

## [캡스톤디자인 결과보고서]

### ■ 과제명

|     |   |      |             |
|-----|---|------|-------------|
| 과제명 | Field painting robot using ROS-based DQN reinforcement learning and PID control | 참여학기 | 2022 년 2 학기 |
|-----|---|------|-------------|

### ■ 강좌정보

|      |  |      |            |
|------|--|------|------------|
| 과목명  | 소프트웨어융합캡스톤디자인                          | 학수번호 | SWCON40100 |
| 과제기간 | 2022 년 09 월 01 일<br>~ 2022 년 12 월 16 일 | 학점   | 3 학점       |

### ■ 팀구성

|      |                                     |            |         |     |
|------|-------------------------------------|------------|---------|-----|
| 팀명   | Gear ( Good efficiency and result ) |            | 팀구성 총인원 | 2 명 |
| 구분   | 성명                                  | 학번         | 학부(과)   | 학년  |
| 대표학생 | 김정윤                                 | 2018100670 | 기계공학과   | 4   |
| 참여학생 | 방지호                                 | 2018100694 | 기계공학과   | 4   |
|      |                                     |            |         |     |
|      |                                     |            |         |     |
|      |                                     |            |         |     |

### ■ 지도교수 확인

|      |    |        |         |          |
|------|----|--------|---------|----------|
| 지도교수 | 성명 | 김동한    | 직급      | 전임교원     |
|      | 소속 | 전자정보대학 | 지도교수 확인 | 성명 : (인) |


### ■ 붙임

[양식] 과제 요약보고서

[결과물] 최종결과물 (최종작품 사진/도면/발표자료 등)

본 팀은 과제를 성실히 이행하고 이에 따른 결과보고서를 제출합니다.

