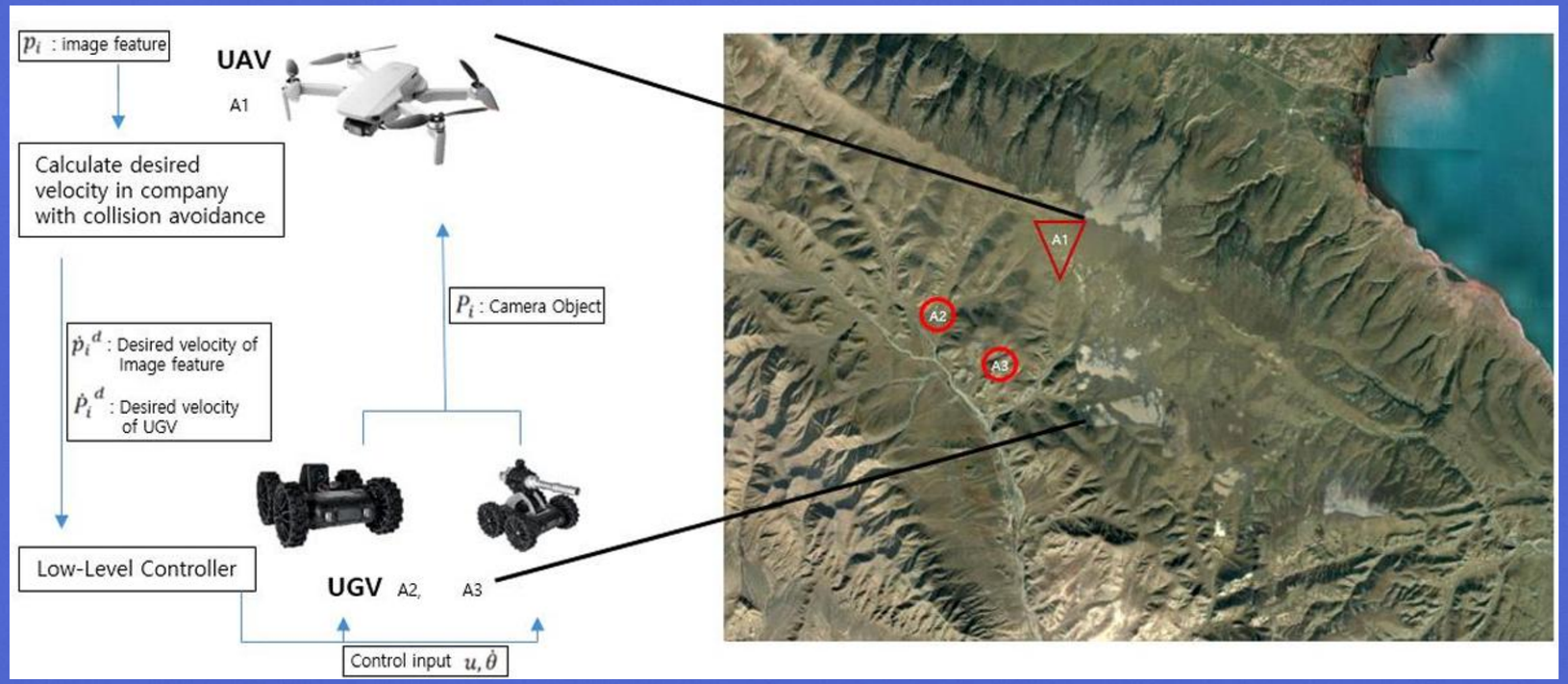
SITL 기반 소형 자율 임무 수행 로봇 개발 연구



팀명 : AURA
기계공학과 박정근 (팀장)
기계공학과 김석균 (팀원)
기계공학과 이태희 (팀원)
기계공학과 임찬혁 (팀원)

INDEX

문제 정의

하드웨어 중심 개발의 한계

✓ 막대한 비용	» 초기 구축 비용의 부담 » 지속적인 운영 및 유지비
✓ 파손 리스크	» 기기 파손 및 고가 센서 손실 » 안전사고 및 협업 장애
✓ 느린 학습 속도	» 실시간 물리 환경의 제약 » 실험 준비 및 리셋 시간의 낭비

문제 정의

프로젝트의 목표

AI 자율 임무 수행

로봇이 스스로 상황을 판단하고
임무를 완수하는 시스템 구현

SITL 기반 사전 검증

실제 로봇 없이 가상 환경에
AI를 완벽히 학습/검증 후 이식

이기종 협력 운용

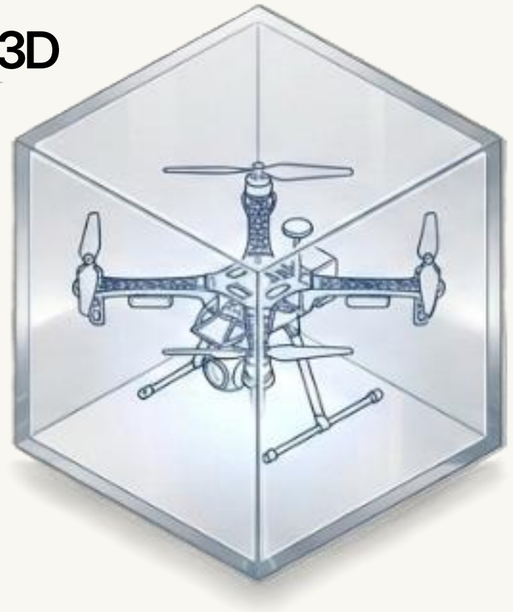
드론과 로버가 서로 데이터를
교환하며 하나의 임무 수행

SITL이란?

