Capstone디자인A 결과 보고서

강화학습을 이용한 밸런싱 로봇 제어 시스템

팀명: 독수리
지능기전공학부 스마트기기공학전공
17011832 김남훈

1. 서론

밸런싱 로봇은 이륜 역진자(Two Wheel Inverted Pendulum, TWIP) 모델이라고도 불리우며, 로봇이 넘어지지 않고 영점에 가까운 균형을 유지하는 것을 목표로 하는 로봇이다. 세그웨이(Segway) 등 이 모델을 활용하는 산업에서는 이를 제어하기 위해 우수한 성능이 입증된, 가장 보편적이고 대중적인 PID 제어 기법을 애용한다.

그러나 PID 제어는 비선형 모델을 대상으로 할 때 이득 값 최적화에 많은 시간이 요구되고, 최적화 후에도 비선형성 특유의 시스템 불안정성은 온전히 해소되기 어렵다. 또한, 제어 파라미터로 입력되는 IMU 센서 데이터는 자이로의 경우 전처리 알고리즘에서 적분 오차가 누적되어 개발자가 임의로 초기화를 해주어야 하며, 가속도는 물리적인 충격에 과민반응 하는 등 정상적인 동작을 완벽히 보증할 수 없다.

따라서 본 프로젝트에서는 강화학습을 이용하여 PID 제어기의 이득 값을 획득하고, 가속도 센서를 카메라로 대체한 후 IMU 센서와 융합하여 급격한 센싱 변화에 강인한 자세 제어기를 설계한다. 설계 모델을 Simulink로 시뮬레이션 한 후, 결과를 기존 제어기법과 비교하여 성능 개선을 검증한다. 이를 실제 로봇에 탑재하여 실험하고 상태 변화 수치를 관측하기 위해 실시간 모니터링 시스템을 웹과 앱으로 구현한다.

2. 제어 시스템

2.1. 밸런싱 로봇

밸런싱 로봇은 추후 학습을 위해 입력은 모터에 인가될 전압, 출력은 기울기 각도인 pitch 값으로 설정하였다. 로봇에 가해지는 수평방향의 힘을 U, 바퀴와 본체의 질량을 각각 M, m, 마찰계수를 b, 관성 모멘트를 I, 무게중심의 높이를 l 이라고 한다면 다음과 같이 나타낼 수 있다.

$$\frac{\Theta(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 + \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \left[\frac{rad}{N}\right] \quad (1)$$

2.2. 자세 값 융합

로봇이 넘어지지 않도록 하기 위해서는 기울기 각도가 일정 범위를 벗어나지 않도록 유지해야 한다. 본 모델에서는 로봇이 좌우 방향 전환이 일어나지 않는 환경에서 각도만 유지하

는 것을 목표로 하였다. 따라서 IMU 센서의 자이로스코프 pitch 값과 카메라에서 유도한 pitch 값을 융합하여 사용한다.

카메라로 촬영한 영상 정보만을 이용하여 pitch 값을 획득하기 위해서는 로봇이 올바르게 서있는 것에 대한 기준점과, 그로부터 얼마나 벗어나 있는지를 계산할 수 있는 척도가 필요하다. 이에 본 프로젝트에서는 객체 인식 방법과 arctan을 활용하였다. 최초 구동 시 로봇의 시작 위치를 고정된 거리(w)에 두고 녹색 원을 기준점으로 삼은 후, 이를 컨투어(contour)로 인식하여 원의 중심을 카메라 상의 좌표로 지정한다. 이후 로봇이 움직이면 카메라의 중심 좌표가 변하는데, 이 때 원의 중심과 카메라 상의 중심 간 거리(h)를 측정한다. 로봇, 원이 붙어있는 벽, 카메라 업 벡터를 삼각형 구조로 보면 로봇의 기울어진 정도 θ는 $\theta = \arctan(h/w)$를 계산하여 얻을 수 있다. 이를 그림으로 나타내면 다음과 같다.

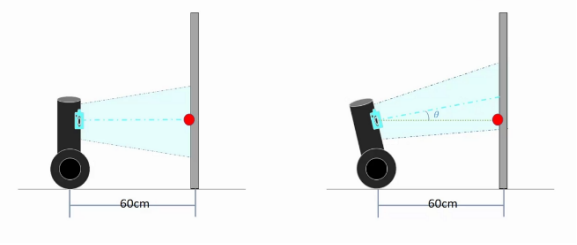


Figure 1 카메라를 이용한 pitch 값 획득

이를 통해 획득한 각도를 센서를 통해 측정한 값과 비교했을 때, 각도가 커짐에 따라 오차가 증가하는 문제가 발생함을 확인하였다. 이에 다항식 보간법을 이용하여 카메라 각도의 스케일을 조정함으로써 해결하였다.