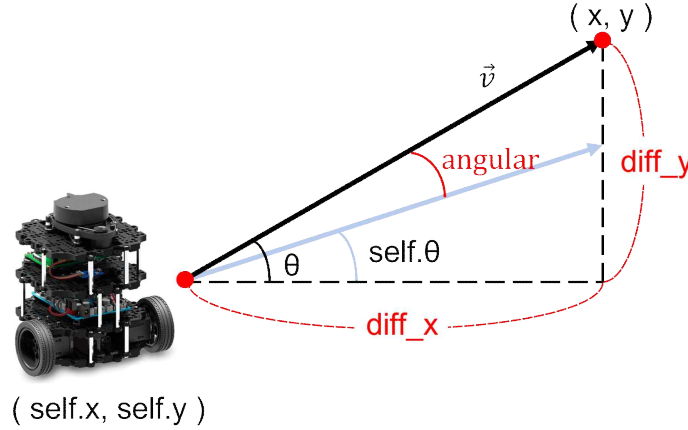
2022 년 12 월 16 일

팀 대표 : \_\_\_\_\_ 김정윤 \_\_\_\_\_ 

## [캡스톤디자인 과제 요약보고서]

| 과 제 명   | Field painting robot using ROS-based DQN reinforcement learning and PID control |
|---|---|
| <p>1. 과제 개요</p> <p>가. 과제 설계 배경 및 필요성</p> <p>모든 아스팔트 도로에는 도로가 가지는 기호인 차선이 존재한다. 이런 차선들은 현재 색칠부터 길이 측정까지 대부분 인간의 손에 의해 그려진다. 몇 백 km나 되는 도로와 경사면, 여름철 등 특정 환경이 차선 그리는 작업을 굉장히 어렵게 만든다.</p> <p>이외에도 축구, 야구, 테니스와 같은 스포츠 계열, 최근 유행했던 오징어게임과 같이 필드를 만들어야 하나 매번 그리는 귀찮은 경우가 많다.</p> <p>위와 같은 경우, Field painting robot을 이용할 수 있는데 PID제어로 속도를 제어해 원하는 위치로 이동하는 방법과 machine learning 중 DQN 강화학습모델을 이용해 원하는 위치로 이동하는 2 가지 방법에 대해 robot model을 설계하였다.</p> <p>나. 과제 주요내용</p> <p>ROS 환경에서 제공해주는 시뮬레이션 기능인 "Gazebo"에서 제작한 터틀봇 패키지 및 소스코드들이 기존 설계한 것과 같이 작동하는지 확인하였다. 위 코드를 기반으로 turtlebot3 Hardware를 직접 조립해 설정을 해주어 실제 구동이 가능한지 확인하였다.</p> <p>제작한 코드로는 (1) PID제어를 통한 속도 제어를 하여 위치 제어 및 스캔 된 레이저의 개수의 따라 속도 저하와 최종적으로 멈출 수 있는 코드와 (2) 머신러닝 중 DQN 학습모델을 활용해 여러 목표점까지 벽에 충돌하지 않고 도착하면 양의 보상을, 충돌 또는 목표점과 거리가 멀어지면 음의 보상을 주는 방식으로 원하는 위치를 따라갈 수 있게 학습하는 코드를 작성하였다. (3) 또한 위 코드의 odometry를 실시간으로 5Hz의 주파수로 받아 로봇의 trajectory를 matplotlib을 이용해 그래프로 저장 및, .csv 파일로 경로 값들을 엑셀 파일로 추출하여 저장하는 코드를 제작하였다.</p> <p>다. 최종결과물의 목표</p> <p>"축구장", "야구장", "오징어 게임장", "테니스 코트 장" 총 4가지의 field를 그리는 것이 최종목표이다. Software 상의 "Gazebo"환경에서 PID제어를 통한 trajectory를 뽑아 "축구장", "야구장", "오징어 게임장" 3가지 field를 제작하였고, DQN 강화학습으로 "테니스 코트 장"을 제작하였다.</p> <p>위 결과를 바탕으로 실제 터틀봇을 구동해 "축구장"을 그릴 수 있음을 구현하였다.</p> |   |
| <p>2. 과제 수행방법</p> <p>(1) 실행환경: Ubuntu 18.04 LTS버전, ROS-melodic, python2.7.17, tensorflow 2.1.0, numpy 1.16.4, keras 2.3.1, Anaconda 3</p> <p>Ubuntu 22.04, ROS2-humble, python 3.9, tensorflow 2.1.0, numpy 1.16.6, keras 2.3.1, Gazebo9</p> <p>(2) Hardware 스펙</p> <p>모델 : turtlebot3 burger, Size (L x W x H)=138mm x 178mm x 192mm, Weight=1kg</p> <p>SBC (Single Board Computers) : Raspberry Pi</p> <p>LDS(Laser Distance Sensor) : 360 Laser Distance Sensor LDS-01</p> <p>IMU센서 : Gyroscope 3 Axis, Accelerometer 3 Axis</p> <p>위 2 가지 실행환경 중 1번째 ROS-melodic 위주로 과제를 수행하였음. ROS1에서 제작한 패키지 및 노드, 소스코드들을 활용하여 실제 turtlebot3(model=burger)를 작동하였음.</p> <p>(3) Waypoint 기능 구현</p>  |   |

Waypoint란 특정 point를 따라 이동하는 것을 의미한다. 도로의 차선과 축구장, 야구장과 같은 원하는 map을 그리기 위해선 정교한 waypoint를 수행할 수 있어야 한다. 터틀봇3에서는 Odometry 즉, 로봇의 위치 정보를 subscriber를 통해 받아올 수 있으며, cmd\_vel 즉, 선속도와 각속도 정보를 publisher를 통해 로봇에게 줄 수 있다. 이 두가지를 가지고 터틀봇3의 waypoint 기능을 구현하였다.



