



강화학습을 이용한 물류 최적화 프로그램

A반 2조

장동언 박성균 전하영

김민지 전예찬 이경로



목차

1. 추진 배경

2. 프로젝트 개요

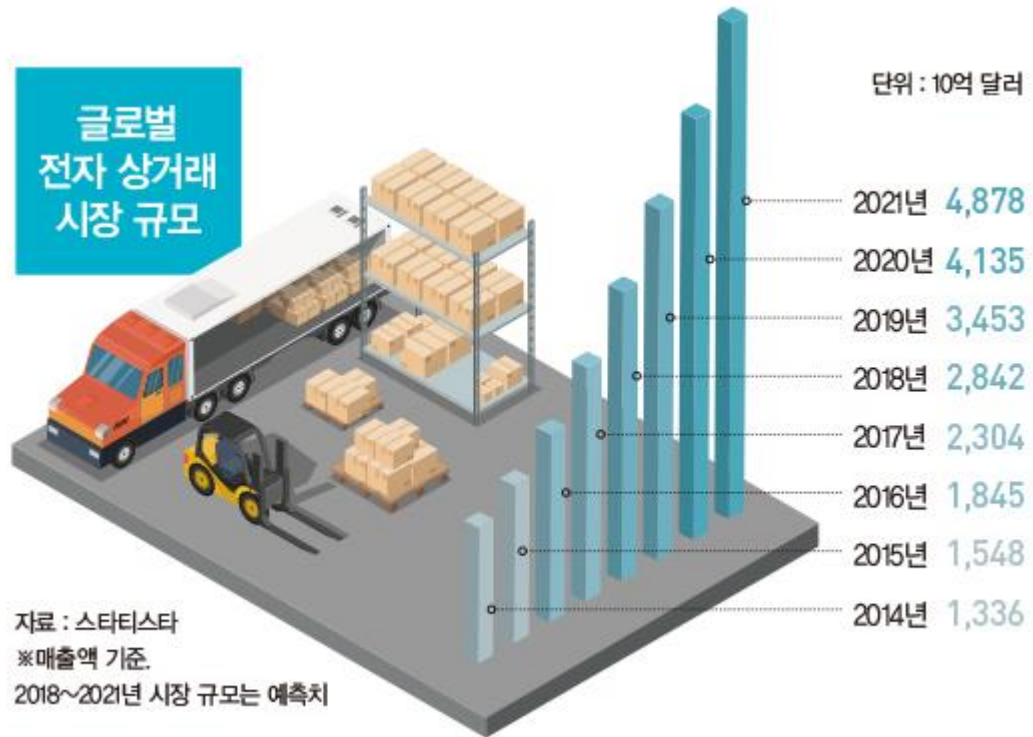
3. 적용 기술 및 기능 시연

4. 한계점 및 개선방향



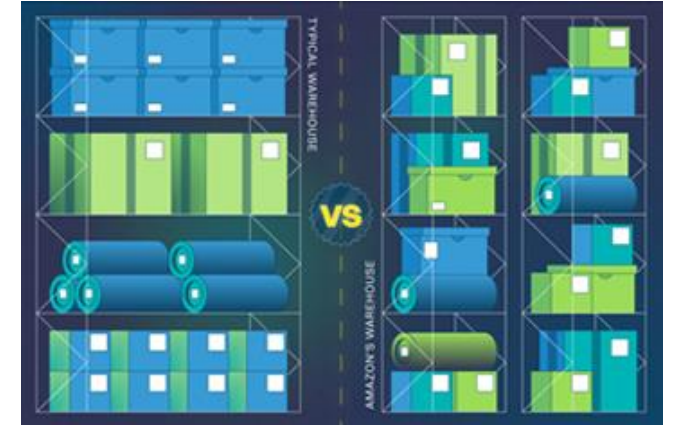
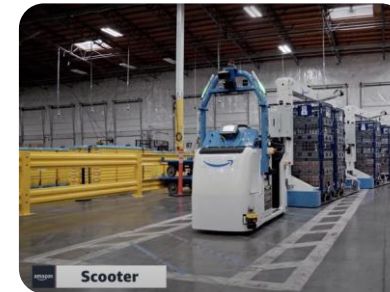
1. 추진 배경

추진 배경



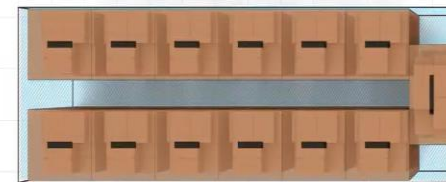
- 물류 시장의 증가로 공간 적재에 대한 비용 상승
- 효율적인 물건 적재 방식이 필요
- 물류환경은 AI, 자동화와 가장 어울리는 산업

기업 동향



[아마존의 Random Stow]

Existing method



CelloTM Loading Optimizer



[삼성 SDS의 Cello Loading Optimizer]

물류관련 하여 이전 기수에서 수행했던 프로젝트 분석

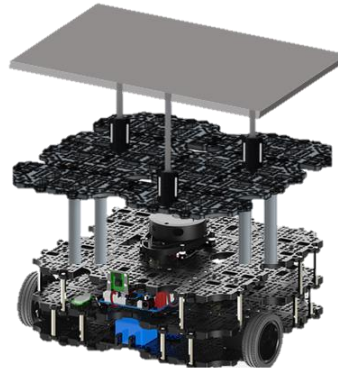
< 16기 A1조 포팡맨 >



업무 효율성 & 편의성 향상을 돕는
사내비품 배송 로봇

송장을 인식하고 앱을 호출하여 **사내**
배송품을 발송인으로부터 수령인으로 전달

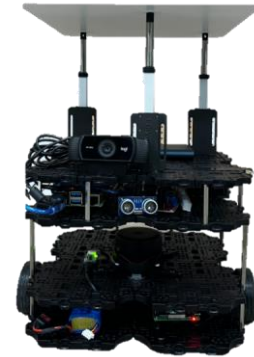
< 18기 A4조 PO가요~♡ >



물류 출하 시 창고에서 화물차로
물류 팔레트를 운반하는 로봇

팔레트의 **목적지를 인식 한 후**
자율주행으로 목적지 까지 팔레트를 전달

< 19기 C2조 트랜스포 >



다중 목적지 자율주행 로봇

물류 정보를 인식해
각 물건 별 적절한 목적지로 배송

- 물건을 **운반하는방법**에 초점을 맞춘 프로젝트 다수
- 새로운 환경의 강화학습 구현 사례 없음

“운반 전 정리를 효율적으로 할 수 있는 알고리즘 필요”

