든 에이전트를 의미한다. COMA 알고리즘의 이득 함수와 다르게 모든 에이전트의 반사실적 기준선을 적용하여 이득 함수를 설정한다. 이를 통해 에이전트가 학습 중에 다른 구성원보다 먼저 에피소드가 끝난 경우에 흡수 상태를 사용하지 않아 효율적으로 그룹에 대한 공헌도를 파악할 수 있다.

Ⅲ. 해양 전장 멀티 에이전트 강화학습 시뮬레이터

3.1 개요

본론에서는 해양 전장 환경에 적합한 시뮬레이터 구성 요소, 에이전트 및 타겟의 기능 소개, 그리고 시뮬레이터 구조에 대해 설명한다.

시뮬레이터 환경 요소는 해양 전장 환경과 비슷하게 에이전트, 타겟, 지형, 바다, 벽으로 구성된다. 군함 에셋을 이용한 에이전트와 타겟은 공격범위, 관찰 범위를 설정하고 실제와 유사한 6-DOF (Six degrees of freedom) 움직임을 갖는 군함 모델을 사용한다. 그리고 바위 에셋을 이용한 지형은 바다 한가운데 350*350 범위에 무작위로 생성되며, 에이전트가 충돌할 경우 즉시 파괴된다. 또한 HEC-RAS(Hydrologic Engineering Center's River Analysis System)[17]의 조류 모델을 기반으로 바다를 구현한다. 또한 바다를 둘러싸는 4개의 벽으로 에이전트가 충돌할 경우 즉시 파괴되어 전장 환경을 벗어나지 않도록 학습 환경을 구성한다.

시뮬레이션 진행 상황을 보여주는 세 가지 로그 출력 화면을 해양 전장 환경과 함께 구현했다. 시뮬레이

그림 4. 시뮬레이터 환경
Fig. 4. Simulator Environment

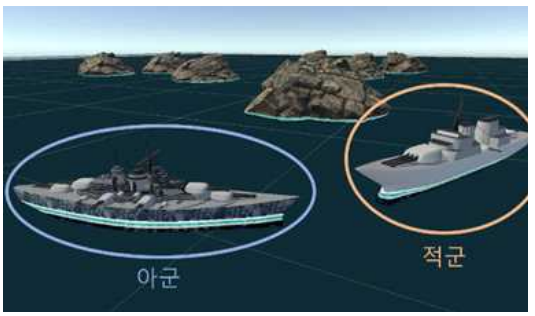

그림 5. 시뮬레이터 환경 내 오브젝트
Fig. 5. Objects in the Simulator Environment

터 환경 내 에피소드 단위의 시간, 스텝 수 그리고 각 에이전트의 누적 보상값을 출력한다.

타겟과 에이전트는 각각 한 가지, 두 가지 형태의 레이더를 보유하고 있다. 타겟은 물체 거리 측정 레이더를 활용하여 사정 거리 내에 가장 가까운 에이전트를 감지하면, 미사일을 해당 에이전트 위치로 자동으로 발사한다. 반면 에이전트는 물체 거리 측정 레이더와 물체 탐지 레이더로 구성된다. 물체 거리 측정 레이더는 해당 에이전트와 물체 간의 거리를 측정하고, 타겟의 레이더와 동일한 기능을 수행한다. 물체 탐지 레이더는 미사일의 사정 거리 내에 위치한 물체의 상대적인 위치를 감지하고 그것이 타겟, 에이전트, 지형, 또는 벽인지를 판별한다. 에이전트가 타겟을 탐지하는 경우 가까이 움직이고, 다른 에이전트, 지형 및 벽을 탐지하는 경우 충돌 방지를 목적으로 행동 경로를 개선하는 등 선택적 조치를 취할 수 있다. 이러한 레이더 시스템은 시뮬레이터 환경 내 여러 구성 요소의 특징을 에이전트가 파악할 수 있도록 중개하는 역할을 한다. 이와 같은 기능은 무인 체계의 작동과 전투 상황에서의 성공률을 높이기 위한 요소로 작용한다.

본 논문은 시뮬레이터에서의 멀티 에이전트 기술 적용 가능 여부 확인을 위해 ML-Agents 내에 정의된 PPO 알고리즘 및 MA-POCA 알고리즘을 활용한다. 위 알고리즘은 멀티 에이전트 강화학습 환경에서의 성능 향상을 목표로 에이전트 간의 상호작용과 협력을 촉진하는 역할을 수행한다. 시뮬레이션을 통해 얻은 결과를 분석하고 멀티 에이전트 기술의 실제 적용 가능성을 평가함으로써, 무인 체계의 성능 개선과 효율성 향상에 기여한다. 위와 같은 방식을 사용하여 시뮬레이터 내에서 에이전트의 학습 환경을 조정하여 원하는 학습 결과를 도출할 수 있다. 무인 체계 학습을 진행하여 시뮬레이션에서의 결과를 토대로 멀티 에이전트 기술의 유효성을 검증하고자 한다.