**Fig. 1** Conceptual schematic for waypoint implementation.

앞서 언급하였듯이 Odometry를 통해 터틀봇3의  $x$ ,  $y$ ,  $\theta$  값을 받아들일 수 있다. 이때 각각의 값을  $self.x$ ,  $self.y$ ,  $self.\theta$ 라고 하자. 도착 지점의 좌표는 명령을 통해 넣어주는 값으로  $x$ ,  $y$ 으로 정의하였다.

$$\vec{v} = (x - self.x, y - self.y) = (diff_x, diff_y) \dots (1)$$

$$\hat{v} = \left( \frac{diff_x}{\sqrt{diff_x^2 + diff_y^2}}, \frac{diff_y}{\sqrt{diff_x^2 + diff_y^2}} \right) \dots (2)$$

이를 활용하면 (1)의 식과 같이 터틀봇3와 목표점까지의 벡터  $\vec{v}$ 를 구할 수 있으며, 터틀봇3와 목표점까지의 거리를 나눠주면 (2)의 식과 같이 크기가 1인 단위벡터  $\hat{v}$ 를 구할 수 있다. (1), (2) 식을 활용하여 이후 터틀봇3를 목표점까지 일직선으로 이동시킬 수 있다.

$$angular = \theta - self.\theta = \tan^{-1}\left(\frac{diff_y}{diff_x}\right) - self.\theta \dots (3)$$

터틀봇3를 안정적으로 이동시키기 위해선 먼저  $self.\theta$  값 즉, 로봇이 바라보는 방향과 목표점까지의  $\theta$ 값을 일치시켜야 한다.  $self.\theta$ 와  $\theta$ 가 일치한다는 것은  $angular$ 가 0이 되도록 만든다는 것이다. 이를 위해선  $cmd\_vel$ 으로 각속도를 조절해야 한다. 각속도는 (3) 식에서 볼 수 있듯  $\theta - self.\theta$ 를 계속 업데이트하는 방식으로 조절할 수 있다. 이후 PID 제어를 추가하여 각속도의 성능과 안정성을 향상시킬 수 있다.

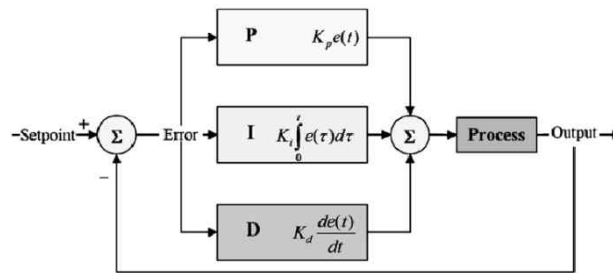
$$linear = \vec{v} \cdot \hat{v} \dots (4)$$

$angular$  값을 0으로 맞춰주었다면 터틀봇3를 바라보고 있는 방향 즉,  $x$ 축 성분의 선속도 명령만을 주어 목표점으로 이동시킬 수 있다. 그러나 이 과정에서 내부와 외부 요인으로 로봇이 일정한 방향으로 갈 수 없는 상황이 생길 수 있으므로 각속도 명령도 포함되어야 한다. 먼저, 선속도는 (4) 식과 같이 계산할 수 있다. 여기서 단위벡터  $\hat{v}$ 는 초기 로봇 위치와 목표점으로부터 계산한 값이며, 벡터  $\vec{v}$ 는

위와 같은 과정으로 거쳐 터틀봇3의 waypoint 기능을 구상 및 구현하였다.

#### (4) PID제어 구현

PID제어는 Proportional Integral Derivative control의 약자로, 얻고자 하는 출력 값 (controlled variable)이 명령을 받아 실제 System을 변화시키는 명령 값 (commanded variable)과 얼마나 차이가 나는지 (error term)를 계산해 0으로 수렴할 수 있게 해주는 것을 의미한다.