2. 프로젝트 개요

물품 데이터 입력



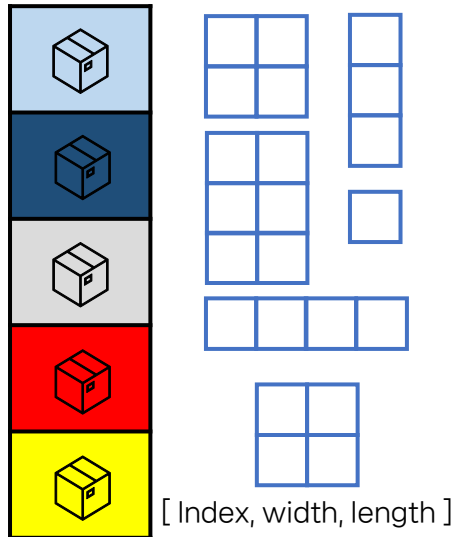
AI 모델을 활용해
위치 및 적재 순서 결정



로봇으로 정보 전송

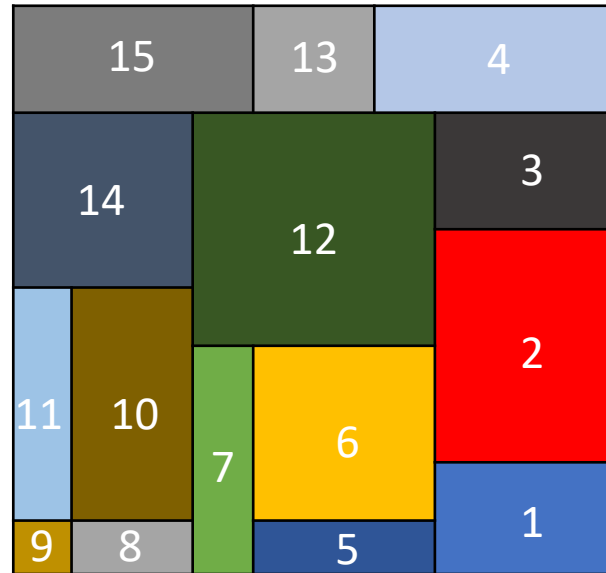


물품 적재



< 물품 정보 >

- 물품의 고유 번호
- 크기 정보 (x * y)



< 최적 적재 위치 계산 >

- AI 모델을 사용해 **최소 공간**으로 물품을 적재할 수 있는 **위치 결정**

* 시뮬레이션 환경과 **강화 학습**으로 AI 모델 학습



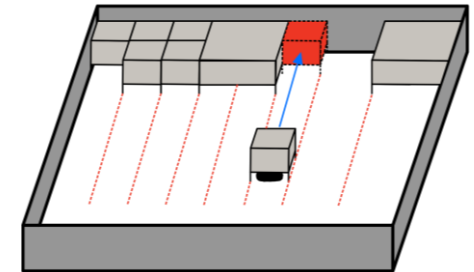
워크스테이션



터틀봇

< 적재 순서 및 위치 정보 전달 >

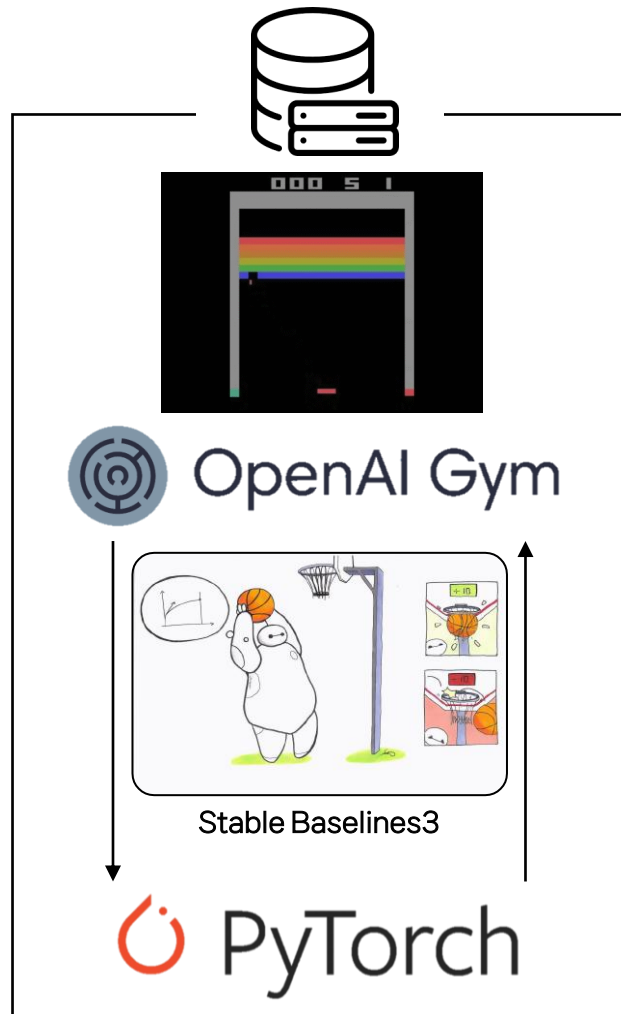
- 적재할 **물품의 위치(x, y 좌표)**와 **순서 데이터**를 로봇으로 전송
- 전달받은 정보를 바탕으로 동작



< 물품 이송 및 적재 >

- 전체 공간(Map)을 중심으로 좌표 기반 이동
- 본래 자리(상차)로 복귀
- 위 과정 반복

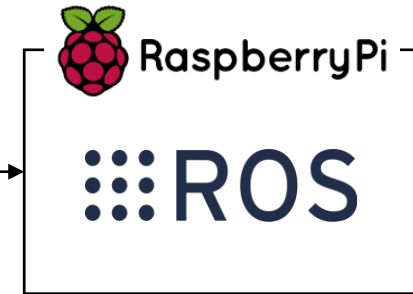
워크스테이션: 강화 학습



원격 PC: 라즈베리 파이,
터틀봇 제어



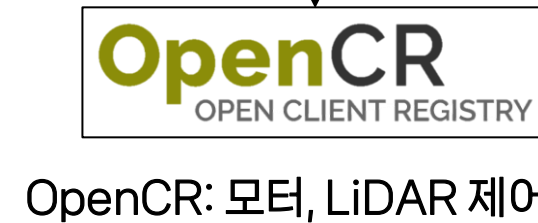
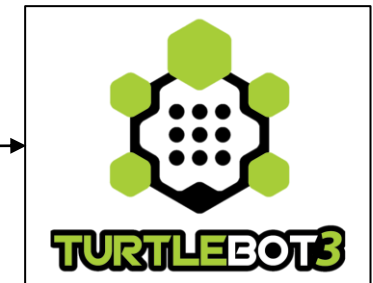
라즈베리 파이: 터틀봇 제어