3.2 전체 시뮬레이터 구조

본 논문에서 멀티 에이전트 강화학습 적용 무인 전함 체계 시뮬레이터 구조는 전체 시뮬레이터 동작 구조도, 에이전트 학습 과정, 에이전트와 타겟의 클래스 다이어그램, 세 가지로 구성된다.

그림 6은 ML-Agent 구조를 적용한 시뮬레이터의 구조도 및 에이전트의 상세 기능을 설명한다. 시뮬레이터 속 스테이지 관리자는 크게 오브젝트 생성, 값 설정, 에이전트 그룹으로 나뉜다. 오브젝트 생성의 경우 USV인 에이전트와 타겟, 섬으로 구성된 지형을 생성한다. 즉, 해상 전장 환경에 알맞은 학습을 위한 오

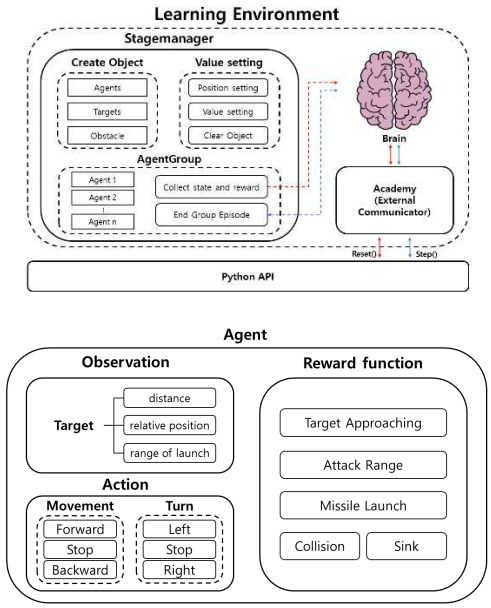

그림 6. 전체 시스템 구조도 및 에이전트 기능
Fig. 6. Overall System Structure and agent functions

브젝트들을 생성하는 역할을 수행한다. 값 설정의 경우 오브젝트에 대한 초기 값을 설정하는 부분으로 학습의 초기 값을 지정하는데 필요한 모듈이다. 내부적으로 에이전트 및 타겟의 위치 값 초기화, 에이전트 및 타겟의 속성 초기화, 에피소드가 종료될 때 남은 오브젝트 삭제 역할을 수행한다. 오브젝트 생성 모듈 다음으로 동작을 수행한다. 에이전트 그룹의 경우 에이전트를 관리하는 역할을 수행한다. 이는 멀티 에이전트 강화학습 알고리즘을 적용하는데 필요한 동작들을 수행한다. 내부적으로 에이전트를 하나의 그룹으로 묶어 모든 에이전트의 관찰값 및 보상값을 수집하고, 모든 에이전트가 함께 에피소드를 종료할 수 있도록 한다. 값 설정 이후에 학습 과정에서 사용된다.

에이전트는 크게 관찰, 행동, 보상 함수로 구성된다. 관찰은 타겟과의 거리, 타겟의 상대 위치 벡터, 발사 범위에 타겟이 존재하는지 여부로 구성된다. 이러한 관찰들은 타겟의 정보를 이용해서 다음 행동을 수행하는데 필요하다. 행동은 크게 두가지로 움직임과 회전 기능이 있다. 움직임은 전진, 정지, 후진 움직임으로, 회전은 좌회전, 정지, 우회전으로 구성된다. 실제 USV와 유사한 행동으로 구성한다. 보상함수는 5가지로, 타겟으로 가까이 가는지 여부, 타겟이 에이전트의 공격 범위에 존재하는지 여부, 미사일을 발사했는지 여부, 에이전트, 타겟, 지형, 벽과 충돌했는지 여부, 미사일로 격추당했는지 여부로 구성된다. 보상함

수를 통해 에이전트가 모든 오브젝트와 충돌하지 않고 타겟으로 가까이 접근하여 격침시키도록 유도할 수 있다. 에이전트는 관찰을 수집하고 행동을 수행하고 행동에 대한 결과를 보상으로 획득하며 관찰 및 보상 그리고 강화학습 알고리즘을 이용하여 내부 네트워크를 학습한다. 이러한 학습된 네트워크는 에이전트가 높은 기대 보상을 얻을 수 있는 행동을 수행하게 한다.

에피소드가 시작되기 전 스테이지 관리자는 먼저 에이전트, 타겟, 지형을 설정한 수만큼 생성한다. 그리고 에이전트들의 경우 멀티 에이전트 학습을 위해 하나의 그룹으로 구성한다. 생성된 모든 오브젝트들의 위치를 무작위로 설정하고, 에이전트와 타겟의 경우 선체의 방향도 동시에 설정한다. 그리고 에이전트와 타겟 각각의 속성들을 초기화한다.