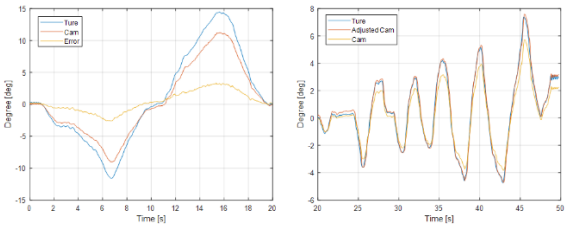


Figure 2 (좌) 다항식 보간 적용 전 (우) 적용 후

그러나 근본적으로 영상 정보는 2차원의 정수형 데이터 수만 개로 이루어져 있으므로 이를 읽고 처리하는 시간을 단축시키는 어렵다. 이로 인해 IMU 센서로부터 들어오는 값과 주기가 맞지 않아 동기화가 되지 않는다. 따라서 rate transition을 위해 상보 필터를 적용하여 카메라 데이터가 들어올 때만 자세 데이터 융합을 진행하였다. 상보 필터는 저주파 통과 필터와 고주파 통과 필터로 구성되어 있으며, 전자는 주기가 긴 카메라를, 후자는 주기가 짧은 IMU 센서를 통과시켜 각도를 융합한다. 결과는 아래와 같다.

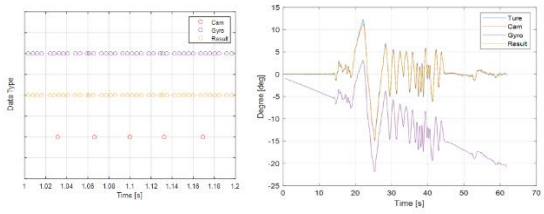


Figure 3 (좌) 각 데이터의 입력 주기 (우) 상보 필터 적용 후

2.3. PID

PID 제어는 오차 $e(t)$, 오차의 적분(Integral) 미분(Differential) 값에 비례(Proportional)하는 제어 값 $u(t)$를 계산하는 방법이다. 피드백 제어 구조이며, 비례 제어는 목표 제어 값에 도

달하는 시간을 단축시키고, 적분 제어는 정상 상태의 오차를 줄이며, D제어는 외란을 억제한다. 이 때 각 항의 이득 값 (K_p, K_I, K_D)을 조정하는 것을 PID 튜닝이라고 하며, 본 프로젝트에서는 이를 강화학습을 통해 진행한다. 이를 수식으로 나타내면 다음과 같다.

$$u(t) = K_p e(t) + K_I \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad (2)$$

3. 강화학습 기반 적응형 PID 제어

로봇의 제어에 강화학습을 적용하는 방법으로는 크게 두 가지가 있다. 하나는 제어기 자체를 강화학습 모델로 대체하는 것이고, 다른 하나는 기존의 제어기에 강화학습을 보조하는 방법이다. 전자의 경우로, Rahman[1]은 Q-learning과 DQN(Deep Q Learning) 알고리즘을 밸런싱 로봇에 적용하여 PID, fuzzy, LQR 제어와 비교하고 준수한 성능을 보였다. 후자의 경우, PID 제어의 일종인 적응형 PID(Adaptive PID) 제어에서 파생되어, Sun[2]은 A3C 알고리즘을 adaptive PID 제어기에 적용하여 다른 제어 방법론에 비해 빠른 학습 수렴과 강한 적응성을 보였다.

본 프로젝트에서는 후자의 방법을 채택하고, DDPG(Deep Determinant Policy Gradient)[3] 알고리즘을 이용하였다. DDPG는 DPG에 DQN의 심층 신경망 등의 성공적인 아이디어를 접목한 actor-critic 알고리즘이다. 이산적인 동작 공간(Action Space)을 가진 기존의 DQN과 달리 연속적이므로 실제 세계에서 동작하는 밸런싱 로봇을 학습하기에 적합하다.

강화 학습은 크게 환경(environment), 에이전트(agent), 정책(policy), 관찰 값(observation), 동작(action), 보상(reward)으로 구성된다. 에이전트는 환경에서 Q-function을 이용하여 현재 관찰 값을 고려하였을 때, 가장 적절한 정책에 따라 동작을 취한다. 그 결과로 관찰 값이 변화하며 보상을 반환한다. 모델은 이 보상을 최대한 크게 받을 수 있는 가장 최적화된 정책을 찾기 위해 학습을 반복하게 된다. 이를 지정된 에피소드 횟수만큼 학습을 진행하는데, 진행 도중 종료 조건을 만족하게 되면 해당 에피소드를 종료한다. PID 제어 학습에 이를 적용하면 환경, 관찰 값, 동작은 각각 밸런싱 로봇, 기울기 정도, P, I, D 이득 값에 대응된다. 각도는 $\pm\pi$, 위치는 $(-\infty, \infty)$ 의 범위를 가진다. 로봇의 각도가 $\pm 10°$를 벗어날 경우 복원력을 상실하였다고 판단, 학습을 종료하도록 설계하였다. 지정된 시뮬레이션 시간 이상 유지에 성공한 경우도 종료하도록 하였다.