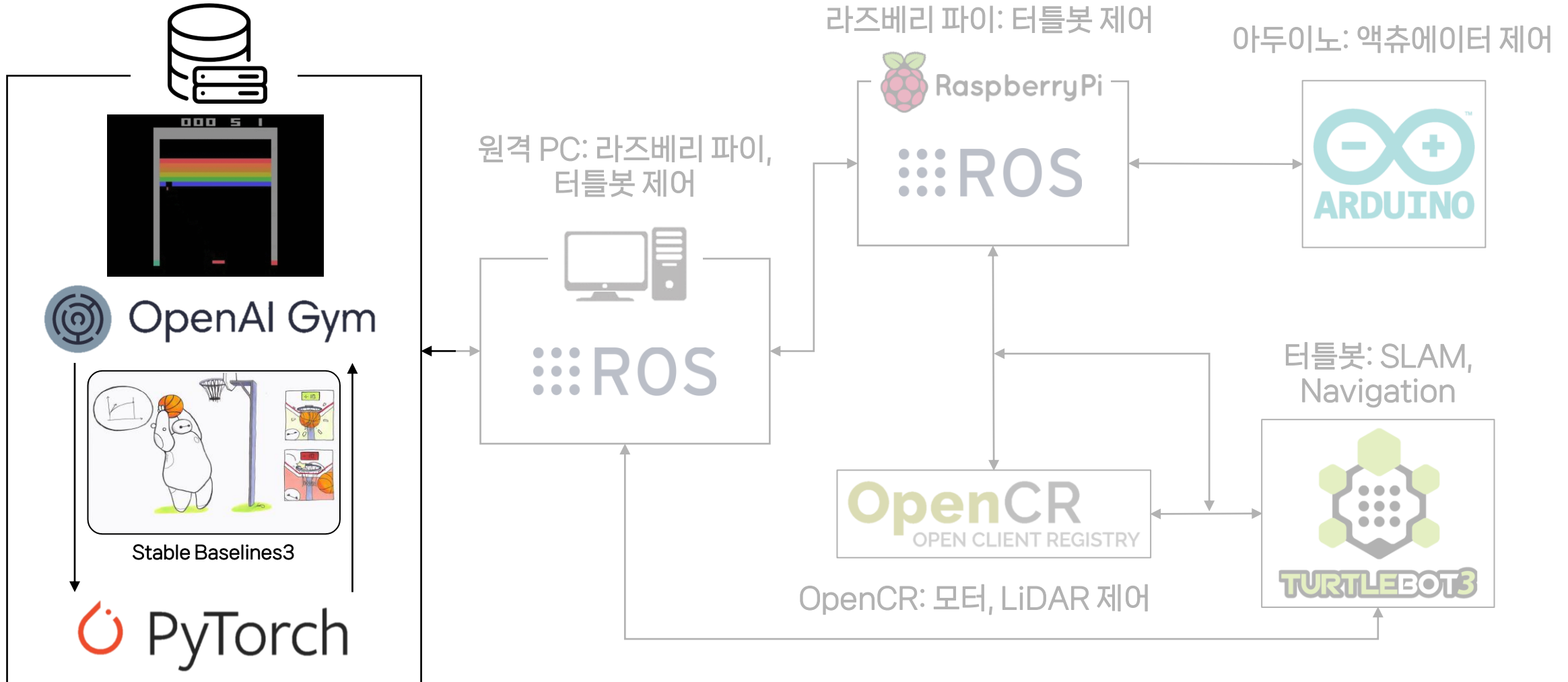
아두이노: 액추에이터 제어



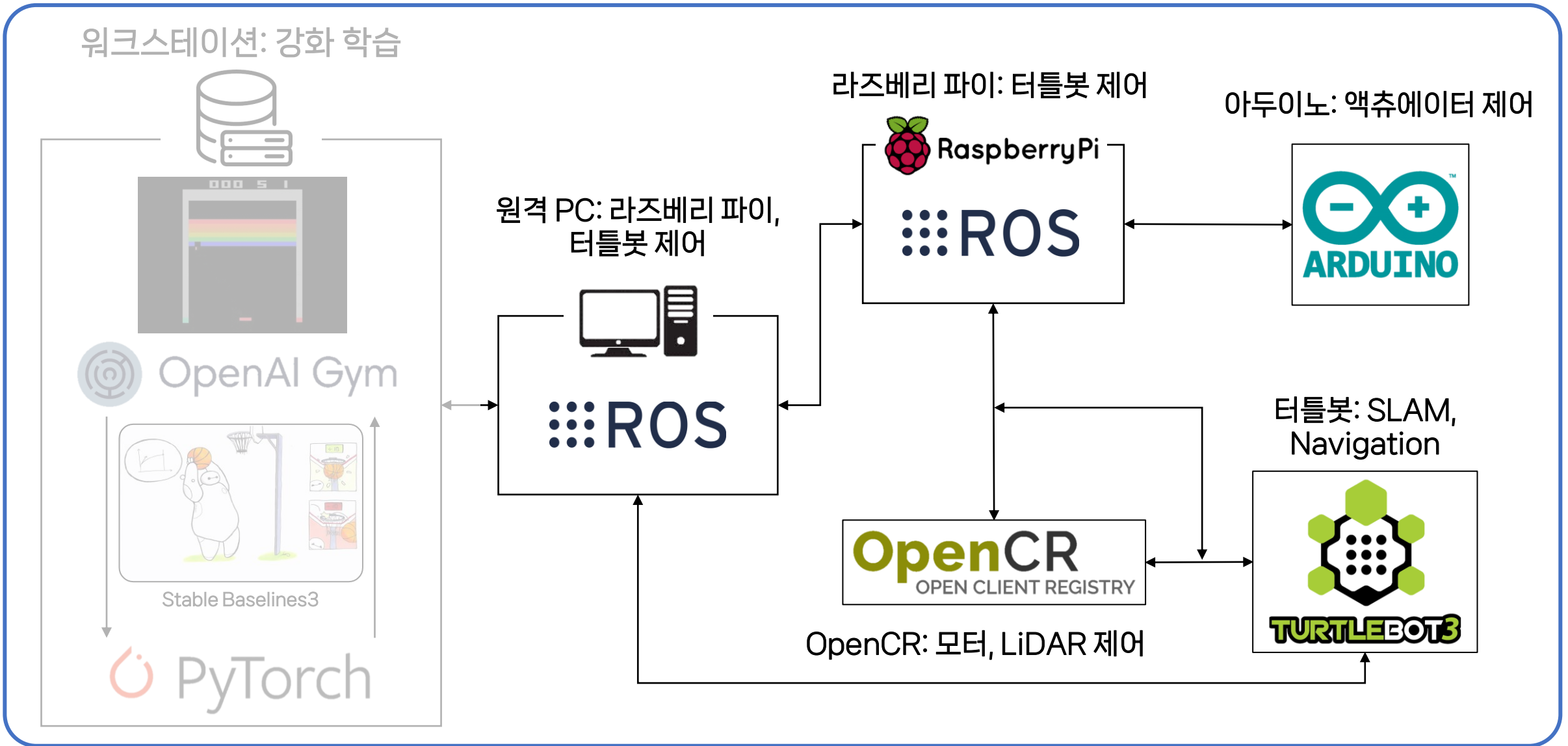
터틀봇: SLAM,
Navigation



워크스테이션: 강화 학습

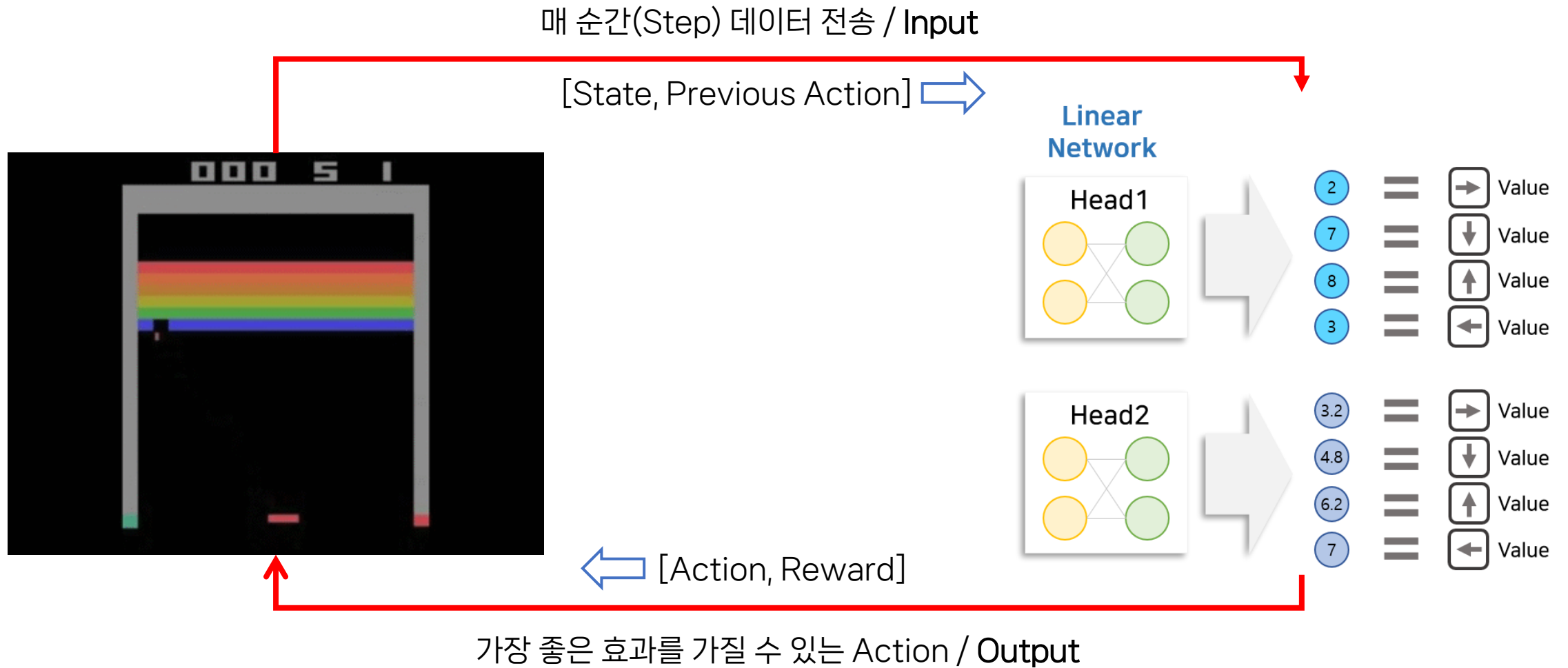


워크스테이션에서 **시뮬레이션을 통한 강화 학습**으로 가장 효과적인 **물류 적재 알고리즘 개발**



알고리즘으로 계산된 물품의 순서, 적재 위치를 넘겨 받아 로봇 동작

3. 적용 기술 및 기능



실시간으로 변화하는 환경을 통해 **매 순간의 상태를 Input data**를 가져오면서,
현재 상태에서 최선의 선택을 Output으로 배출

강화학습을 하기 위해서는 학습이 이루어지는 환경을 정확하게 구현해야 한다.
강화학습을 이용한 가장 대표적인 게임인 Atari의 환경과 우리가 구현한 환경은 아래와 같다.

[강화학습 Environment 예시]



[State]

Map: 게임이 진행되는 맵

Block: 깨야 하는 블록

Ball: 블록을 깨는 공

Ball velocity: 공의 움직임

Paddle: 공을 튕기게 하는 판

[Reward]

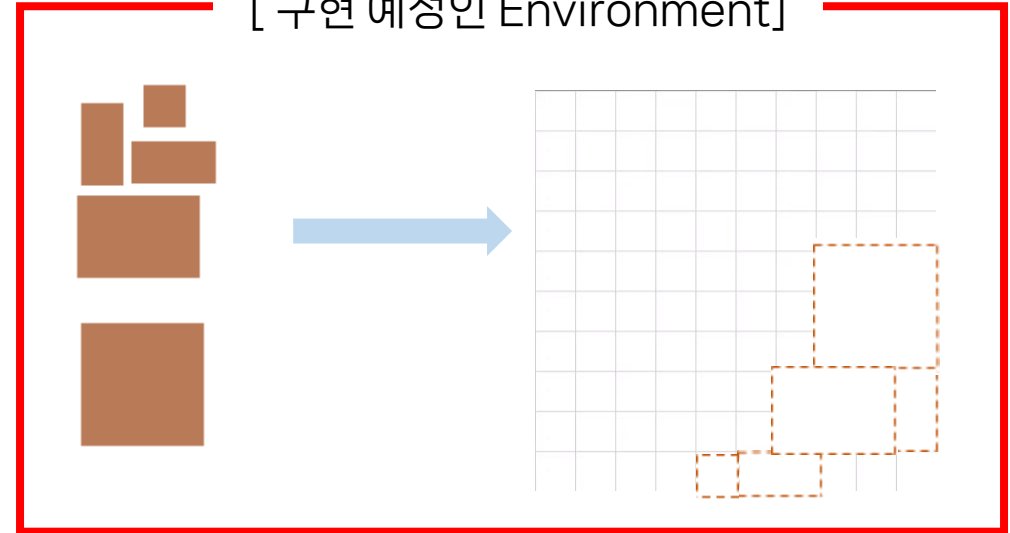
블록을 하나 제거 시 +1

[Action]

왼쪽으로 움직임

오른쪽으로 움직임

[구현 예정인 Environment]



[State]

Map: 현재 창고 상황 (2D matrix)