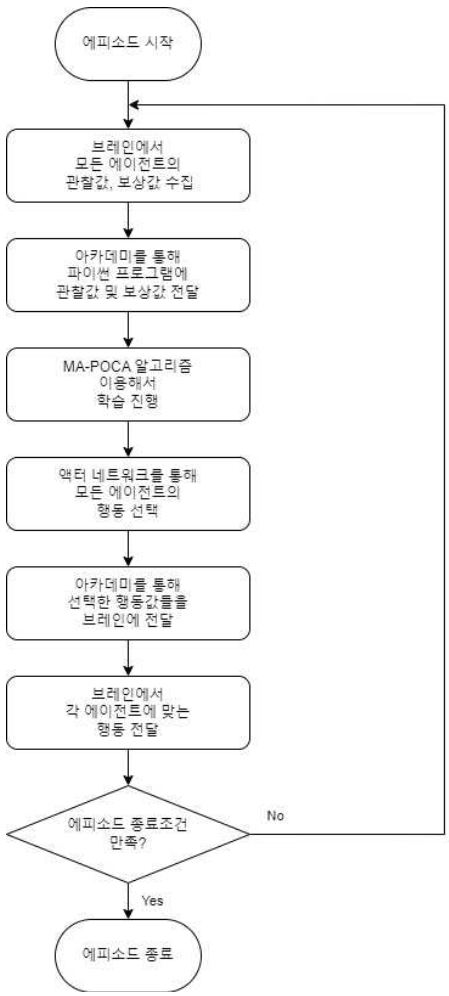

그림 7. 에이전트 학습 과정 플로우차트
Fig. 7. Flowchart for the agent's learning process

모든 설정이 끝난 후 에피소드가 시작되며 에이전트는 타겟과 미사일을 통해 전투를 진행하고, 레이더를 통해 관찰한 타겟 정보를 이용하여 행동을 수행한다. 그리고 설정한 보상 함수를 통해 행동에 대한 보상을 획득한다. 또한 획득한 보상을 통해 에이전트 학습을 진행한다. 관찰값, 행동값, 보상값에 대한 자세한 내용은 4.1장에서 설명한다.

에피소드가 시작될 때부터 끝날 때까지의 학습 과정을 다음과 같이 플로우차트로 나타낼 수 있다.

학습 진행에 사용된 MA-POCA 알고리즘은 그림 8에서 상세히 설명한다.

에피소드가 종료되는 조건은 다음과 같다. 에이전트가 모두 전멸된 경우, 타겟의 승리로 에피소드가 종료된다. 반대로 타겟이 모두 전멸된 경우, 에이전트의 승리로 에피소드가 종료된다. 이외에 지정한 에피소드당 스텝 수를 초과할 경우 무승부로 에피소드가 종료된다. 에피소드가 종료되는 경우 학습 환경에 남은 오브젝트의 속성을 초기화하고, 위치를 무작위로 재배치한 후 다음 에피소드를 시작한다. 이러한 과정은 하이퍼파라미터로 설정한 학습 스텝 수까지 반복한다.

Algorithm 1 MA-POCA

Input: o_t^i, a_t^i for agent i and time step t
k_t = Number of Active agents at time step t
N = maximum number of agents

for all agent i ∈ 1,...,k_t **do**
  get $g_i(o_t^i)$ by encoding block g
end for
get $RSA(g_i(o_t^i)_{1\le i\le k_t})$
for all agent j ∈ 1,...,N **do**
  get $g_j(o_t^j)$ by encoding block g
  for all agent i ∈ 1,...,k_t such that i ≠ j **do**
    get $f_i(o_t^i, a_t^i)$ by encoding block f
  end for
  get $RSA(g_j(o_t^j), f_i(o_t^i, a_t^i)_{1\le i\le k_t, i\ne j})$
end for

[Critic Update]
$J(\phi) = (V_\phi(RSA(g_i(o_t^i)_{1\le i\le k_t})) - y^{(\lambda)})^2$
$J(\psi) = (Q_\psi(RSA(g_j(o_t^j), f_i(o_t^i, a_t^i)_{1\le i\le k_t, i\ne j})) - y^{(\lambda)})^2$
Where
$y^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^n$
$G_t^{(n)} = \sum_{t=1}^{n}\gamma^{l-1}r_{t+l} + \gamma^n V_\phi(RSA(g(o_{t+n}^i)_{1\le i\le k_{t+n}}))$
update Q and V by TD Learning and gradient descent

[Actor Update]
$Adv_j = y^{(\lambda)} - Q_\psi(RSA(g_j(o_t^j), f_i(o_t^i, a_t^i)_{1\le i\le k_t, i\ne j}))$
$\nabla_{\theta_j} J(\theta_j) = E[\nabla_{\theta_j} log\pi_j(a^j|o^j)(Adv_j)]$
update θ by policy gradient