보상은 로봇의 각도를 중심으로 설계하였다. 우선 종료 조건인, 로봇의 각도가 $\pm 10°$를 벗어날 경우 무거운 패널티를 부여한다. 동시에 0도에 가깝게 유지하지 못할 경우, 멀어진 정도에 비례하여 패널티를 부여한다. 추가로 로봇이 초기 위치를 이탈하지 않고 최소한의 제어만으로 자세를 유지하도록 유도하기 위해서 이동 거리가 작을수록 큰 보상을 부여한다. 전체 보상 모델은 다음과 같다.

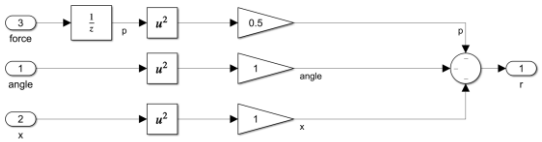


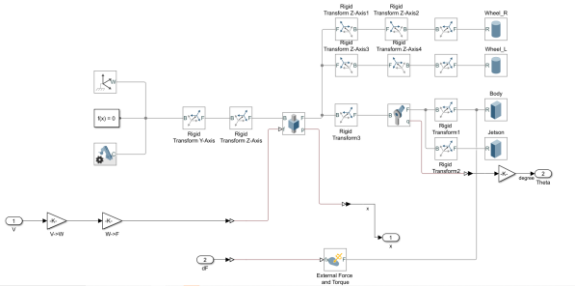
Figure 4 보상 모델

에이전트는 주어진 관찰 값을 기반으로 적절한 동작을 취해 환경과 상호작용하여 상태를 변화시킨다. 본 프로젝트에서는 DDPG 알고리즘을 사용하는데, 이는 Actor 네트워크와 Critic 네트워크로 구성되어 있다. 전자는 정책 함수

를 학습하며, 후자는 해당 정책의 가치(Value)를 평가한다. 이를 통해 에이전트가 최적의 정책을 수립하고 동작할 수 있도록 한다. 에이전트의 동작으로 ±5e+03 범위의 P, I, D 이득 값 K_p, K_I, K_d이 선택된다.

3.1. 학습 및 시뮬레이션

Simulink로 시뮬레이션을 진행하기 위해 Simscape Multibody를 이용하여 수식을 기반으로 단순화한 모델을 디자인하였다. 파츠는 양 쪽 바퀴, 본체, Jetson으로 이루어져 있으며, 바퀴와 본체는 조인트를 통해 본체와 결합되어 있다. 로봇은 지형과 조인트를 통해 연결되어 상호작용하며, 인가된 전압을 수식을 통해 처리하여 이동시킨다. 본체에 외란이 작용하며, 로봇의 위치와 기울기 각도가 출력된다. 모든 강체에는 자동으로 계산된 마찰력이 작용한다. 모델과 시스템 구조는 다음과 같다.



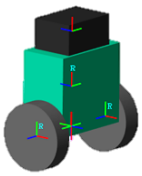


Figure 5 밸런싱 로봇 모델링

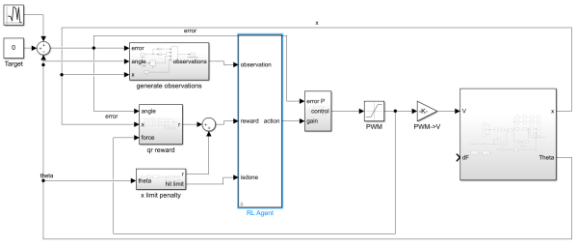


Figure 6 전체 시스템

학습에서 초기 각은 0도, 0.17N/s의 힘을 0.4초간 외부에서 로봇에 작용하였으며, 한 에피소드 당 20초 유지를 최대로 95번의 에피소드를 학습하였다. 학습 결과는 왼쪽과 같다. 녹색 선에 리워드가 근접할수록 학습이 잘 이루어졌다는 지표이다. 학습 에피소드를 700회로 증가시키고 다시 진행한 결과는 오른쪽과 같다. 주황색이 평균 보상값이며, 도달 가능한 최대 보상값에 수렴한 것을 확인할 수 있다.

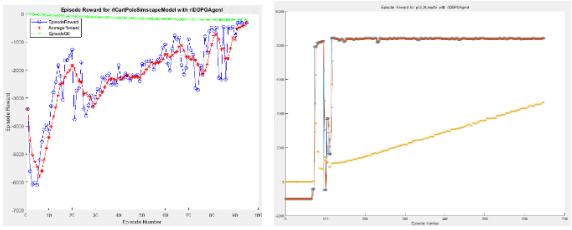
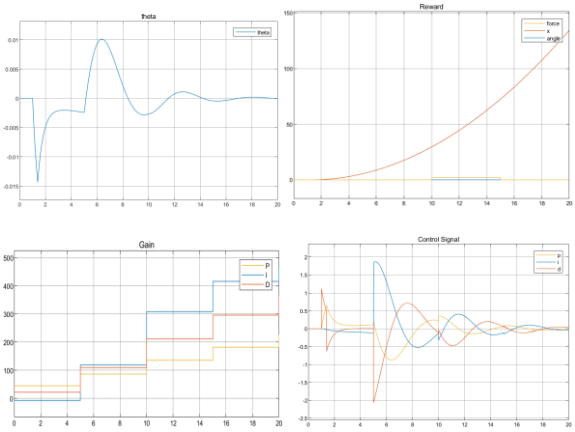