SITL (Simulation in the Loop)

: 실제 로봇 없이 가상 환경에서 AI를 무한 반복 학습 및 검증 후 실제 기체에 이식하는 기술

Unity3D



수천 번의 가상 임무 반복

검증된 AI 실기체 완벽 탑제

SITL이란?

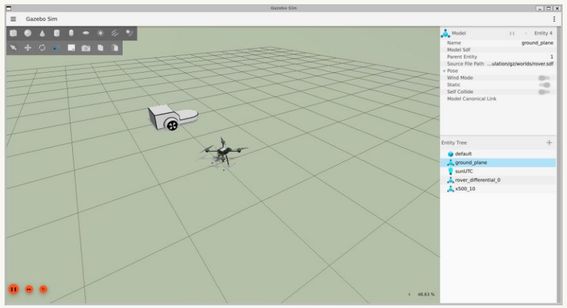
SITL의 이점

✓ 파손 리스크 제로	» 가상 환경에서 무한 반복 테스트 가능
✓ 반복 검증 가능	» 빠른 AI 학습 사이클 확보 환경 자유 설정
✓ 다중 기체 동시 테스트	» 드론/로버 Multi-Agent 안전 검증

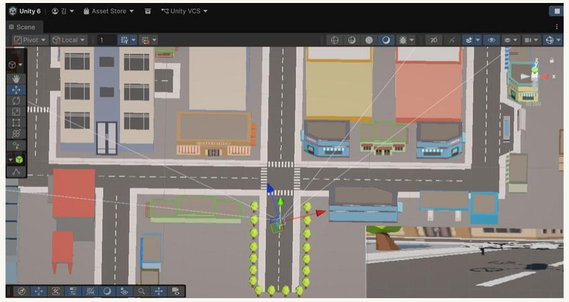
SITL이란?

SITL 시스템 구조 : 3계층 파이프라인

지상통제소 (GCS)
사용자가 임무를 부여하고 드론/로버의 상태를 실시간 모니터링하는 화면



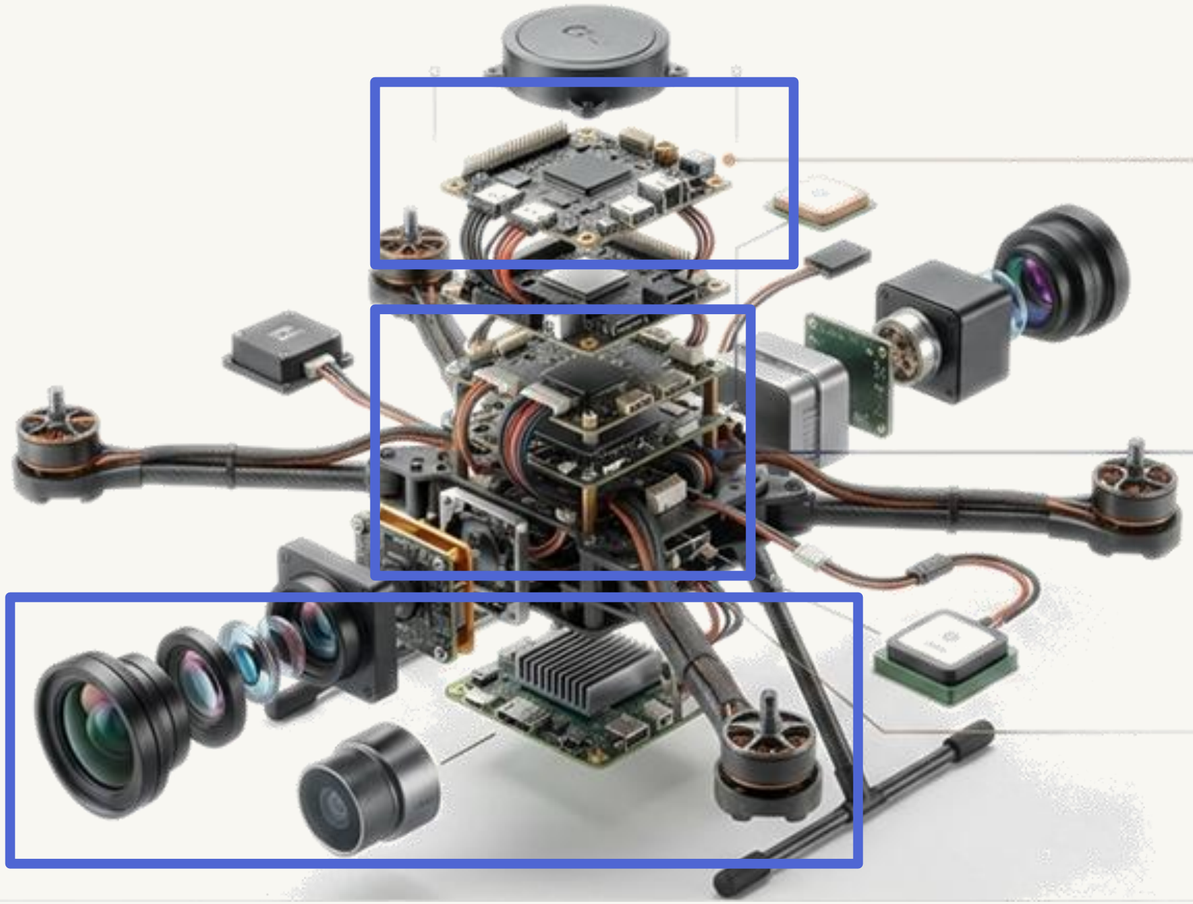
비행제어 (FCC) : PixHawk Firmware(PX4)
드론/로버의 모터/서보를 제어하는 핵심 Firmware