그림 8. MA-POCA 알고리즘
Fig. 8. MA-POCA Algorithm

에이전트와 타겟은 각각의 유니티 인스펙터를 통해 이동 속도, 회전 속도, 미사일 속도, 미사일 개수, 관찰 레이저 길이, 관찰 레이저 개수, 공격 범위와 같은 속성들을 개별적으로 조절할 수 있다. (그림 9 USV 및 Target 속성 참조)

에이전트와 타겟의 기능은 서로 유사하다. 에이전트는 이동, 누적 보상 측정, 오브젝트들과의 거리 측정, 타겟이 시야각에 존재하는지 측정, 타겟과 가까워지는지 측정, 타겟의 상대 위치 측정, 미사일 발사, 폭발과 같은 기능을 수행한다. 그리고 타겟은 에이전트와의 거리 측정, 미사일 발사, 폭발과 같은 기능을 수행한다. 타겟의 경우 에이전트와 다르게 규칙 기반 알고리즘을 통해 단순한 움직임을 구현한다. (그림 9 USV 및 Target 메소드 참조)

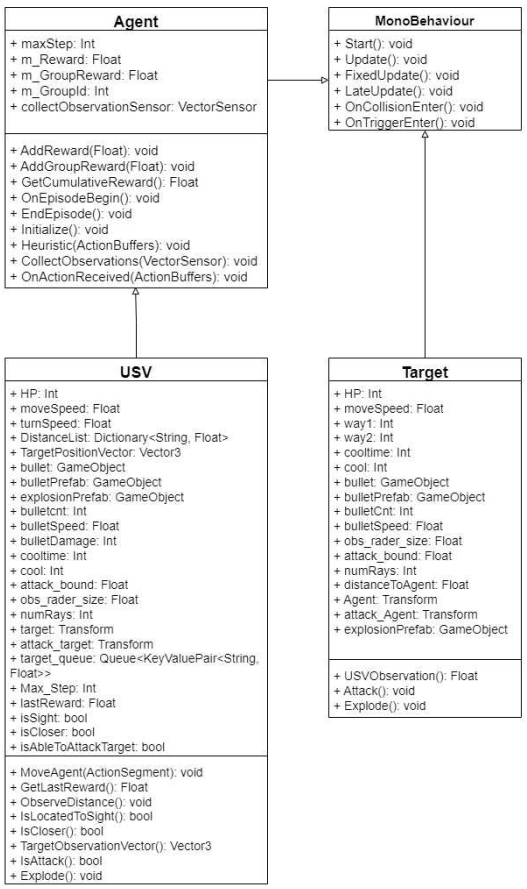

그림 9. 에이전트, 타겟 클래스 다이어그램
Fig. 9. Agent, Target class diagram

Ⅳ. 실 험

4.1 실험 조건

본 실험에서는 3장에서 제안한 해양 전투 시뮬레이터를 이용하여 에이전트 학습을 진행하고 학습 결과 비교 및 성능에 대한 평가를 진행한다. 실험에는 유니티 2022.1.21f1, ML-Agents 2.3.0-exp.3, 파이썬 3.8.5, 파이토치 1.13.0 버전을 사용했다. 그리고 학습 알고리즘으로는 IPPO 알고리즘 및 MA-POCA 알고리즘을 사용했다.

4.1.1 실험 데이터 정보

실험은 타겟과 에이전트를 동일한 조건에서 실험하도록 조건을 설정했다. 스테이지 관리자 인스펙터 창에서 타겟과 에이전트의 프로퍼티 수치를 동시에 조절이 가능하다.

표 1. 스테이지 관리자 인스펙터
Table 1. Stage manager inspector

속성	값
에이전트 HP	200
에이전트 미사일 개수	20
에이전트 미사일 재사용 대기시간 (초)	5
에이전트 속도 (m/s)	20
에이전트 회전 속도 (°/s)	20
에이전트 미사일 속도 (m/s)	1000
에이전트 미사일 개수	20
에이전트 미사일 데미지	80
에이전트 관찰 레이더 길이 (m)	3000
에이전트 관찰 레이더 개수	400
에이전트 공격 범위 (m)	200
타겟 HP	200
타겟 미사일 개수	20
타겟 미사일 재사용 대기시간 (초)	5
타겟 속도 (m/s)	20
타겟 회전 속도 (°/s)	20
타겟 미사일 속도 (m/s)	1000
타겟 미사일 개수	20
타겟 미사일 데미지	80
타겟 관찰 레이더 길이 (m)	3000
타겟 관찰 레이더 개수	400
타겟 공격 범위 (m)	200
에이전트 개수	3
타겟 개수	3
에피소드 당 최대 스텝 수	10000

타겟과 에이전트 모두 동일한 조건으로 설정했고, 미사일의 데미지와 HP를 조절하여 서로 다른 팀의 미사일을 3대 맞을 때 비활성화되도록 구현했다.