Figure 7 강화학습 에피소드 보상 그래프

학습된 이득 값을 바탕으로 시뮬레이션 모델을 테스트한 결과 성공적으로 자세 제어를 수행함을 확인하였다. 테스트 시 자세 오차는 없으며, 외란은 1N으로 설정하였다.



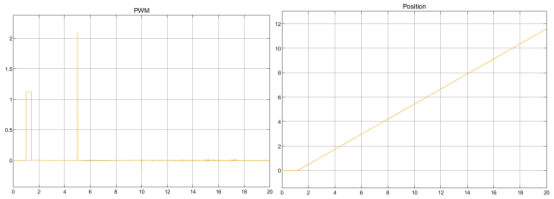


Figure 8 강화학습을 통해 얻은 이득 값을 이용한 테스트 결과.
(왼쪽부터 시계 방향으로 로봇의 각도, 보상, 이득 값, 제어 값, 모터 PWM, 로봇의 위치)

4. 실험: 구현

본 프로젝트의 실제 구현은 크게 아두이노(Arduino Uno), Jetson Nano, 모니터링(웹, 안드로이드 앱) 파트로 나누어 진행되었다. 시스템의 개략적인 동작 순서는 다음과 같다. Jetson에 부착된 카메라를 이용해 영상을 촬영하고, 영상으로부터 각도를 계산하여 아두이노에 전송한다. 아두이노는 IMU 센서로부터 획득한 데이터를 Jetson으로부터 전달받은 값과 융합하여 PID 제어기에 전달한다. 제어기는 강화학습으로 획득한 이득 값을 바탕으로 모터를 제어한다. 동시에 카메라, 자이로, 융합 값 3개를 앱으로 전송하고, 앱에서 그래프로 보여준 후 웹 서버에도 전송한다. 웹 서버는 그래프와 로봇의 기울어진 정도를 시각화하고, 동시에 Jetson으로부터 흑백 축소한 영상을 전송받아 보여준다. 실제 구현 시에는 모델 및 구현 난이도의 경량화를 위해 적응형 PID가 아닌 마지막으로 도출된 이득 값을 대입하여 진행하였으며, 로봇은 벽으로부터 60cm 떨어진 곳으로 위치하였다. 기존 방법과 대조하기 위해 가속도계를 이용한 실험도 함께 진행하였다. 실험 결과는 아래와 같다.

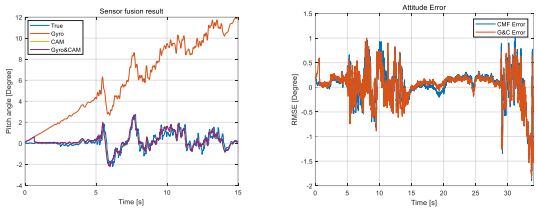


Figure 9 실험 결과 (좌) 센서 융합 (우) 자세 오차 (CMF: W/O Cam, G&C: W/Cam)

CMF	G&C
0.6605	**0.3287**

표 1 카메라 사용 유무에 따른 RMSE(Root Mean Square Error) 비교

실험 결과를 RMSE를 이용하여 정성적 평가를 하였을 때, 단순히 IMU 센서만을 이용했을 때보다 안정적인 자세를 유지함을 확인하였다.