디지털 트윈 엔진 : Unity3D ⇔ AI Module (CNN / PPO)
가상 환경에서 LiDAR/Camera 센서 데이터를 생성하고 AI가 경로를 인지 후 장애물 회피

임무 시나리오

자율 주행의 원리



라즈베리파이
: 센서 데이터를 기반으로 AI 연산 및 최적의 경로 판단

PixHawk
: 명령을 받아 모터와 서보의 움직임 제어

센서
: 주변 환경 스캔 후 장애물 인식

임무 시나리오

전체적인 임무 시나리오 개요

1. 광역 스캔 : 상공에서 장애물 파악 후 Waypoint 데이터 생성

2. 데이터 전송 : 드론에서 로버로 Waypoint 데이터 실시간 전송

MAVLink

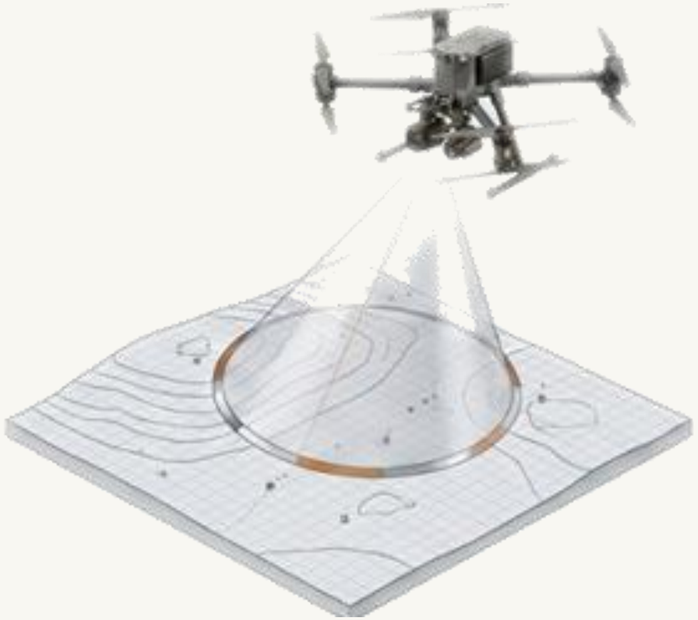
MAVLink

3. 자율주행 : Waypoint 데이터 기반 최적 경로 자율 주행

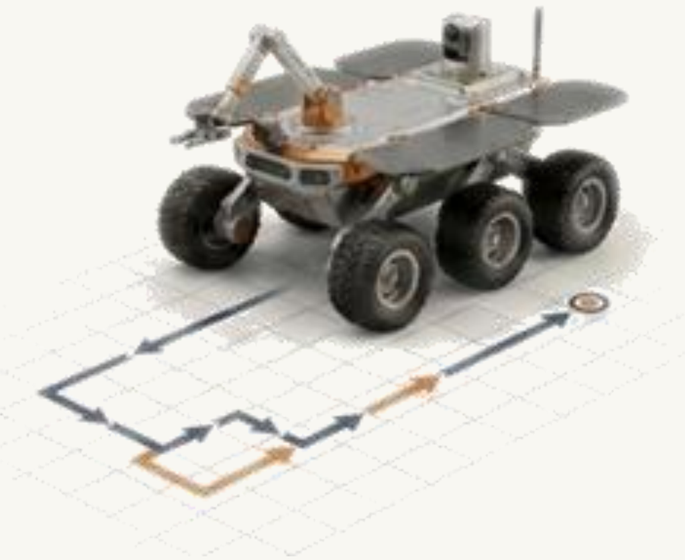
목표 도달 : 자체 카메라로 미인식 장애물 실시간 회피

임무 시나리오

드론과 로버의 임무 시나리오



Drone : 3D 공간 제어
Waypoint 생성 후 로버에 전송
광역 지형 및 장애물 스캔



Rover : 2D 평면 제어
Waypoint 기반 경로 계획
미탐지 장애물 실시간 회피

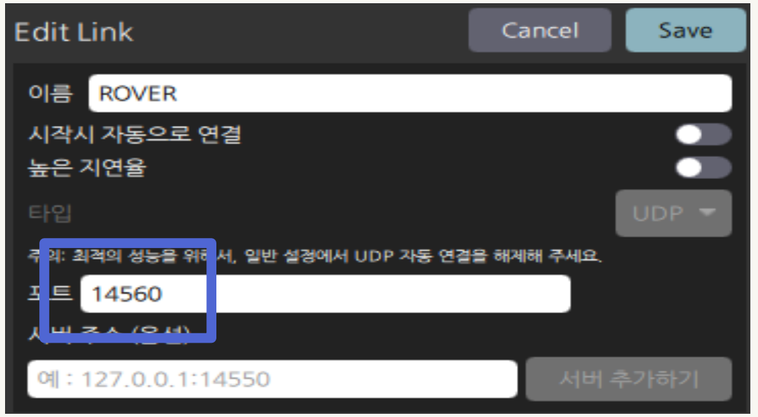
Unity3D 환경 구축

Unity3D, PX4 연동 완료

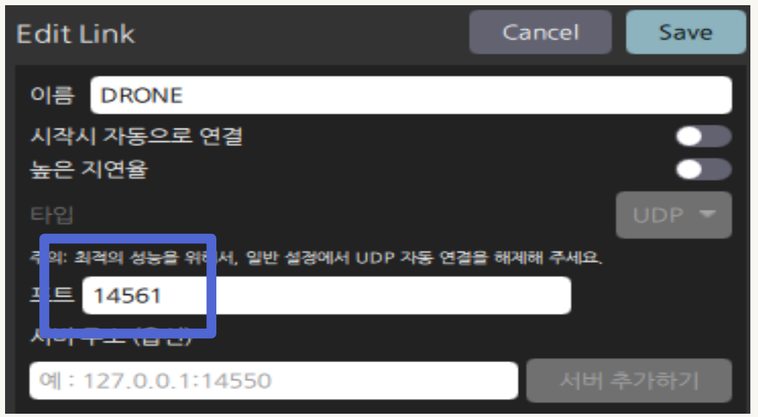
```
taehee@DESKTOP-B31C41M:~$ mavproxy.py --master=udp:
4560
Connect udp:127.0.0.1:14541 source_system=255
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from 127.0.0.1:14541
MAV> GPS lock at 0 meters
Detected vehicle 2:1 on link 0
online system 2
LOITER> Mode LOITER
fence breach
AP: GCS connection regained
waypoint 9
Received 825 parameters
Saved 826 parameters to mav_2_1.parm
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
Got COMMAND_ACK: REQUEST_MESSAGE: ACCEPTED
```