관찰 레이더는 실험 스테이지의 일부를 관찰할 수 있도록 레이더 범위를 조정했고, 레이더 각도를 360도로 설정하여 모든 방향을 모두 측정할 수 있도록 했다.

4.1.2 상태 및 행동

에이전트가 매 스텝 측정하는 관찰값은 총 106개로, 2개는 에이전트의 정보(방향벡터 x, 방향벡터 z), 4개는 물체 거리 측정 레이더를 이용하여 측정한 가장 가까운 타겟의 정보(상대위치 x, 상대위치 z, 거리, 발사 범위에 존재 여부), 100개는 20개의 레이더에서 각각 5개의 정보로, 물체 탐지 레이더를 이용하여 측정한 정보(레이더에 탐지 여부, 거리, 물체의 종류)를 사용한다.

행동의 경우 실제 수상함정과 유사한 방식으로 구현했다. 행동 종류는 추진 장치 기반 행동과 조향 장치 기반 행동으로 총 2가지이다. 추진 장치를 기반으로 구현한 전진, 후진, 정지 행동과 조향 장치를 기반으로 구현한 시계 방향 회전, 반시계 방향 회전, 정지에 대한 행동으로 구성된다. 매 스텝 각각의 행동 종류에서 하나씩 행동을 선택하고, 선택한 두 행동을 동시에 수행한다.

표 2. 관찰값
Table 2. Observation

관찰값	데이터 타입
방향 벡터(x)	float
방향 벡터(z)	float
가장 가까운 타겟의 상대 위치(x)	float
가장 가까운 타겟의 상대 위치(z)	float
가장 가까운 타겟과의 거리	float
가장 가까운 타겟이 발사 범위에 존재	boolean

물체 탐지 레이더	값
레이더 탐지	boolean
거리	0~1
에이전트	One - hot Vector
타겟	
벽 및 지형	

4.1.3 보상

행동을 수행하고 난 후 얻는 보상값의 종류로는 크게 특정 조건을 만족할 때 얻을 수 있는 희소 보상값과 매 스텝 얻는 부분 보상값이 있다.

희소 보상값은 그룹과 개인으로 나뉜다. 그룹 보상은 타겟이 모두 전멸한 경우에 얻을 수 있고, 에이전트가 많이 생존할수록 보상을 많이 받는다. 즉, 에이전트의 전투 수행 능력과 생존율을 높이는 방향으로 보상을 설계했다. 개인 보상은 충돌, 미사일 발사, 타겟 격추에 대한 조건을 만족할 때 얻을 수 있다. 이를 통해 오브젝트와 충돌을 방지하고 미사일 발사를 유도하며 타겟을 격추하는 방향으로 행동을 유도했다.

부분 보상값은 개인에게만 주어진다. 이전 스텝에 비해 현재 스텝에 타겟에게 가까이 갈 경우, 에이전트의 시야각 30도 안에 타겟이 존재하는 경우, 에이전트의 발사 범위에 타겟이 존재하는 경우 양의 보상값이 주어진다. 즉, 매 스텝 에이전트가 시야각을 유지하며 발사범위 내로 타겟에게 가까이 갈 수 있도록 보상을 설계했다. 마지막으로 매 스텝 음의 보상을 주어 에이전트가 효율적인 행동을 탐색하도록 유도했다.

표 3. 그룹 희소 보상값
Table 3. Group sparse reward

조건	그룹 보상값
에이전트 전멸	0
타겟 전멸	5 * (남은 에이전트 수)

표 4. 개인 희소 보상값
Table 4. Personal sparse reward

특정 조건	보상값
벽, 지형, 타겟, 에이전트에 충돌했을 때	-50
미사일을 발사할 때	+10
타겟을 격추 시킬 때	+50

표 5. 개인 부분 보상값
Table 5. Personal partial reward

조건	보상값
타겟에게 가까이 갈 때	+0.015
시야각에 타겟이 존재할 때	+0.005
발사범위에 타겟이 존재할 때	+0.02
-	-0.01f

4.2 실험 내용

본 실험 내용에서는 4.1장에서 언급한 조건들을 일정하게 유지시키고 아군 및 적군 함정을 다양한 시나리오에 배치하여 학습을 진행한다. 먼저 360도의 방향을 무작위로 정한 후 적군 함정은 그림 10과 같이