Box: 적재할 물건의 크기
(width * length)

[Action]

특정 좌표에 물건을 내려 둠 (x, y)

[Reward]

- 물리적으로 가능한 행동 시 지급

- 남은 공간에 반비례하게 지급

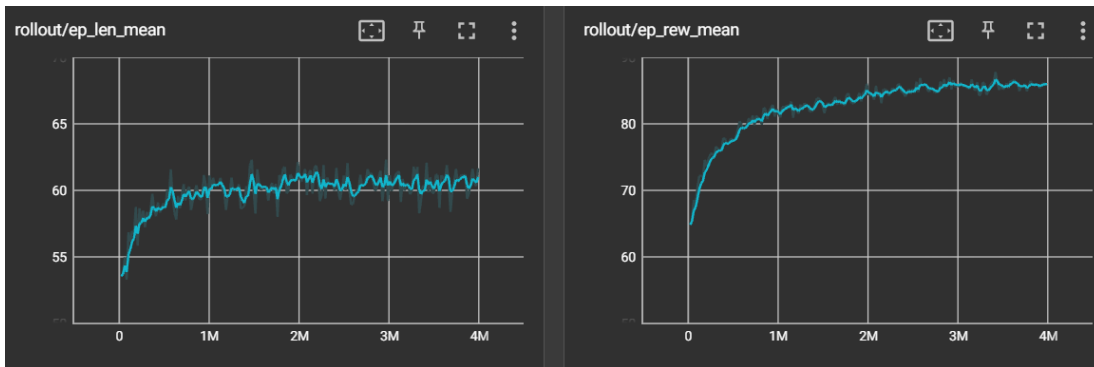
- 채운 공간에 비례하여 지급

Keras RL은 알고리즘 다양성이 적고 편리성이 떨어짐 → **Stable Baselines3 채택**

- Stable Baselines3 : OpenAI의 baseline에서 분화된 라이브러리
OpenAI-gym 환경과 Deep learning 모델을 연결 시켜줌

라이브러리	Multi-Processing	최신 알고리즘 반영	알고리즘 다양성
Keras RL	X	X	7종
SB3	O	O	12종

* GPU로 구동하는 model과 달리 환경과의 상호작용은 CPU로 구동되어 multi-processing 필요



```
from stable_baselines3 import A2C
```

```
model = A2C("MlpPolicy", "CartPole-v1").learn(10000)
```

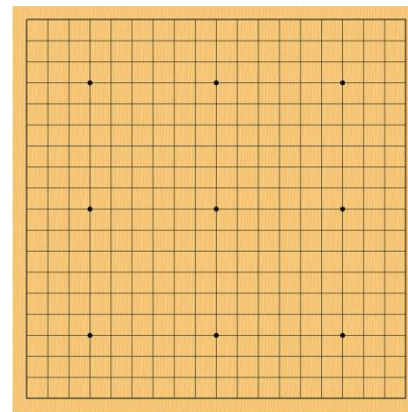
batch size, learning rate 등 일반적인 딥러닝에 필요한 parameter와 exploration rate, update, memory 등 수많은 parameter가 존재하지만

단 한 줄의 코드로 학습이 가능하게 해준다.