드론, 로버 충돌 방지를 위한 포트 분리

로버



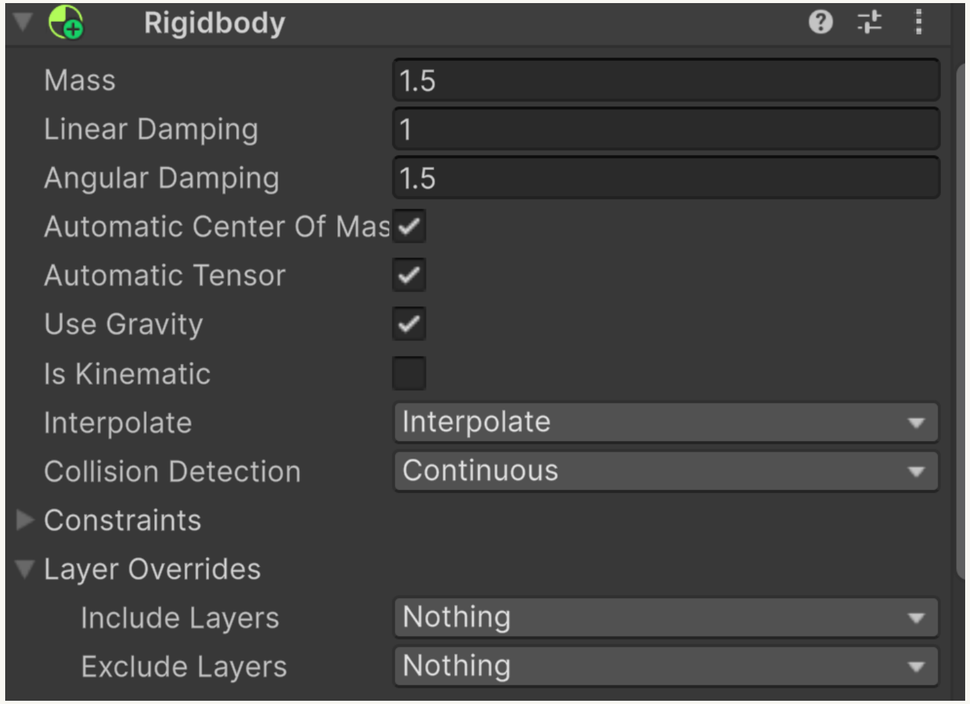
드론



Unity3D 환경 구축

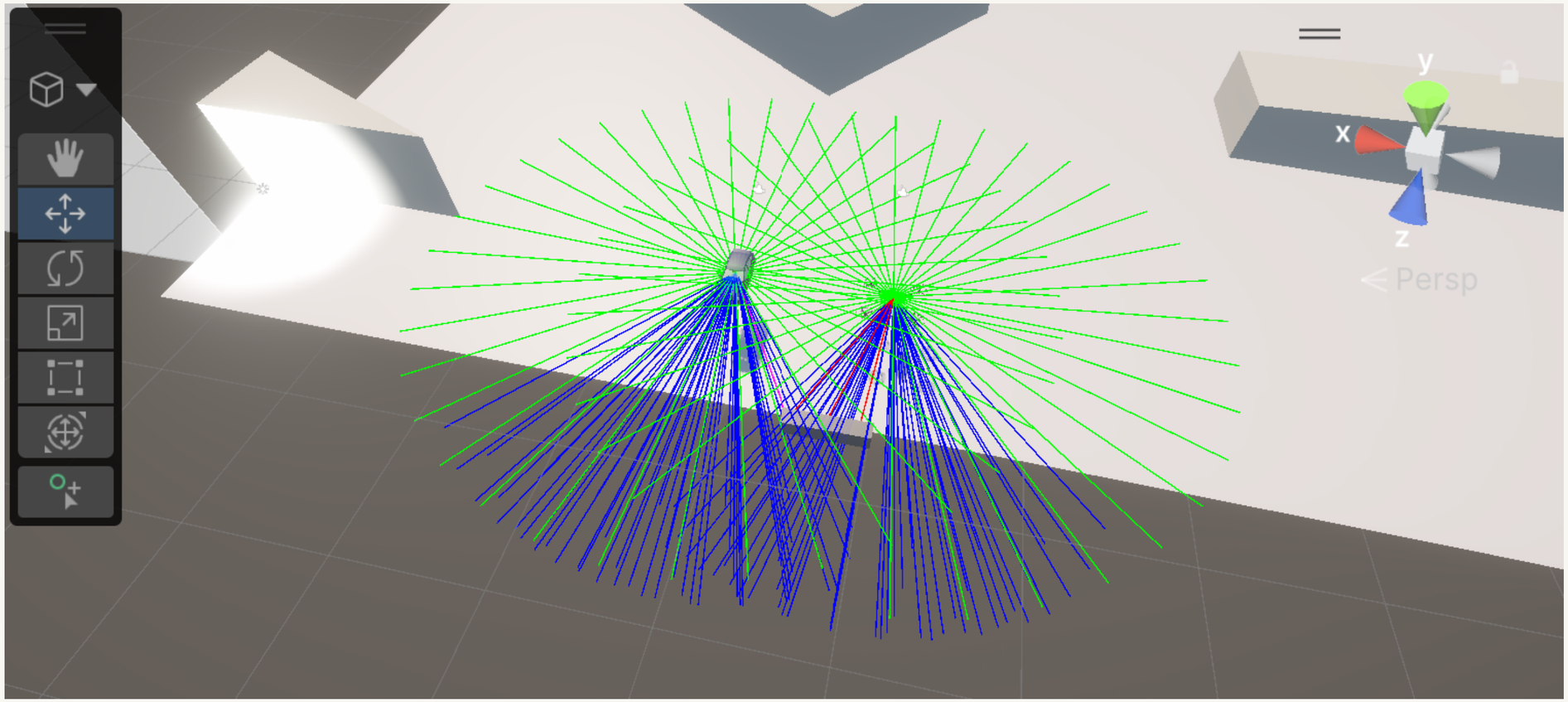
Unity3D 상 환경 구축

시뮬레이터/구조물 물리 엔진 최적화



드론/로버 학습 및 시뮬레이션 검증

드론/로버 센서 부착 – LiDAR, 3D Depth Camera

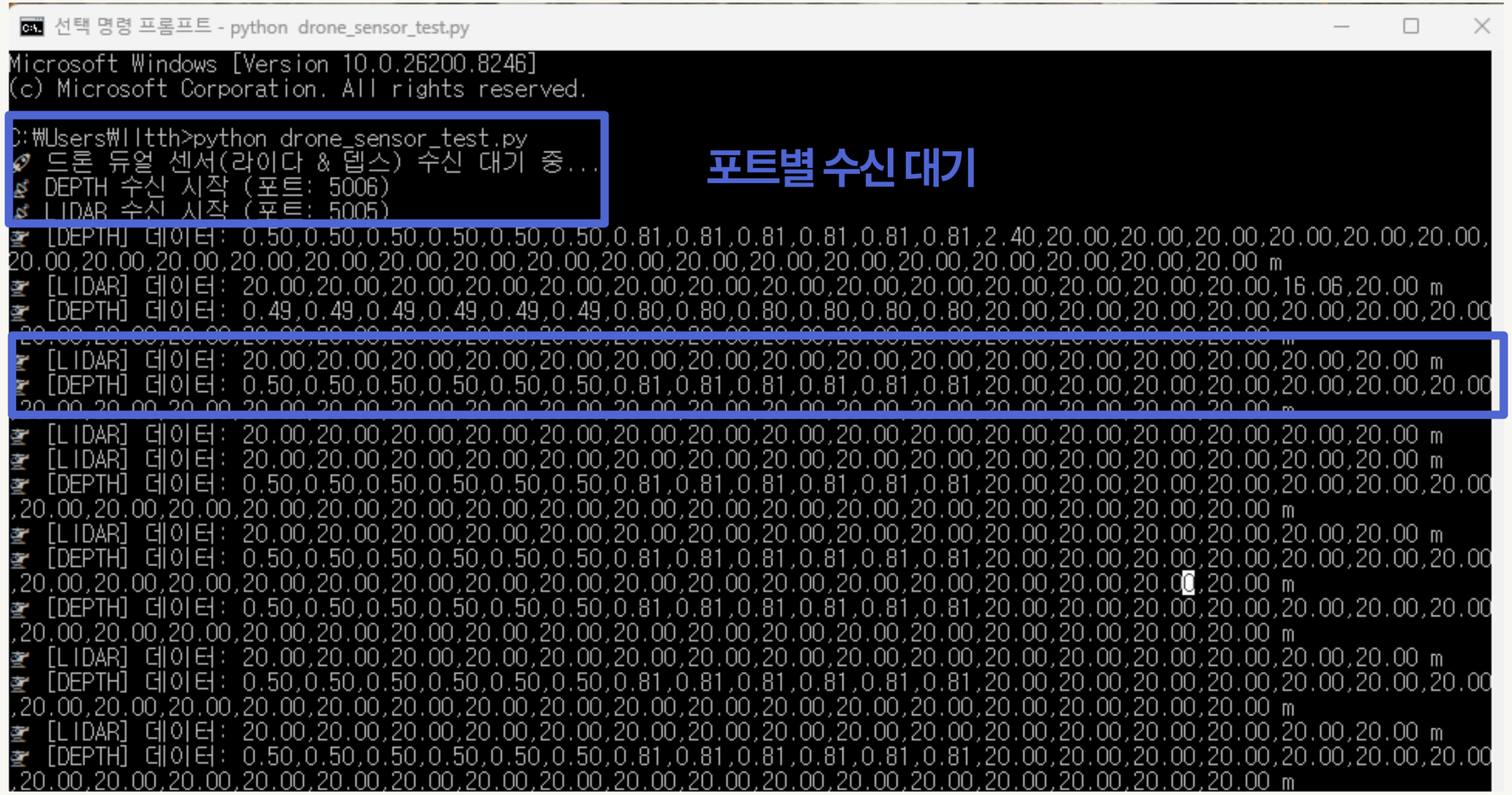


초록색 : LiDAR 센서

파란색 : 3D Depth Camera

드론/로버 학습 및 시뮬레이션 검증

LiDAR 센서, 3D Depth Camera 데이터 PX4에 송신 완료

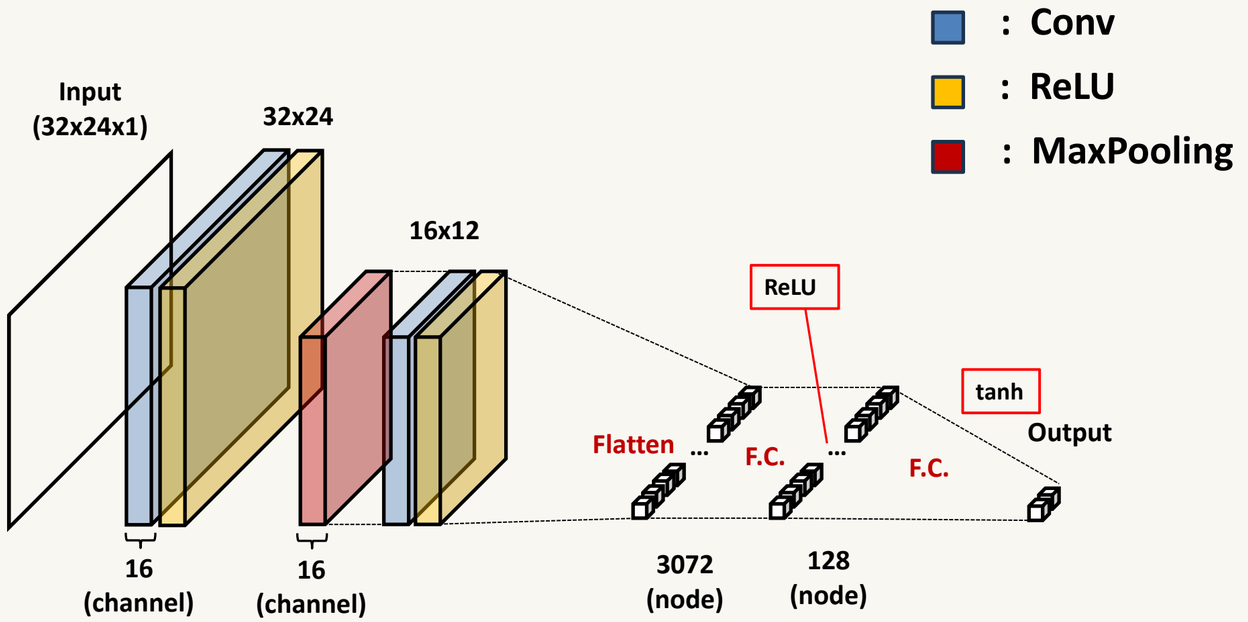


드론/로버 학습 및 시뮬레이션 검증

Unity3D에 CNN 코드 입력

```python
class PPOActorCriticCNN(nn.Module):
    def __init__(self, cam_shape, action_dim=2):
        super(PPOActorCriticCNN, self).__init__()

        self.cnn = nn.Sequential(
            nn.Conv2d(cam_shape[0], 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Flatten()
        )

        with torch.no_grad():
            dummy_input = torch.zeros(1, *cam_shape)
            cnn_out_dim = self.cnn(dummy_input).shape[1]

        self.actor = nn.Sequential(
            nn.Linear(cnn_out_dim, 128),
            nn.Tanh(),
            nn.Linear(128, 128),
            nn.Tanh(),
            nn.Linear(128, action_dim),
            nn.Tanh()
        )
        self.action_log_std = nn.Parameter(torch.zeros(1, action_dim))

        self.critic = nn.Sequential(
            nn.Linear(cnn_out_dim, 128),
            nn.Tanh(),
            nn.Linear(128, 128),
            nn.Tanh(),
            nn.Linear(128, 1)
        )
```