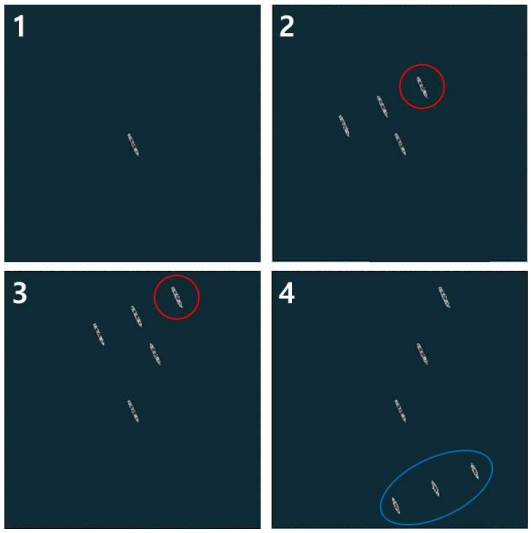

그림 10. 아군 및 적군 배치 순서
Fig. 10. Arrangement order of a allies and enemies

중앙에 하나를 배치한 후 그 뒤로부터 앞에 위치한 함정의 왼쪽, 가운데, 오른쪽을 무작위로 선택하여 배치했다. 그리고 아군 함정을 정중앙과 600m 떨어진 위치에 횡대로 마주보도록 배치했다.

하이퍼파라미터 설정의 경우 MA-POCA[14]를 참고했다. 배치와 버퍼 크기는 각각 512, 204,800, 그리고 학습률은 0.0003을 사용했으며 베타는 0.006, 총 학습 스텝 수는 이천만 스텝을 사용했다. 하이퍼파라미터 설정에서 가장 큰 성능 변화를 보인 부분은 의사 결정 주기이다. 이는 행동을 결정하는 시간 주기로 수치를 변경하며 모의 실험을 진행했다. 이에 대한 결과는 다음과 같다.

의사 결정 주기가 1일 때보다 5 이상에서 수렴성을

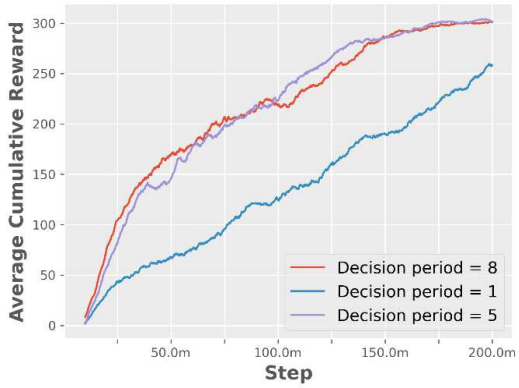

그림 11. 의사결정주기 비교 실험 결과
Fig. 11. Result of comparative experiment on the decision period

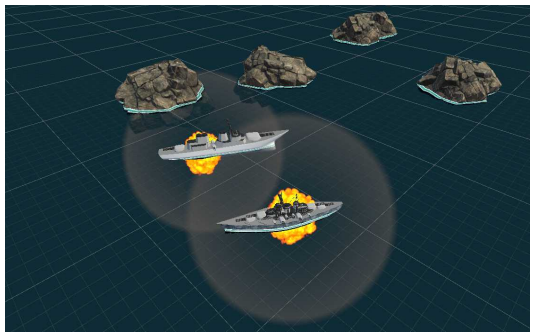

그림 12. 시뮬레이터 환경 내 전투 장면
Fig. 12. Combat Scenes in Simulator environment

보이며 이를 통해 안정적인 학습 결과를 확인할 수 있다. 따라서 의사 결정 주기를 5로 설정했다. 그리고 유니티에서 제공하는 IPPO 알고리즘 및 MA-POCA 알고리즘을 이용하여 학습을 진행한 후 결과 및 성능 분석을 했다.

4.3 실험 결과

학습이 진행되는 동안 두 알고리즘 모두 개인 보상이 점차 증가하다 100 근방에서 수렴하는 모습을 볼 수 있다. 이 때 IPPO 알고리즘이 MA-POCA 알고리즘보다 빠르게 보상값이 증가했다. 하지만 학습이 점점 진행되면서 MA-POCA 알고리즘이 더 높은 보상값을 얻는다.

에피소드 길이의 경우 학습이 진행되면서 두 알고리즘 모두 점차 감소한다. 그림 13와 비교할 때, 보상값이 교차하는 근방에서 에피소드의 길이도 교차하는 모습을 확인할 수 있다. 이를 통해 MA-POCA 알고리즘이 IPPO 알고리즘보다 학습이 완료된 시점에서 빠른 시간 안에 큰 보상을 얻고 승리하는 경험을 터득한

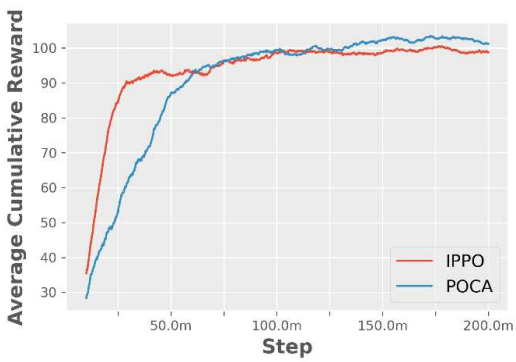

그림 13. 두 알고리즘 보상 값 비교 (IPPO, MA-POCA)
Fig. 13. Comparison of reward values for two algorithms, IPPO and POCA

