(물병을) 잘 세우는 로봇

임준범(팀장, 기계공학과), 최준빈(기계공학과), 최수용(기계공학과), 한혁규(기계공학과), 김도엽(기계공학과), 권혜령(새내기과정학부), 박관호(새내기과정학부), 정인서(새내기과정학부), 이정민(새내기과정학부), 최우준(새내기과정학부), grubler

MICROROBOT RESEARCH

Abstract

위 연구는, 일상생활 속에서 쉽게 풀 수 없는 문제를 DQN을 통해 풀고자 진행되었다. bottle filp은 사람은 몇 번의 수행으로 그 방법은 쉽게 터득할 수 있지만, 역학적으로 분석하기에는 어려움이 있는 문제이다. 이에 따라, 목표 문제를 bottle flip로 설정하였다.

처음 목표는 3자유도의 로봇팔에 대해 학습을 계획했지만, 로봇팔의 어깨-팔꿈치를 잇는 부분의 진동하여 학습에 방해가 되었다. 이 부분은 물병 던지기 모션에 큰 영향을 주지 않음을 깨닫고, 제거하여 2자유도를 바꾸었다. 그리퍼는 연구에서 지정한 물병을 바탕으로 설계되어, 물병의 머리 부분을 잡았을 때 흔들림이 없도록 하였다.

330fps 카메라를 통해 물병 착지 시 각도와 각속도를 구하고, 이를 통해 보상값을 계산한다. 물병을 놓는 관절의 각도를 올리기/유지/내리기 3가지 액션에 대해 강화학습을 진행하였다. Ros2를 통해 강화학습 환경 및 로봇팔 구동 환경을 분리하였으며, 학습은 pytorch를 이용하여 구현했다.

Theoretical background

• 01 강화학습

DQN은 "Deep Q-Network"의 약자로, 강화 학습(Reinforcement Learning)에 기반한 딥러닝 알고리즘이다. DQN은 Q-learning이라는 강화 학습 알고리즘을 딥러닝에 통합한 형태이다. Q-learning은 에이전트가 특정 상태에서 특정 행동을 취할 때의 기대 반환값을 나타내는 Q함수를 학습하는 알고리즘이다. DQN은 이 Q함수를 근사하기 위해 신경망을 사용하며, 이 신경망은 상태를 입력으로 받아 각 가능한 행동에 대한 Q값을 반환한다. 에이전트는 이 Q값을 기반으로 어떤 행동을 취할지 결정한다.

02 Ros2

ROS2는 Robot Operating System의 두 번째 버전으로, 로봇 소프트웨어 개발을 위한 오픈 소스 미들웨어 플랫폼이다. ROS2는 다양한 로봇 응용 프로그램을 개발하고 관리하는 데 사용되며, 분산 시스템을 위한 통신, 하드웨어 추상화, 디바이스 드라이버, 패키지 관리 등을 지원한다.

03 OpenCV

OpenCV는 컴퓨터 비전 라이브러리로, 이미지 및 비디오 처리에 사용돼. 주로 실시간 이미지 프로세싱, 객체 검출, 얼굴 인식, 모션 추적 등에 활용된다. OpenCV는 다양한 언어에서 사용 가능하고, 컴퓨터 비전 알고리즘을 쉽게 구현할 수 있는 다양한 함수와 도구를 제공한다.

MATERIALS & METHODS.