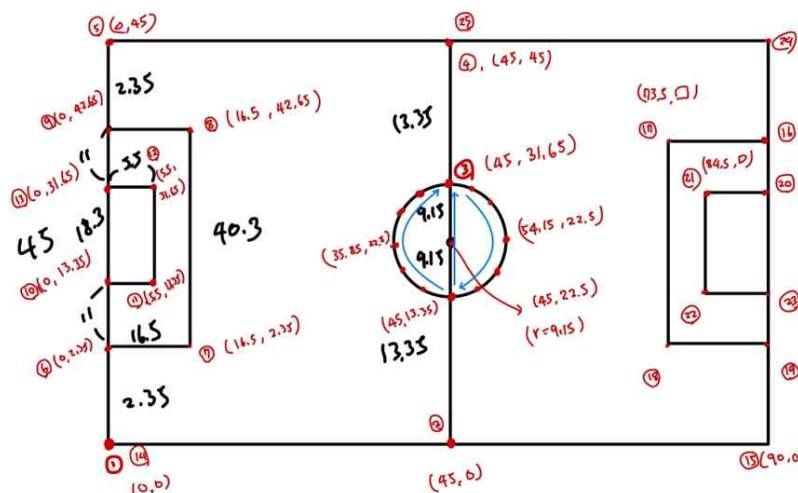


**Fig. 2** A generic closed-loop process-control system with PID controller.

$$MV(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de}{dt}$$

위 그림에서 알 수 있듯, PID제어는 P I D 3종류의 제어를 받는다. 따라서 각 제어에 적절한 이득 값 (gain)을 주어 오차항을 줄일 수 있는데 이 적절한 이득 값을 계산하거나 구하는 과정을 튜닝 (tuning)이라고 한다. PID제어는 Closed loop에서 오차를 이용하여 조절하는 구조로 그 구조가 간단하고, 파라미터 조정을 비교적 쉽게 할 수 있으며, 제어 성능 또한 우수하여 가장 많이 사용된다. 하지만 각각의 Gain을 찾는 과정이 번거롭고 어렵다. 아래는 각 Gain값이 PID제어에서 어떤 역할을 하는지에 대한 설명이다.

아래는 PID제어이론을 기반으로 속도제어를 해 지나야 할 경로들에 대한 설계이다.



**Fig. 3** Design for football field

PID제어에서 축구장, 야구장, 오징어 게임장 총 3개 종류의 field 설계 중 축구장만을 대표 예시로 들겠다.

추가적으로, field를 그리면서 생길 수 있는 돌발 상황으로 갑작스런 장애물이 생기면 감지된 라이더의 개수로 멈추는 소스코드를 제작하였다. 코드에 대한 설명은 다음과 같다. 왼쪽 15개 오른쪽

15개의 범위의 라이다를 센싱하여 3개씩 평균낸 값을 뽑아내면 10개 묶음의 라이다 값을 받아들인다. 그 중 40cm이내의 범위에 들어온 묶음이 4개 이상이 감지되는 동안 정지한다. 그 물체가 사라지면 다시 이동을 재개한다.

#### (5) DQN학습

DQN은 강화학습의 일종으로 주어진 환경에서 agent가 어떤 action을 취하게 했을 때 가장 큰 보상을 받았던 에피소드를 찾아내는 것을 뜻한다. 또한 'Greedy algorithm'을 따른다. Greedy Algorithm을 따르면 2 가지 문제점이 생기는데, 1번째로 '그 상황에서의 최적값이 전체에서의 최적값인지 알 수 없다.'와 2번째로 '지연된 보상'이다.

1번째 문제점의 해결책으로 탐험지수 '엡실론( $\epsilon$ )'이다. 학습된 최적값을 로봇이 알고 있음에도  $\epsilon$  확률에 의해서 무작위 행동을 취하게 하여 또 다른 최적값을 찾을 수 있게 하는 것이다.