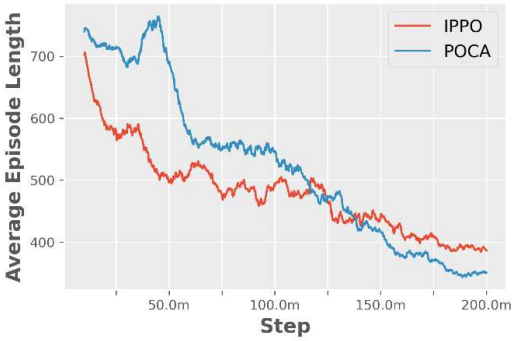

그림 14. 두 알고리즘 에피소드 길이 비교 (IPPO, MA-POCA)
Fig. 14. Comparison of episode length for two algorithms, IPPO and POCA

것을 알 수 있다.

4.4 승률 비교

IPPO 알고리즘과 MA-POCA 알고리즘으로 학습한 모델을 이용하여 타겟의 배치가 기존과 다른 시나리오에서 테스트를 진행했다. 100번의 에피소드를 수행하여 모든 타겟을 물리칠 시 승리, 모든 에이전트가 격추될 시 패배, 에피소드 동안 승리나 패배 조건을 달성 못할 시 무승부의 조건을 가지고 승률을 분석했다.

표 6을 통해 두 알고리즘 모두 높은 승률을 보이지만 MA-POCA 알고리즘이 IPPO 알고리즘보다 근소하게 승률이 높은 것을 알 수 있다. 이를 통해 멀티 에이전트 환경에서 MA-POCA 알고리즘이 IPPO 알고리즘보다 근소하게 좋은 성능을 보이는 것을 알 수 있다.

표 6. 승률 비교 결과
Table 6. Winning rate comparison result

	IPPO	MA-POCA
승	79	86
패	18	14
무	2	0

V. 결 론

본 논문에서는 해상 전함 시뮬레이터를 구현하고 멀티 에이전트 강화학습 알고리즘을 활용한 무인 협력 모델을 제안하며 이를 통해 해양 전함 무인 체계 연구의 발전 방향을 제시한다.

시뮬레이터 개발 연구를 통해, 무인 체계 분야의 실전 운용에 특화된 멀티 에이전트 강화학습 알고리즘 학습을 위한 기반을 마련한다. 멀티 에이전트 학습을 위한 최적의 시뮬레이션 환경을 제공함으로써 연구자들은 다양한 시나리오에서 멀티 에이전트 기술을 활용하여 멀티 에이전트 모델을 실험하고 결과를 체계적으로 분석할 수 있다. 이를 통해 멀티 에이전트 모델의 성능을 비교적 짧은 시간 내에 객관적으로 평가하고 향상시킬 수 있는 기회를 제공한다.

두 개의 멀티 에이전트 강화학습 알고리즘을 이용하여 해양 전장 시뮬레이터에서 학습을 진행한 결과, 두 알고리즘 모두 보상이 증가하고 에피소드 길이가 짧아지는 것을 확인할 수 있다. 하지만 MA-POCA 알고리즘이 IPPO 알고리즘보다 보상값이나 에피소드 길이 측면에서 근소한 우위를 보이는 것을 확인할 수 있다. 또한 승률 비교를 통해 MA-POCA 알고리즘의 근소한 우위를 알 수 있다. 이를 통해 멀티 에이전트 해양 환경에서 MA-POCA 알고리즘이 IPPO 알고리즘보다 근소하게 더 우수한 성능을 보임을 확인할 수 있다.

본 논문은 멀티 에이전트 강화학습 알고리즘을 활용한 무인 체계 자율 운용을 통해 실제와 비슷한 시뮬레이터의 형태로 제시한다. 향후 해당 시뮬레이터를 활용하여 멀티 에이전트 학습을 위해 다양한 실험 및 시나리오를 수행 및 적용함으로써 실전 해전에서의 멀티 에이전트 강화학습 알고리즘을 활용한 멀티 에이전트 모델의 성능 발전에 있어 심층적인 탐구가 가능하다. 시뮬레이션을 통해 얻은 결과를 토대로 현실적이고 효과적인 무인체계 임무수행 전략 연구수행과 더 나아가, 실질적 임무 달성에 주요한 장치가 될 것이다.