PPO(Proximal Policy Optimization) 모델 사용

: 2,000,000 steps(약 40,000 episode) 학습에 약 10시간 소요 (8 core 기준)

수 백만 가지의 경우의 수 중 정답이 단 1개 라면,
그 정답을 우연히 찾기 전까진 학습이 시작되지 않는다.



- **제한사항 설정** : 최대한 많은 제한 사항으로 State 및 Action의 수를 제한하여 정답에 다가갈 확률을 높임
- **Action masking** : 물리적으로 더 이상 불가능한 경우의 수를 제한함 (매 순간마다 유동적인 Action 수 사용)
- **Dynamic programming** : 행동을 세분화 한 후 작은 행동에도 Reward를 부여함 (정답의 세분화)

제한사항

1. 물건의 **순서는 고정**되어 있다.
2. 물리적으로 둘 수 없는 공간일 시 **반복 시행**
3. 모든 물건은 **정사각형**이다.
4. 물건 모양은 **사각형**이다.
5. 물건 적재 시 이동할 길은 고려하지 않는다.



제한사항 해결

1. 물건의 순서는 **무작위**이다.
2. 물리적으로 둘 수 없는 공간은 **두지 못한다**.
3. 물건은 **직사각형**이다. (정사각형 포함)
4. 물건 모양은 **제약이 없다**.
5. 물건 적재 시 로봇의 **이동경로를 고려**한다.

* Action masking : 물리적으로 더 이상 불가능한 경우의 수를 제한하여 학습 속도를 높임

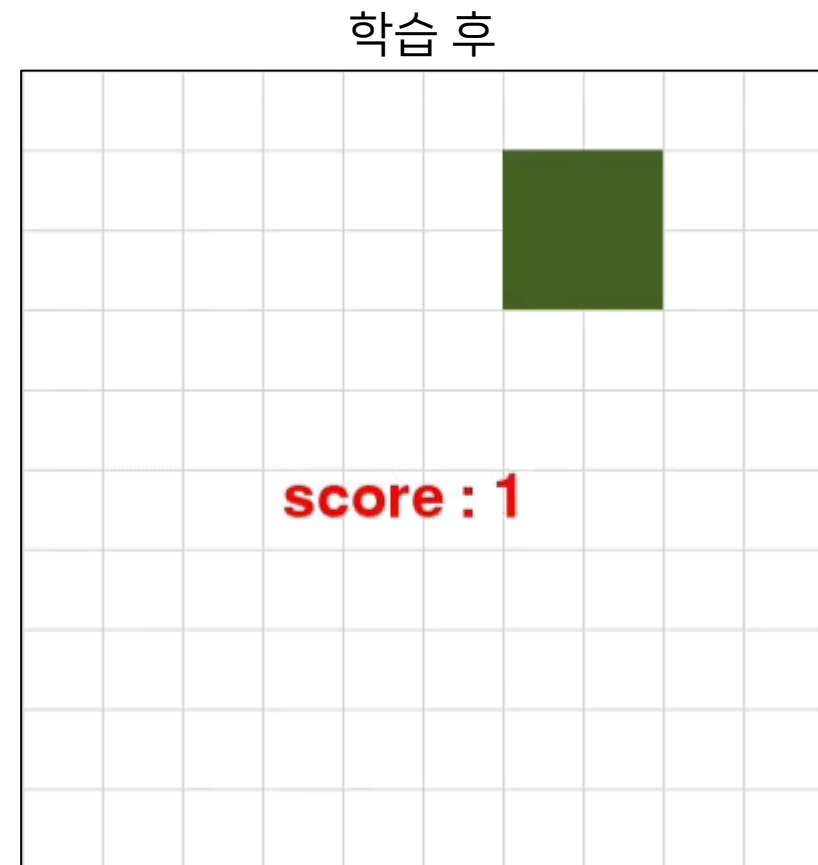
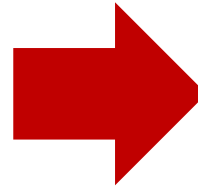
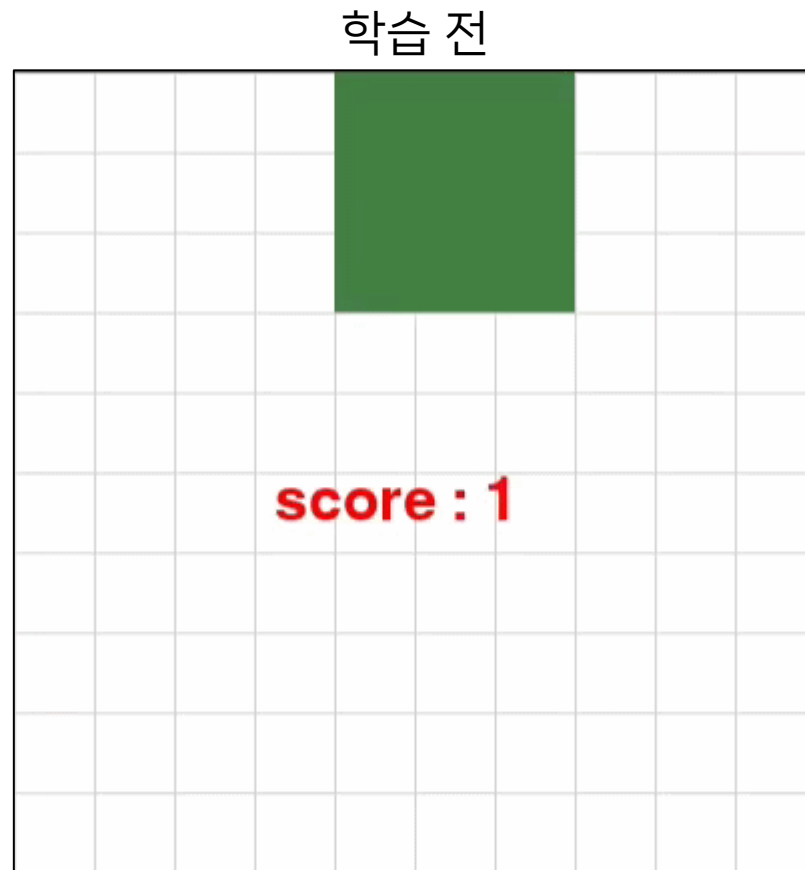
학습 순서 ↓

Ver.	Action Masking	순서 랜덤	직사각형 (회전)	모양	경로 고려
V1	X	X	X	X	X
V2	✓	X	X	X	X
V3	✓	✓	✓	X	X

제한 사항을 하나씩 해제하며 점진적으로 학습하여 학습 속도를 높이하고자 하였음.