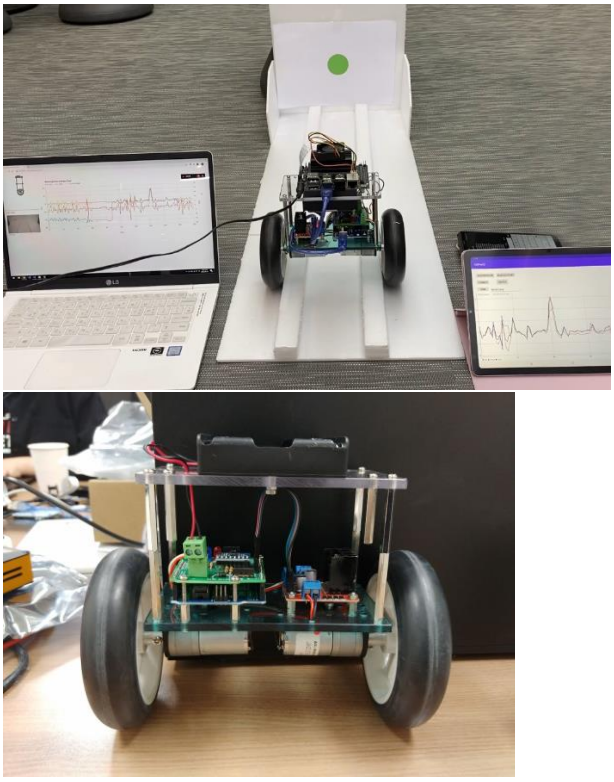


Figure 10 실제 로봇 모델

5. 모니터링 시스템

본 프로젝트는 실시간으로 데이터를 처리하

기 때문에 이를 지연 없이 처리하는 것이 매우 중요하다. 따라서 모니터링 또한 실시간으로 진행되어야 한다. 만약 Jetson에 모니터링 웹 서버를 설치한다면 아두이노와 USB 시리얼 통신을 통해 데이터를 제공받아야 하는데, 이미 Jetson에서 아두이노로 카메라 데이터를 전송하고 있으므로 duplex 통신의 특성상 지연이 늘어나게 된다. 게다가 아두이노 우노 특성상 연결 가능한 핀 수가 부족하고, 모터 드라이버 등 여러 부속이 이미 점유하고 있어 이더넷 드라이버 확장 또한 쉽지 않다. 또한 웹 서버에서도 로그 형태로 데이터를 저장 후 클라이언트에 제공하는 형태는 지속적인 파일 I/O가 발생함에 따라 버퍼 스트림의 활용도가 떨어지고 시스템의 부하를 초래하며, 클라이언트 또한 초 단위로 페이지 리로드가 발생하여 모니터링이 원활하게 진행될 수 없다. 값을 디스플레이할 때에도 각각의 값을 별개의 그래프에 띄우게 된다면 리소스 점유율이 증가하게 된다. 이에 본 프로젝트에서는 지연을 발생시키는 문제들을 최적화하기 위해 여러 가지 대안을 제시한다.

5.1. Jetson Nano

카메라에서 추정한 각도 값은 [-180, 180] 범위의 실수인데, 이를 아두이노에서 수신하면 별도의 처리를 해주는 메소드를 거쳐야만 한다. 그러나 이는 약 1초정도의 지연이 발생하게 된다. 따라서 연산량 감축을 위해 Jetson에서 계산된 각도 값을 100배 업스케일링 하여 8비트 이내의 정수부로 변환한 후 전송하고, 아두이노에서 이를 다운스케일링 하여 지연 없이 처리한다.

획득한 영상 정보를 전송할 때 흔들림 정도와 중심 이탈 거리만 모니터링 하는 것을 목표로 하므로 영상을 흑백으로 전환하고 해상도를 20% 비율로 최소화한 후 클라이언트에 전송한다. 이를 iframe 구조로 그래프 페이지에 보여주기 위해 Jetson에 카메라 서버와 클라이언트를 구동하고, 외부에서 localhost로 접속하도록 설정하여 영상 정보에 접근할 수 있도록 한다.

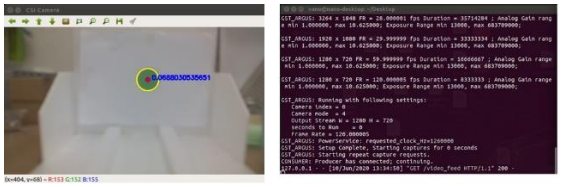


Figure 11 Jetson Nano에서의 구현 화면. (좌) 카메라 각도 계산 결과 (우) 아두이노 간 통신

5.2. 앱

앱은 카메라, 자이로, 융합각 세 개를 보여주는 그래프와, 블루투스 페어링, 서버 연결 기능으로 구성되어 있다. 아두이노에서 블루투스를 이용하여 페어링 상태로 대기중인 기기로 데이터를 전송하면, 앱은 버퍼를 통해 데이터를 읽는다. 이 때 여러가지 이유로 인해 데이터 스트림이 끊기거나 일부 패킷이 손실된 채 전송되는데, 이를 정제하기 위해 한 번에 여러 회차의 데이터를 전송받으며, 첫 회차는 절삭하고 그 다음 데이터를 사용한다. 동시에 데이터 손실이 확인되었을 경우 이전 출력 데이터로 대체하여 보간한다. 그래프를 출력한 후 소켓을 이용해 입력된 서버로 데이터를 전송하는데, 이는 시리얼 통신 충돌 문제를 해결하기 위해 웹 서버로의 데이터 전송을 앱을 이용하여 우회한다.