CNN 아키텍처



드론/로버 학습 및 시뮬레이션 검증

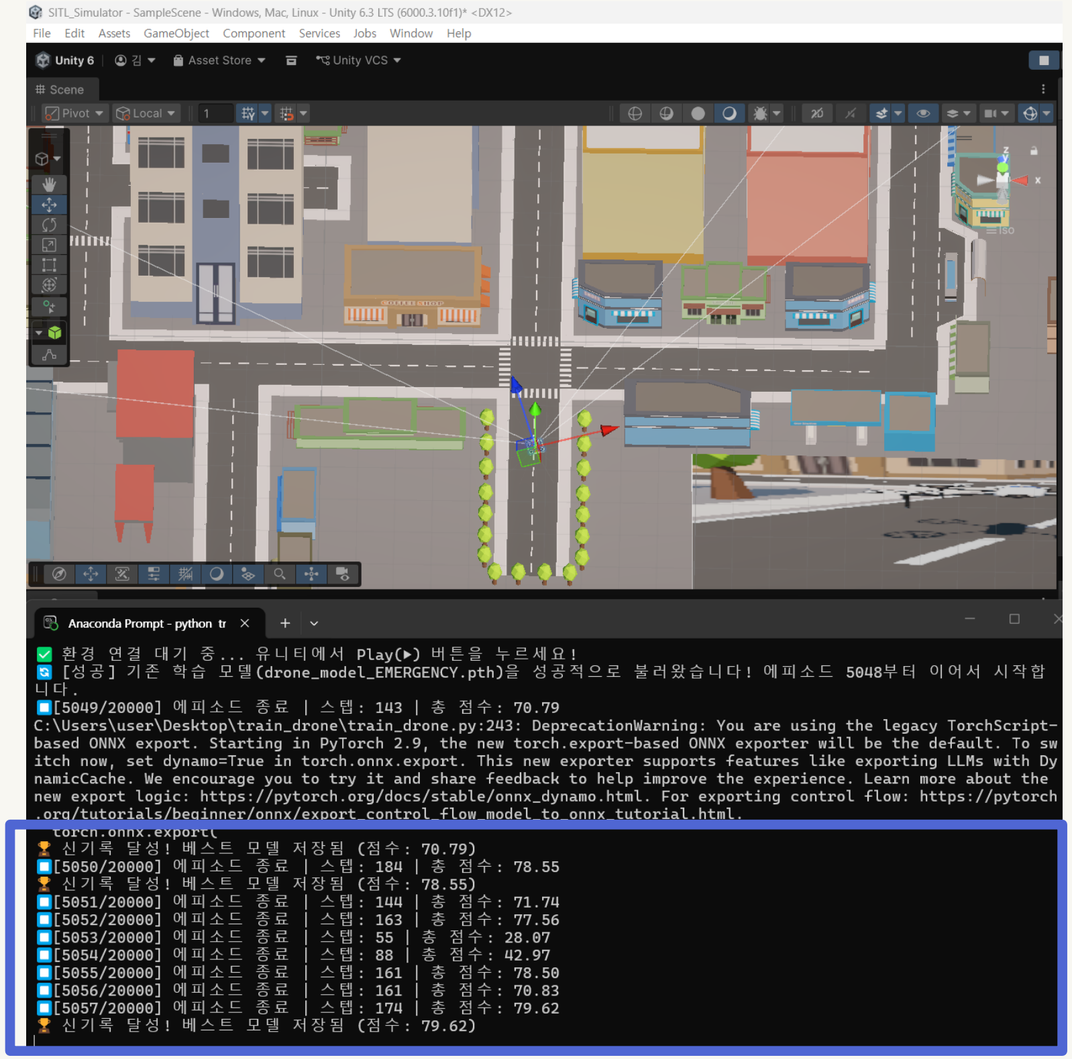
Unity3D에 보상 함수 입력

```
// ── 🎯 보상 및 다중 웨이포인트 로직 ──
if (waypoints != null && waypoints.Length > 0 && currentWaypointIndex < waypoints.Length)
{
    Transform targetWaypoint = waypoints[currentWaypointIndex];
    float currentDistance = Vector3.Distance(transform.position, targetWaypoint.position);
    Vector3 dirToTarget = (targetWaypoint.position - transform.position).normalized;

    // 실제 날아가는 궤적 방향이 타겟을 향하면 보상
    if (rb.linearVelocity.magnitude > 0.1f)
    {
        Vector3 currentVelocityDir = rb.linearVelocity.normalized;
        float velocityTowardsTarget = Vector3.Dot(currentVelocityDir, dirToTarget);
        AddReward(velocityTowardsTarget * 0.02f);
    }

    // 시선 보상은 방향을 잡아주는 보조용
    float lookAtTargetReward = Vector3.Dot(transform.forward, dirToTarget);
    AddReward(lookAtTargetReward * 0.005f);

    // 핵심: 거리 단축 보상 강화
    float distanceChange = previousDistance - currentDistance;
    AddReward(distanceChange * 1.0f);
    previousDistance = currentDistance;

    // 평형 유지 보상 (기체가 수평을 잘 잡을수록 가산점)
    float uprightReward = Vector3.Dot(transform.up, Vector3.up);
    AddReward(uprightReward * 0.002f);
```

```
// Exploitation 방지 1: 너무 심하게 뒤집어지면 아웃
if (uprightReward < 0.4f)
{
    SetReward(-2.0f);
    EndEpisode();
    return;
}

// Exploitation 방지 2: 바닥으로 추락하면 아웃
if (transform.localPosition.y < 0.2f)
{
    SetReward(-2.0f);
    EndEpisode();
    return;
}

// 도착 판정 (1.5m 이내 진입 시 다음 웨이포인트로)
if (currentDistance < 1.5f)
{
    AddReward(5.0f); // 중간 목적지 도달 보상
    currentWaypointIndex++;

    // 모든 웨이포인트 완주
    if (currentWaypointIndex >= waypoints.Length)
    {
        SetReward(10.0f); // 강력한 최종 성공 보상
        EndEpisode();
    }
    else
```