V0환경을 통해 학습이 실제로 진행됨을 보고 PPO model이 가장 빠른 학습속도를 가짐을 확인하였다.

Ver	Reward	종료 조건
V0	가능한 위치 : +1 불가능한 위치 : -1	전체의 80%를 채우거나 두지 못 하는 곳에 20회 돌 시 종료



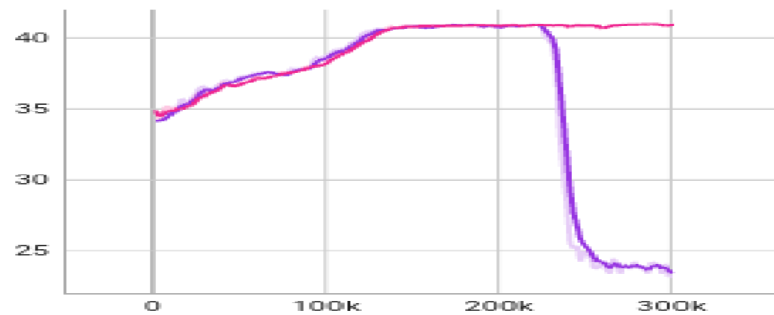
Action Masking을 이용한 결과 학습 시간이 단축되는 것을 확인할 수 있다.

Ver	Reward	종료 조건
V2	매번 물건을 내려 둘 때 크기만큼 지급	더 이상 물건을 내려 둘 수 없을 때 종료

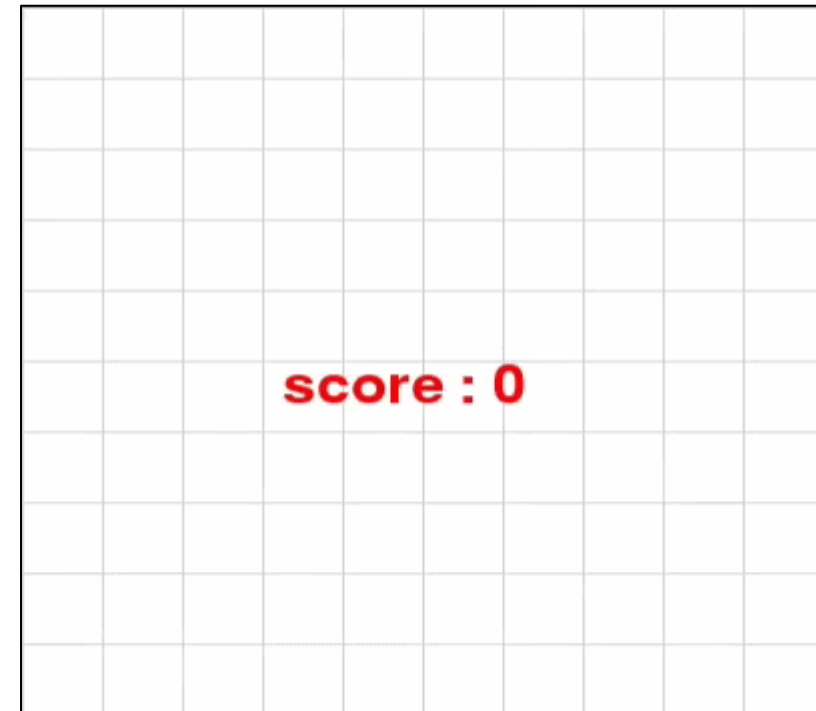
* Action masking : 물리적으로 더 이상 불가능한 경우의 수를 제한하여 학습 속도를 높임

	Masking
—	X
—	O

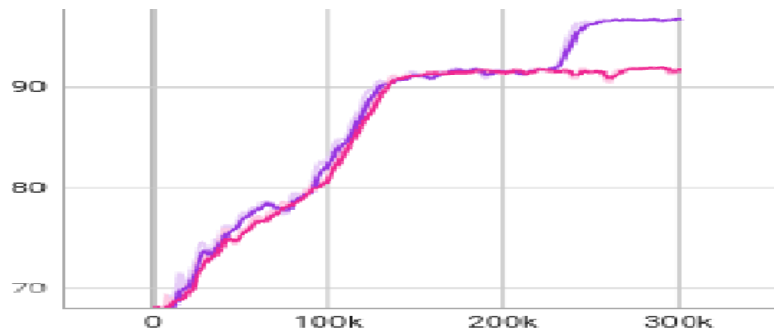
<Action 수>



학습 후



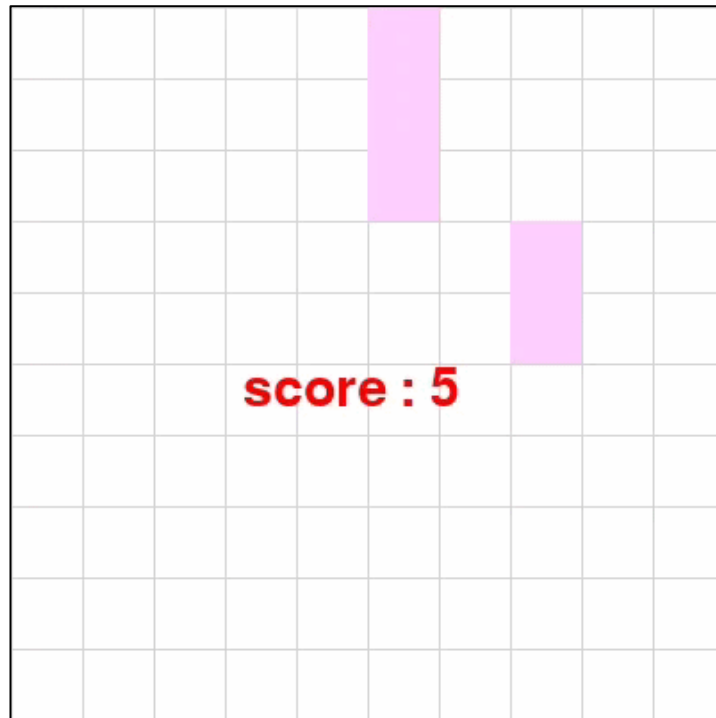
<reward>



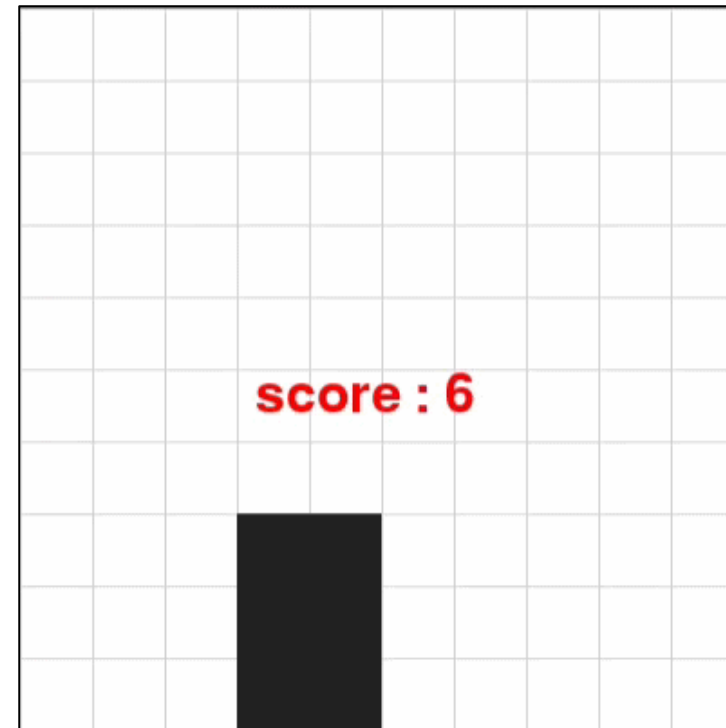
이 프로젝트의 최종 목표였던 Version 3에 대한 학습이 아래와 같이 이루어 짐을 확인할 수 있다.