2번째 문제점의 해결책은 적절한 discount factor의 할당이다. 현재 좋은 값만 취하게 되면 즉각 보상이 높아지게 된다. 하지만 다른 루트로 갔을 때 잠재적인 미래 가치를 알 수 없다. 또한 좋은 길을 택했음에도 특정 값들을 얻지 못하여 보상이 없는 경우가 있다. 이런 경우들에 즉각보상과 할인율, 미래보상에 대한 DQN의 비용함수를 아래의 방정식과 같이 유도할 수 있다.

$$Cost = \left[ Q(s,a;\theta) - \left( r(s,a) + \gamma \max_a Q(s',a;\theta) \right) \right]^2$$

위 방정식을 참고하여, 터틀봇3에 적용한 parameter들은 아래와 같다.

##### - Action parameter

직진, 우측 각속도, 우측 주행, 좌측 각속도, 좌측 주행 총 5개의 Action size임.

##### - Reward

Collision 시 -1500, 목표 도달 시 3000,

목표점과 가까울 시 (current pose/target pose)  $200 \times 2^{\text{distance rate}}$  등을 주요 보상값으로 잡음.

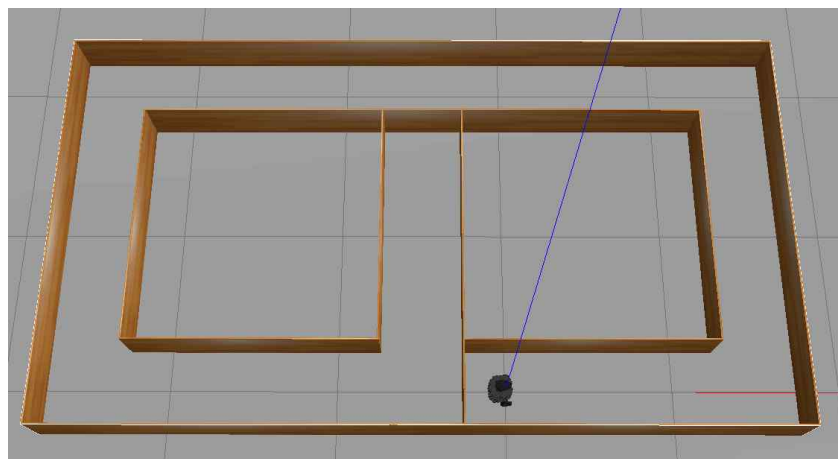
##### - 탐험지수 엡실론( $\epsilon$ )

시작 값=1, episode마다 감소하는 비율 0.99 최소값 0.0001

위 맵을 기반으로 DQN학습으로 간단한 테니스 코트장을 구현하고자 한다.

좌우가 2m x 2m인 정사각형이고 총 4.5m x 2.5m인 직사각형 코트의 경로를 그리며 학습하기 위해서 아래와 같이 벽을 설계하였다.

##### -학습할 map



**Fig. 4** Maps used for dqn learning

#### (6) Hardware 터틀봇3 모델=burger 구현

터틀봇의 SBC인 라즈베리 파이 PC를 고정 IP할당과 Remote PC를 연결해야 한다. 그 과정으로는

PC setup과 SBC setup을 마친 후 두 PC ubuntu에 ROS 의존성 패키지를 설치한 이후 remote PC에 필요한 ros 패키지들을 모두 설치하여야 한다. 이후 remote PC의 Ubuntu와 터틀봇의 라즈베리파이를 원격 연결할 수 있다. 다음으로 remote PC에서 turtlebot3\_bringup 패키지를 실행할 수 있는데 이 패키지에 의해서 실제 터틀봇의 odometry와 battery state, IMU 등등의 터틀봇의 data값들을 가져올 수 있다. 이후에는 gazebo상에서 시뮬레이션 하던 것과 같이 소스코드와 패키지를 실행할 수 있었다. 실제 구현한 사진은 아래와 같고, simulation에서 움직이는 것과 같이 teleop 패키지를 이용해 키보드로 터틀봇을 간단히 조작할 수 있었다.