01 로봇팔 제작

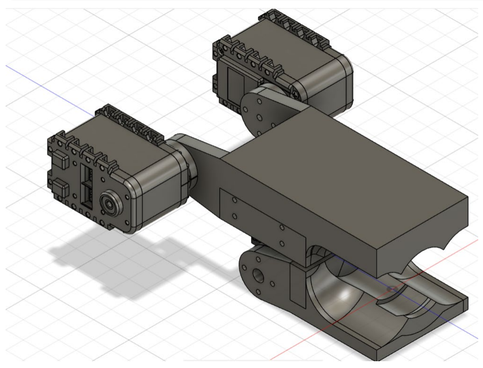


그림 1. 그리퍼 설계 사진

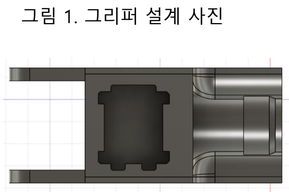


그림 2. 그리퍼(상부) 설계 사진

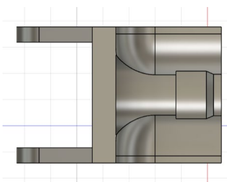


그림 3. 그리퍼(하부) 설계 사진

- 그리퍼는 연구에 사용할 임의의 물병을 바탕으로 설계되었으며, 물병의 머리 부분을 잡았을 때 흔들림이 없도록 설계하였다.
- 상부 그리퍼는 물병을 던질 시, 물병 머리 부분에 반발력을 가해주어 회전을 유발할 수 있어야 한다. 이에 따라 그리퍼는 하부 그리퍼만 움직여 물병을 놓거나 집을 수 있는 구조로 설계되었다.
- 로봇팔의 액추에이터는 AX-12A, AX18A 다이나믹셀을 사용했다. 물병 던지기 시, 충분한 토크가 물병에 가해져야 물병이 회전할 수 있다, 이를 위해 지지면과 로봇팔을 잇는 관절에는 다이나믹셀을 2개 사용하였다.
- 로봇팔은 프로파일로 제작된 지지대에 부착된다.
- 로봇팔은 컴퓨터와 u2d2(통신 변환 장치)를 통해 연결되어 제어된다.

02 강화학습 및 로봇팔 구동 프로그램

그림 4. 작동 사진1

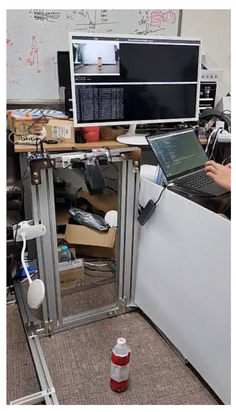


그림 5. 작동 사진2

- Ros2 를 사용하여, 전체 프로그램을 그리퍼 구동 및 카메라 감지/학습(DQN) 2개의 부분으로 나누었다. 각 프로그램은 취해야할 액션(물병을 놓는 각도)와 보상 함수에 들어갈 값(물병 착지 시, 물병의 각도 및 각속도) 값을 Topic 형식으로 주고 받는다.
- 물병은 상/하 부분이 빨간색 프린팅 종이로 감싸져 있다. OpenCV를 이용하여, Thresholding을 통해 상/하 각각의 빨간색 사각형을 감지하고, 두 사각형의 중심점을 잇는 선의 기울기 및 변화량을 분석하여 각도 및 각속도를 도출한다.
- 그리퍼는 컴퓨터 키보드의 입력값을 통해 제어한다. 물병 던지기 전, o(열림)/c(닫힘) 키를 통해 그리퍼를 조작할 수 있다. Q를 누르면 그리퍼 조작이 종료되며, action을 전송한다. Action을 전송받으면 카메라 화면이 나타나며, 이 화면에서 r을 누르면 물병을 던진다. 물병이 일정 범위 아래로 떨어지면, 그때의 각도와 각속도를 전송한다.
- Info 함수를 이용하여 그리퍼를 놓는 위치 및 물병의 각도 및 각속도를 실시간으로 확인할 수 있다.

MATERIALS & METHODS.