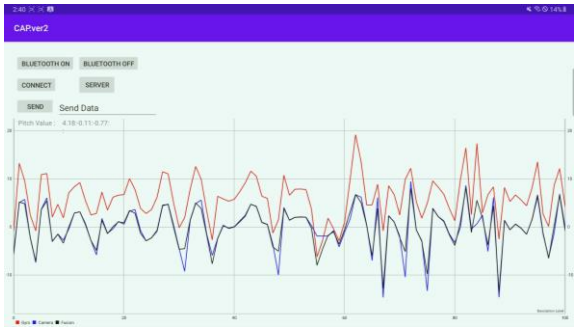


Figure 12 앱 구동 화면

5.3. 웹

웹 페이지는 앱의 기능에 카메라 화면 뷰를 추가한 구성이다. Node.js의 Serial 모듈을 이용하여 시리얼 버퍼의 데이터를 곧바로 서버로 전달받아, 큐(Queue)를 이용하여 새 데이터가 들어올 때마다 그래프를 처리한다. 동시에 로봇의 기울어짐을 직관적으로 확인하기 위해 그래프와 함께 로봇 이미지를 회전시켜 보여줌으로써 모니터링 시 편의성을 높였다. 카메라는 Jetson의 Python Flask 서버에 접근하여 페이지를 불러온 후 이를 하단의 iframe에 디스플레이한다.

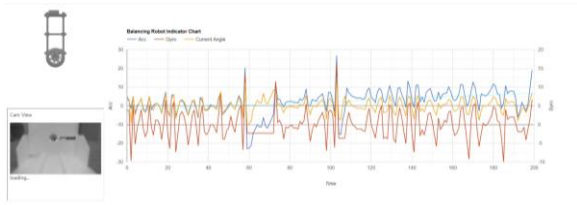


Figure 13 웹 구동 화면

6. 결론 및 개선사항

본 프로젝트를 통해, 강화학습을 이용하여 획득한 PID 이득 값이 실제 모델 적용에도 준수한 성능을 보임을 확인하였고, 단기간 내에 간편한 방법으로 획득함으로써 추후 모델 변경 등의 이유로 재학습이 필요할 시에도 용이하게 이득 값 획득이 가능할 것으로 예상된다. 또한 카메라와 자이로계를 융합한 자세 추정 기법은 가속도계 기반 추정보다 더 적은 오차율을 보이며 성능 개선에 성공하였다. 모니터링 시 다수의 장치와 실시간으로 통신할 때 직렬 구조로 진행하여도 치명적인 지연은 발생하지 않아 추후 제한된 실험 환경에서 유사한 문제가 발생할 시 대처 가능한 방법을 수립하였다. 추후 카메라를 응용하여, 컴퓨터 비전과 딥 러닝을 이용해 단일 영상으로 깊이를 획득하고, 장애물 인식 및 회피 기능도 수행 가능할 것으로 기대된다.

몇 가지 개선사항 또한 존재한다. 자세추정 시 로봇의 이동에 따른 거리 값 변화가 발생하게 되는데, 이 때 기준원 대비 넓이 변화를 계산하여 거리 값을 조정하면 보다 정밀한 각도 값 추정이 가능할 것으로 기대된다.

DDPG 이외에도 TD3, A3C, PRO 등 다양한 강화학습 알고리즘이 존재한다. 각 알고리즘별로 학습을 진행하여 수렴 속도를 비교하고 제어 성능을 비교하는 등 정성적, 정량적 평가를 후속 연구로 진행할 수 있다.

MATLAB 패키지와 관련한 애로사항도 남아있다. 완벽한 적응형 PID를 구현하기 위해서는 MATLAB에서 학습한 적응형 PID 추정 에이전트를 아두이노에 탑재해야 한다. 그러나 MATLAB Reinforcement Learning Toolbox에서는 아직 MATLAB Compiler를 통한 C언어 변환 및 패키지 배포를 지원하지 않고 있다. 대신 Jetson Nano에 직접 MATLAB을 세팅하는 툴박스가 있지만, 이는 GPU Coder를 구매하지 않으면 이용할 수 없다. 따라서 학습을 완료한 에이전트를 ONNX(Open Neural Network Exchange)같은 호환성이 뛰어난 라이브러리로

변환 후, 임베디드 시스템에 탑재 가능한 Tensorflow Lite 모델로 불러와 적용하거나, 추후 배포 패키지 지원이 된다면 후속 연구를 진행할 수 있을 것으로 기대된다. 이에 관련하여 진행했던 내용을 부록으로 첨부한다.

부록

- 강화학습 기반 제어기 설계

앞서 로봇의 제어에 강화학습을 적용하는 방법 중 제어기 자체를 강화학습으로 설계하는 방법 또한 본 프로젝트에서 진행되었다. 환경은 로봇이 움직이는 공간이며, 관찰 값은 IMU 센서의 높이, 로봇의 속도, 자이로 3축, 각속도, 양쪽 바퀴의 회전각과 회전속도, 이전 토크로 총 15개이다. 동작은 바퀴의 토크 값 1개이고, 실제로는 MUX를 통해 양쪽 모두 제어한다. 에이전트와 종료 조건은 동일하다.