References

[1] J. Kim, "Navy reduces fleet by one and established 'Maritime Unmanned Force Command'," *YONHAP NEWS*, Oct, 2022. (https://www.yna.co.kr/view/AKR20221021046500504)

[2] M. Kim, "A Study on the application of a single agent reinforcement learning in marine combat system based on OpenAI Gym," in *Proc. Korea Commun. Soc. Conf.*, pp. 718-719, Nov, 2021. (https://www-dbpia-co-kr.libproxy.kw.ac.kr/journal/articleDetail?nodeId=NODE11022870)

[3] Samvelyan, Mikayel, et al. "The StarCraft

Multi-Agent Challenge.“ Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 2186-2188, May. 2019. (https://doi.org/10.48550/arXiv.1902.04043)

[4] W. Park, “Implementation of an emergency driving training simulator using Unity,” *J. Korea Comput. and Inf. Soc.,* vol. 27, no. 10, pp. 131-136, Oct, 2022. (https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE11150451)

[5] S.-K. Kim, et al., “An Efficient Use Method for Unity 3D Engine,” *Korean Comput. and Inf. Soc. Academic Conf.,* vol. 21, no. 1, pp. 333- 334, Jan, 2013. (https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE06530056)

[6] H.-H. Choi and S.-H. Jo, “Flight simulator Development for a Large Number of UAVs,” *J. Korean Comput. and Inf. Soc. Academic Presentations,* vol. 26, no. 2, pp. 299-300, Jul, 2018. (https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE07513902)

[7] V. H. Andaluz, F. A. Chicaiza, C. Gallardo, W. X. Quevedo, J. Varela, J. S. Sánchez, and O. Arteaga, “Unity3D-MatLab simulator in real time for robotics applications,” in *Proc. Augmented Reality, Virtual Reality, and Computer Graphics*: *Third Int. Conf.*, *Springer Int. Publishing,(AVR 2016)*, Part I 3, pp. 246-263, Lecce, Italy, Jun. 2016. (https://doi.org/10.1007/978-3-319-40621-3_1)

[8] S. Lee, “Mando develops Unity-based ADAS camera simulator,” *e4dsnews*, May, 2021. (https://www.e4ds.com/sub_view.asp?ch=31&t=0&idx=12836)

[9] J. Choi, et al., “Urban Design for an Autonomous Delivery Robot using the Unity based Simulation,” *Korean Spatial Inf. Soc. Conf.,* pp. 293-295, May, 2022. (https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE11078158)

[10] H. Kim, “Development of reinforcement learning environment for smart seismic isolation control system using unity,” *J. Korean Soc. Industry-Academia-Technol.*, vol. 24, no. 5, May, 2023. (https://doi.org/10.5762/kais.2023.24.5.433)

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017. (https://doi.org/10.48550/arXiv.1707.06347)

[12] T. Haarnoja, et al., “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *Int. Conf. Mach. Learn. PMLR*, pp. 1861-1870, 2018. (https://proceedings.mlr.press/v80/haarnoja18b)

[13] A. Cohen, E. Teng, V. P. Berges, R. P. Dong, H. Henry, M. Mattar, and S. Ganguly, “On the use and misuse of absorbing states in multi-agent reinforcement learning,” *arXiv preprint arXiv:2111.05992*, 2021. (https://doi.org/10.48550/arXiv.2111.05992)

[14] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *Proc. Tenth Int. Conf. Mach. Learn.*, 1993. (https://doi.org/10.1016/b978-1-55860-307-3.50049-6)

[15] C. S. de Witt, et al., “Is independent learning all you need in the starcraft multi-agent challenge?,” *arXiv preprint arXiv:2011.09533*, 2020. (https://doi.org/10.48550/arXiv.2011.09533)

[16] J. Schulman, et al., “Trust region policy optimization,” *Int. Conf. Mach. Learn. PMLR*, 2015. (https://doi.org/10.48550/arXiv.1502.05477)

[17] G. W. Brunner, “HEC-RAS river analysis system: Hydraulic reference manual, Version 5.0.,” *US Army Corps of Engineers - Hydrologic Engineering Center,* 547(2016), Retrieved Nov, 11, 2023 from https://apps.dtic.mil/sti/citations/ADA311952

김 경 범 (Kyungbeom Kim)

2018년 3월~현재 : 광운대학교 컴퓨터정보공학부 재학 (학사)

<관심분야> 강화학습, 무인체계, 머신러닝

문 수 정 (Sujeong Moon)

2020년 3월~현재 : 광운대학교 전자통신공학과 재학 (학사)

<관심분야> 임베디드, 인공지능

진 우 빈 (Woobeen Jin)

2022년 : 광운대학교 컴퓨터 정보공학부 졸업 (학사)

2022년 3월~현재 : 광운대학교 컴퓨터공학과 재학 (석사)

<관심분야> 유무선네트워크, 인공지능

이 혁 준 (Hyukjoon Lee)

1987년 : 미시간대학교 컴퓨터과학과 졸업 (학사)

1989년 : 시라큐스대학교 컴퓨터과학과 졸업 (석사)

1993년 : 시라큐스대학교 컴퓨터과학과 졸업 (박사)

<관심분야> 유무선 네트워크, 인공지능, 머신러닝