Ver.	Action Masking	순서 랜덤	직사각형 (회전)	모양	경로 고려
V3	✓	✓	✓	✗	✗

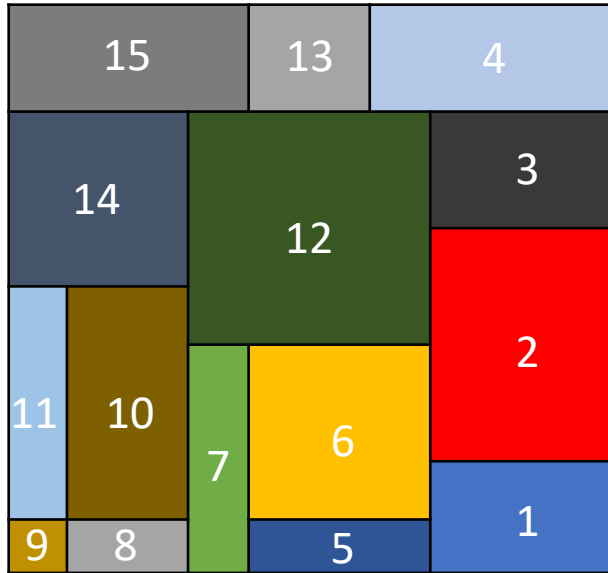
학습 전



학습 후



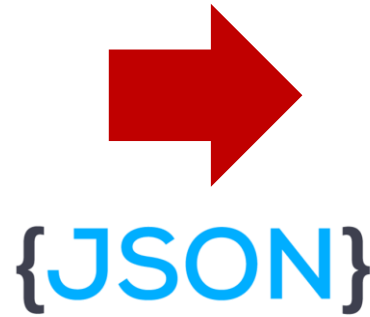
“직사각형 범위 내에서 랜덤하게 물건을 전해줘도
최소 10% 미만의 공간을 남기고 다 활용하게 학습 성공”



< 최적 적재 위치 계산 >

- AI 모델을 사용해 **최소 공간**으로 물품을 적재할 수 있는 **위치 결정**

* 시뮬레이션 환경과 **강화 학습**으로 AI 모델 학습.



[Index, width, length]



< 적재 순서 및 위치 정보 전달 >

- 적재할 **물품의 위치(x, y 좌표)**와 **순서 데이터**를 로봇으로 전송
- 전달받은 정보를 바탕으로 동작

TurtleBot을 이용해 적재 물품을 이송하기 위해 **적합한 리프트 방식** 선정. TurtleBot 최상단에 장착할 것을 고려해야 함.
또한, **안정적으로 이송**할 수 있어야 하며 **중량은 최대한 줄이는 것**을 고려해서 부품 후보 선정.

후보1. 리니어 액추에이터



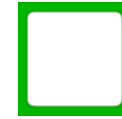
Stroke	최대 100mm
Load	750N
Weight	2kg
Size	60 x 30 mm



후보2. 전동 테이블 리프트



Stroke	160 - 690mm
Load	3000N
Weight	25kg
Size	400 x 300 mm



후보3. 소형 리니어 액추에이터



Stroke	max 50mm
Load	15N
Weight	40g
Size	10 x 10 mm



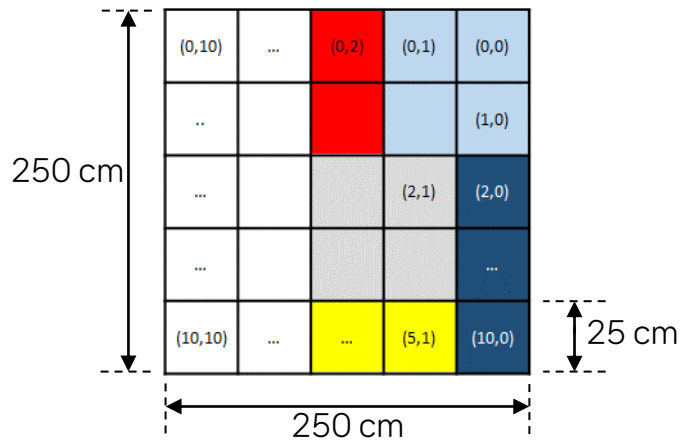
Stroke, Load, Weight 등 여러 요소를 고려한 결과, **리니어 액추에이터를 리프트 부품으로 선정**.

하지만, 단일 부품으로 **리프트시 안정성을 확보**하기 위해 **리프트 플레이트(Lift Plate)**와 결합 부분을 강건하게 설계.

AI 모델 학습 결과를 바탕으로, 실제 TurtleBot이 동작해야 하는 MAP과 Product를 구현할 방안에 대한 고려.

TurtleBot의 크기를 고려해서 최소 단위의 물품 사이즈를 정사각형의 경우 50 x 50 cm, 직사각형의 경우 50 x 25 cm로 선정.

1안



MAP Size : 250 x 250 cm

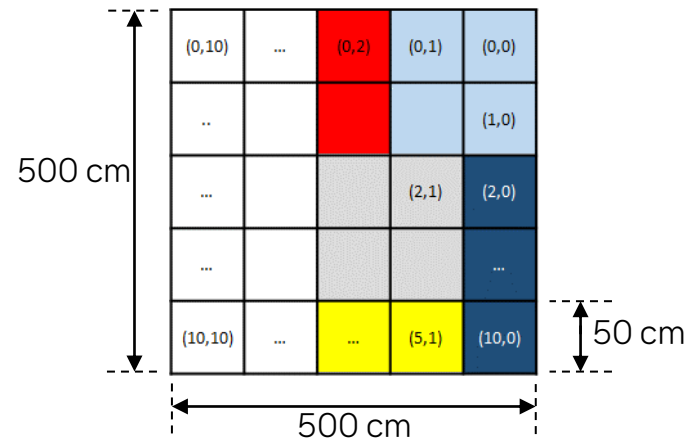
Product Size :

정사각형, 최소 25 x 25 cm

직사각형, 고려 X

! 물품 크기가 Turtlebot이 Rack 아래로 들어갈 수 있는 공간이 부족하여 개선 필요.