드론/로버 학습 및 시뮬레이션 검증

드론/로버 강화학습 시작

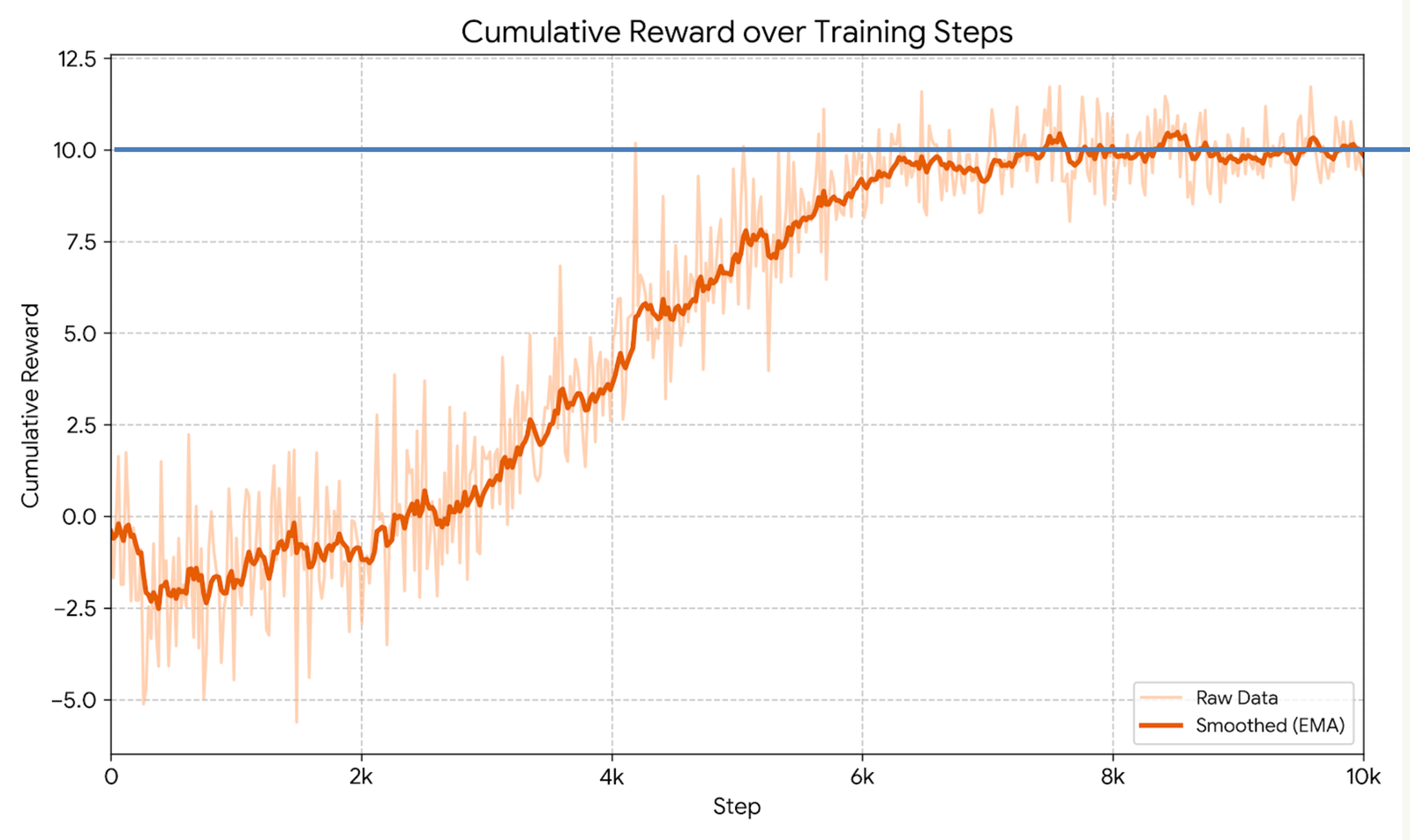


로버 강화학습 과정

베스트 모델 가중치 저장

드론/로버 학습 및 시뮬레이션 검증

로버 강화학습 누적 보상 그래프



Step이 진행될수록

누적 보상 10에 수렴

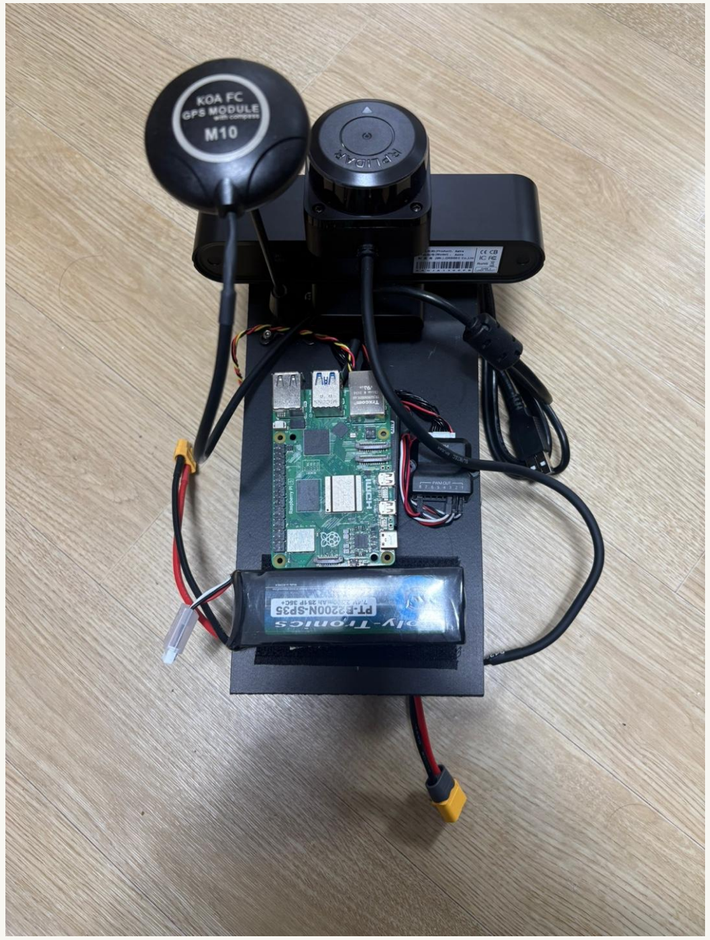
드론/로버 학습 및 시뮬레이션 검증

로버 Waypoint 지정

학습된 로버 시뮬레이션

향후 계획 – 실기체 이식

3MAX-24 로버 실기체



F450 드론 실기체

향후 계획 – 실기체 이식

실기체 HW 통합	드론/로버 협력 통합
1. Companion Computer 탑재	1. 드론 맵 데이터 로버에 전달
2. LiDAR, 3D Depth Camera 연결	2. 전체 임무 시나리오 실기체 시연
3. AI 모델 Edge Computing 이식	3. LLM 연동 자율 기능 고도화
4. 드론/로버 단독 자율주행 테스트	4. 최종 성과 정리 및 발표

감사합니다.