**Fig. 5** Connecting the actual Turtlebot 3 and the auxiliary battery

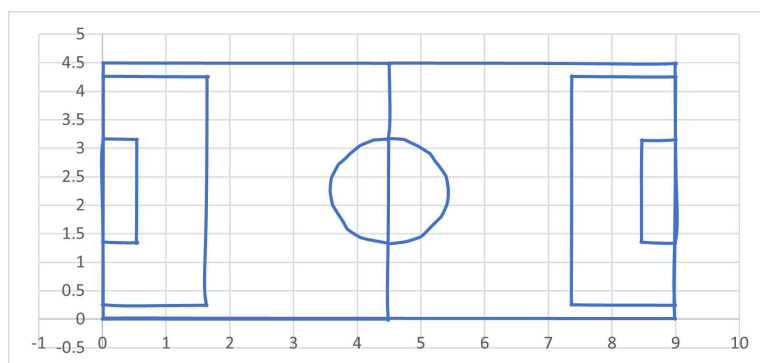
3. 실습비 사용내역  
없음

4. 수행결과

가. 과제수행 결과

(1) PID제어를 통한 Field painting

ROS-melodic에서 gazebo로 구현하여 뽑은 trajectory는 아래와 같다.



**Fig. 6** A soccer field implemented by the trajectory of turtlebot3

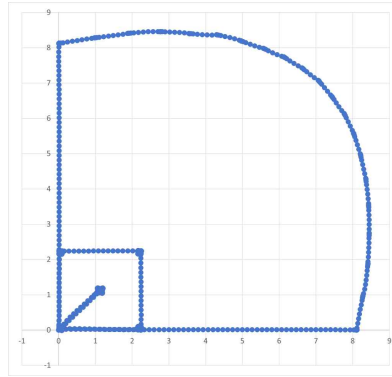


Fig. 7 A baseballfield implemented by the trajectory of turtlebot3

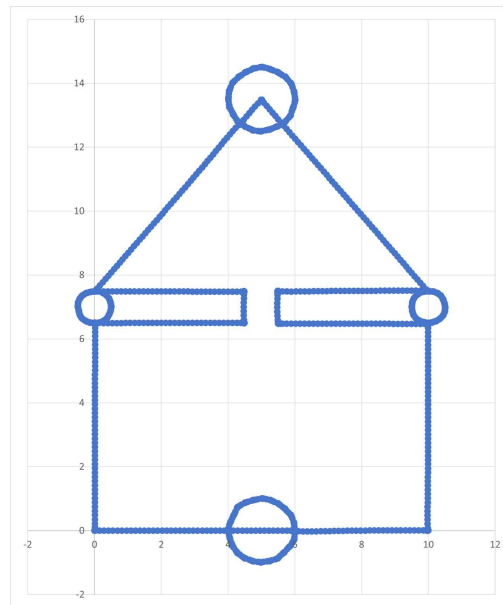


Fig. 8 A squid game field implemented by the trajectory of turtlebot3