2안



MAP Size : 500 x 500 cm

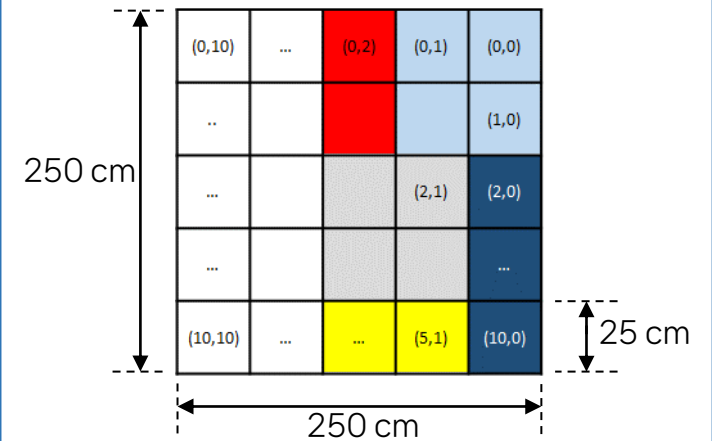
Product Size :

정사각형, 최소 50 x 50 cm

직사각형, 고려 X

! TurtleBot 크기의 한계로 구현할 수 있는 물품의 사이즈가 제한적. 또한, 직사각형에 대한 AI 모델 학습 결과를 반영할 필요

최종안



MAP Size : 250 x 250 cm

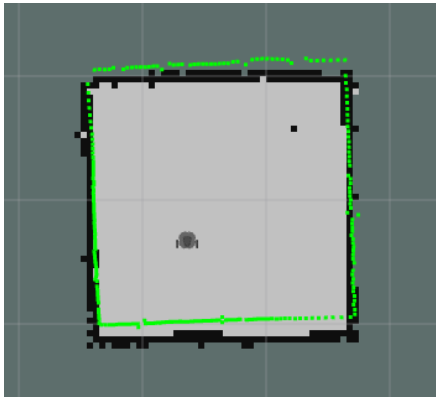
Product Size :

정사각형, 최소 50 x 50 cm

직사각형, 최소 50 x 25 cm

! 최종안으로, TurtleBot 크기와 리니어 액추에이터의 스펙을 고려해서 실제 구현할 수 있는 MAP과 Product의 사이즈를 결정.

SLAM (Simultaneous Localization And Mapping)



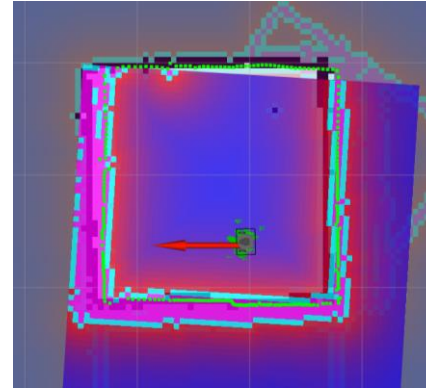
자율주행 차량에 사용되어 주변 환경 지도를 작성하는 동시에 작성된 지도 안에서 차량의 위치를 추정하는 방법.

Gmapping과 **Cartographer** 방식이 존재.

	Gmapping	Cartographer
특징	2D	2D, 3D
	Static한 상황에 유리	Dynamic한 상황에 유리

다시 물건을 놓을 때마다 바로 전에 놓았던 물건은 장애물로 인식 되기 때문에 Cartographer 채택

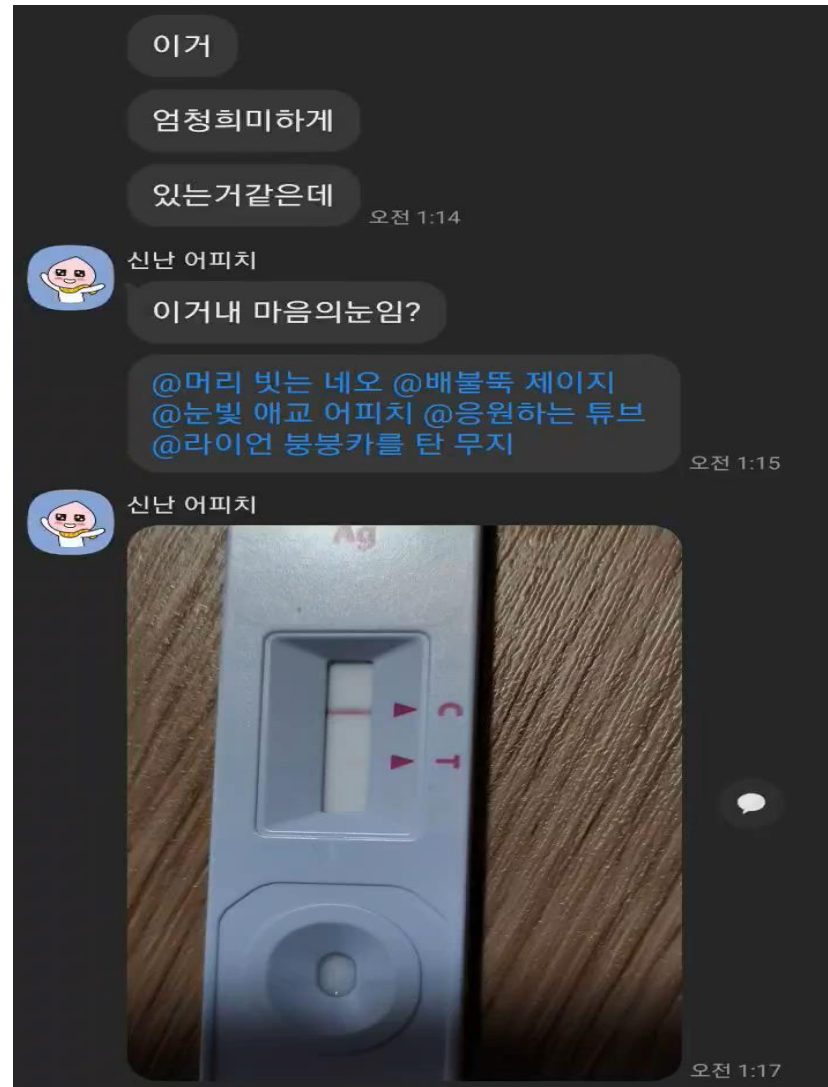
Navigation



SLAM으로 생성된 맵으로 목표지점을 설정, 가는 방법을 제시하고 목적지까지 자율주행.

센싱 → 위치 추정
→ 모션 계획 → 장애물 회피

- ! JSON 형식으로 목적지의 x, y 좌표를 받아 자율주행할 예정이었으나 터틀봇에서 파일을 수신하지 못해 직접 좌표를 입력하는 형식으로 구현



4. 한계점 및 개선방향


1. 항상 모든 공간을 꽉 채울 수 있도록 설계하여 물건을 안쪽부터 차곡차곡 쌓지 않는다.

→ 모든 물건을 넣더라도 공간이 남을 수 있도록 학습 환경 변경

2. 시뮬레이션 환경에선 오차 없이 적재할 수 있으나,
실제 환경에서 동작 시 물리적인 오류 및 오차가 생길 수 있음.

→ 실제 환경에서 수 차례 시험하면서 물건 적재 안정성, 적재 위치의 오차 등을 수집하고 개선방안 필요

**기존 환경이 정의되지 않은 상태에서
Action, State, Reward 등 환경 정의를 이끌어 내어
성공적으로 강화학습을 수행 함.**



감사합니다