보상의 초점 또한 기존과 동일하게 각도에 맞춰 설계하였다. 차이점은, 각도 값의 제곱 비례항을 더 무겁게 설정하고, 기존 각도와의 차이값을 구하여 만약 줄어들었을 경우, 즉 더 올바르게 섰다면 보상을 주고, 반대라면 패널티를 부과하였다. 또한 기존 토크의 크기에 비례하여 패널티를 부과하여 움직임을 최소화하였다. 더불어 종료되지 않도록 에피소드 유지 시간에 비례하여 상수 값을 보상으로 부여하였다. 전체 보상 모델은 다음과 같다.

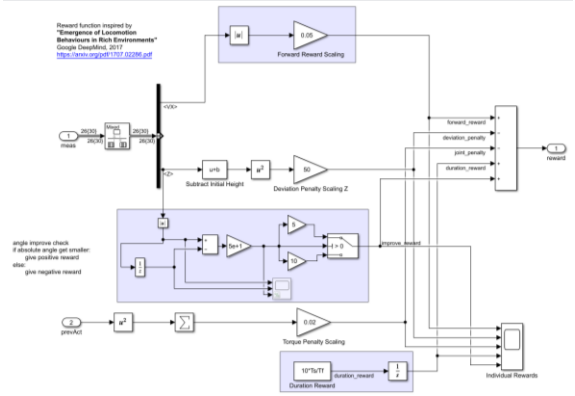


Figure 14 강화학습 기반 제어기의 보상 모델

로봇의 구조는 동일하지만 Simulink 모델링에는 차이가 있다. Bushing 조인트를 사용할 경우 시뮬레이션 도중 바퀴에서 짐벌락 현상이 발생하여 학습이 중단되는 치명적인 오류가 발생한다. 이를 해결하기 위해 조인트를 6-DOF로 교체하고 3축 카르테시안(cartesian)을 4축 쿼터니언(quaternion)으로 대체하여 문제를 해결하였다. 또한 본 모델은 ROS 범용 로봇 모델링 포맷인 URDF(Unified Robot Description Format)를 불러와 사용하였다. 이 과정에서 MATLAB은 Collision 속성을 지원해주지 않으므로 수직항력, 마찰력 등 접촉력을 설정해야 하는데, 이를 위해 Contact Force Library를 활용하여 지면과 바퀴, 본체 간 접촉부위에 속성을 추가하였다. 따라서 로봇이 평면을 투과하지 않도록 모델링하였다. 또한, 실제 로봇에 사용될 모터의 속성을 참고하여 구동 가능한 최대 토크값으로 변환 후 적용하였다. 전체적인 로봇과 전체 시스템 구조는 아래와 같다.

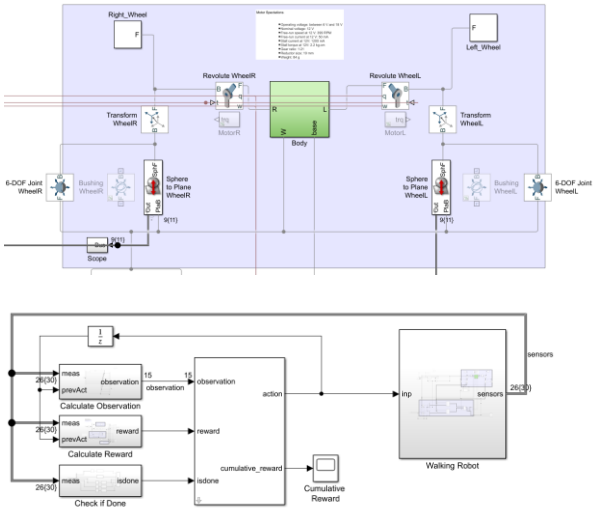


Figure 15 (상) 밸런싱 로봇 모델 (하) 전체 시스템

본 모델을 기반으로 2000회 에피소드에 대한 학습을 진행하였다.