03 실험 환경

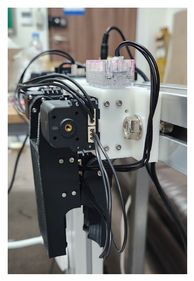


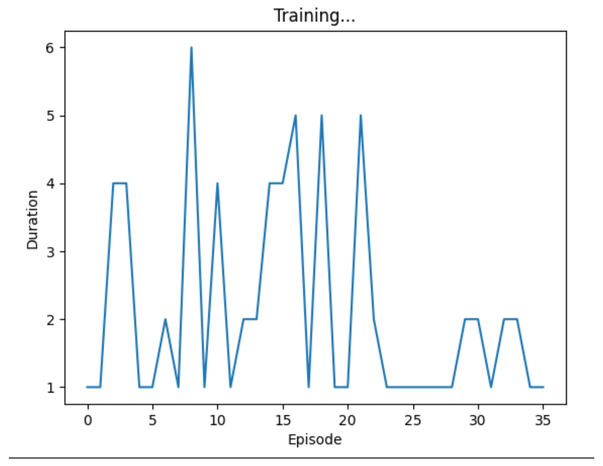
그림 6. 로봇팔

그림 7. 물병

그림 7. 물병 던지기 학습 환경

- DQN에 사용된 Policy_net은, layer 3개이며 hidden layer의 노드는 128개이다. Action은 물병 놓는 각도 증가/유지/감소 3가지로 설정하였으며, 각도의 변화 폭은 다이나믹셀 기준으로 4position 변화에 해당한다.
- 그리퍼는 프로파일 지지대 상단부에 설치되어, 물병이 회전하며 바닥에 도달할 때까지 충분한 시간을 확보할 수 있도록 하였다.
- 한 쪽 면을 우드락으로 막아, 물병을 이외에 다른 물체를 감지하지 못하도록 하였다. 카메라에서 물병 방향으로 조명을 설치하여 물병을 잘 감지할 수 있도록 하였다.
- 물병의 각도 및 각속도는 칼만 필터를 사용하여 안정화시켰다.

Result & Conclusion



- DQN을 사용하여 물병을 성공적으로 세우는 데 성공하였다.
- 위 그래프에서, duration은 물병 던지기의 성공과 그 다음 성공 사이 걸린 시간이다. 학습을 진행할수록, 이 시간이 줄어드는 것을 확인할 수 있다.
- 약 40번 정도 던진 후, 물병 던지기의 성공률은 안정화되었으며, 이때의 로봇의 관절은 586이였다. 이후, 학습을 더 진행한 결과, 오버 피팅이 발생했다.
- 본 저자가 실험을 관측한 결과, 관절의 각도가 586에서 큰 성공률을 보였지만, 594에서도 높은 성공률을 보였다.

Further research task

- 이번 연구에선 Action을 물병 놓는 각도의 증가/유지/감소 3가지로 설정하였으나 물병이 서는 데 영향을 받는 변인으로 회전하는 모터의 속도, 회전이 시작하는 각도 등 여러 가지 변인이 있을 수 있다. 알고리즘을 단순화하기 위하여 나머지 변인들을 모두 고정한 채로 연구를 진행하였으나 후속 연구에서는 위 변인들을 DQN 알고리즘에 넣어서 보다 물병 세우기의 정확도를 높일 것이다.
- 이번 연구에선 특정한 물병에 대해 DQN 알고리즘을 적용하여 Q 함수값을 도출해 내기 때문에, 물병에 채워진 물의 양이 달라질 경우 다시 학습을 진행해야 한다는 단점이 있다. 물의 양과 관절의 각도 간의 관계를 DQN 알고리즘으로 구할 시 물병에 담긴 물의 양이 달라질 때마다 학습을 다시 진행해야 한다는 단점을 없앨 수 있을 것이다.
- 이번 연구에선 사람이 몇 번의 수행으로 방법을 쉽게 터득할 수 있으나 역학적으로 분석하기 어려운 과제인 물병 세우기 로봇을 DQN 알고리즘을 통해 만들었다. 이번 연구를 좀 더 발전시킬 시 물병을 뚜껑으로 세우거나 두 바퀴를 돌려서 세우는 등 사람이 시도해도 방법을 터득하기 어려운 과제들을 로봇과 DQN 알고리즘을 통해 해결할 수 있을 것이다.

Comment

- 연구를 진행하면서, 초기 제작한 그리퍼에서, 물병을 놓친다는 문제를 발견하여 처음부터 다시 모델링하여 새로운 그리퍼를 만들었습니다. 로봇팔 관절 수를 줄이고, 로봇팔 전체의 높이를 조정하는 등 많은 시행착오를 겪었습니다. 이러한 시행착오들을 통해 더 나은 설계 방법을 알게 되었고 다음 연구에서는 시행착오 횟수를 더 줄일 수 있을 것입니다.
- DQN 알고리즘 및 학습 환경 설정에서 많은 어려움을 겪었습니다. Pytorch 및 ros2를 처음 다루는 만큼 어려움이 많았고, 또 이 과정에서 DQN 알고리즘에 대한 이해도를 높일 수 있었습니다. 이러한 이해도를 바탕으로 다양한 방면에서 DQN 알고리즘을 어떻게 활용할 수 있을지에 대해 생각해 볼 수 있었습니다.