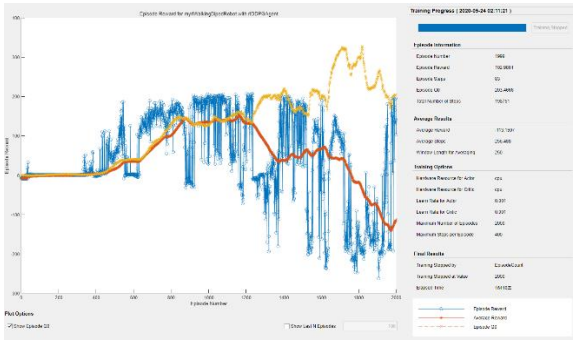


Figure 16 DDPG 알고리즘 강화학습 진행 그래프

그래프의 양상을 보면, 초기에는 전혀 학습이 이루어지지 않는 듯하다가 약 300회차 시도부터 급격히 학습이 진행되었다. 다만 본론의 결과와 달리 학습이 일정 경계 이상으로 진행되지 못하고 수렴하지 못하는 이유는 분석 결과 두 가지로 확인되었다. 우선 시뮬레이션 평면의 넓이가 충분하지 못해, 로봇이 추락하며 학습이 종료되는 문제가 있다. 두 번째로, 학습 최대 시간이 정해져 있어 이 이상 진행되지 못했다. 따라서 에이전트가 잘못된 방식이라 판단하고 다른 탐색 경로를 찾다가 local minima에 빠지며 진동하는 결과가 발생하였다. 본 문제는 위 두가지 문제를 해결함과 동시에 이전 학습 결과의 비중을 조절하는 형태로 파라미터를 수정하면 해결 가능할 것으로 예상된다. 학습된 에이전트를 이용하여 시뮬레이션한 결과는 다음과 같다. 첫 번째 토크 그래프는 강화학습에서 동작으로 출력하는 값으로, [-1,1]의 범위를 갖고, 이를 본 모델의 모터 스펙에 맞게 최대 토크값에 비례하여 조절하였다. 시계 방향 두번째는 누적 보상 그래프로서, 시간에 따라 보상이 선형적으로 증가하는 것을 확인할 수 있다.

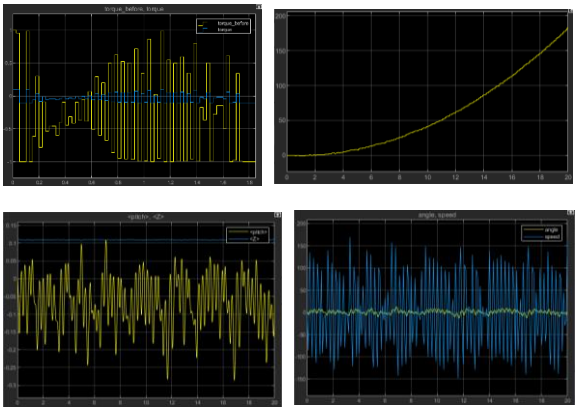


Figure 17 강화학습 기반 제어 시뮬레이션 결과

로봇의 질량을 0.1kg에서 0.01kg으로 낮추었을 때, 제어 상태가 빠르게 안정적으로 수렴하는 것을 확인하였다.

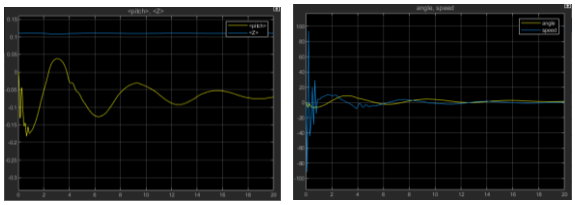


Figure 18 질량 변화 후 제어 시뮬레이션 결과

이후 MATLAB을 이용하여 IMU 센서를 연결한 아두이노와 시리얼 통신을 통해 센서값을 읽고 모터를 제어할 수 있음을 확인하였다. 다만 강화학습 모델은 아두이노에 직접 업로드할 수 없었다.

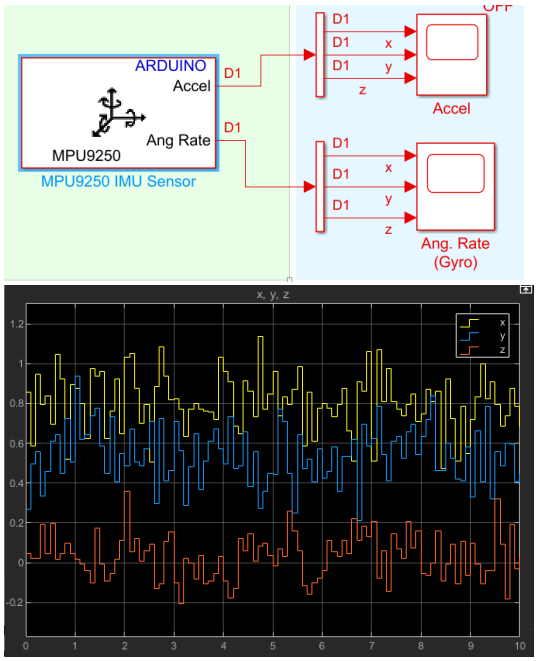


Figure 19 MATLAB과 아두이노 IMU 센서 간 시리얼 통신 결과

모든 코드는 Github에 오픈소스하여 후속 연구가 진행될 수 있도록 기여한다.

[MATLAB 코드]
(https://github.com/nhk9680/Capstone)

[웹/앱 코드]
(https://github.com/nhk9680/capstone-App-Web)

참조

[1] Rahman, MD Muhaimin, SM Hasanur Rashid, and M. M. Hossain. "Implementation of Q learning and deep Q network for controlling a self balancing robot model." *Robotics and biomimetics* 5.1 (2018): 8.

[2] Sun, Qifeng, et al. "Design and application of adaptive PID controller based on asynchronous advantage actor–critic learning method." *Wireless Networks* (2019): 1-11.

[3] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

S. Mohapatra, R. Srivastava and R. Khera, "Implementation of a Two Wheel Self-Balanced Robot using MATLAB Simscape Multibody," *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, Gangtok, India, 2019, pp. 1-3, doi: 10.1109/ICACCP.2019.8883007.