## (2) DQN학습을 통한 Field painting

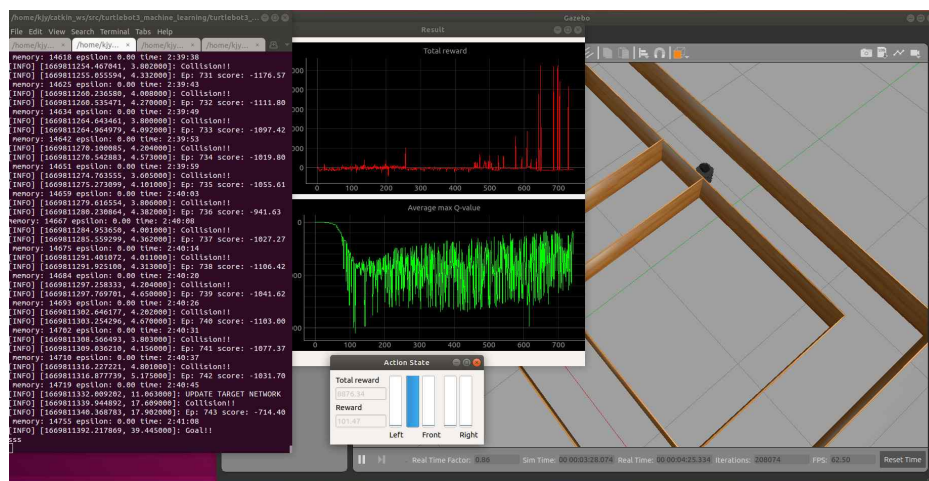
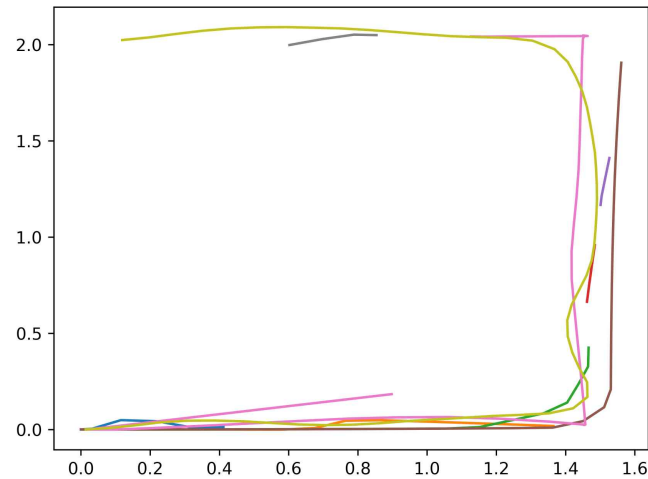


Fig. 9 goal reward 3000 collision reward -1500

gazebo 시뮬레이션 환경에서 터틀봇이 학습하며 맵을 나아가 경로를 그리는 모습이다. reward는 가장 크게는 도착 보상과 충돌 보상을 가지며, yaw축이 회전한 정도, goal box와 가까워지는 정도 생존한 시간 등을 종합적으로 고려해 총 보상값을 적용했으며 상단 그래프가 reward에 대한 값을 나타낸다. 또한 아래의 그래프는 DQN모델의 비용함수 값의 파라미터 중 하나이며, state와 action의 조합값인

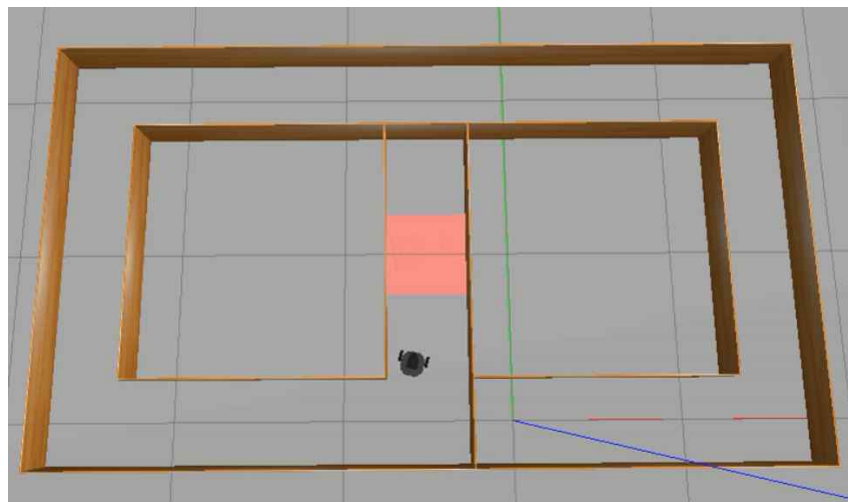
Q-value이다.

학습으로 얻어진 경로는 아래와 같다.



**Fig. 10** trajectory graph obtained from DQN learning

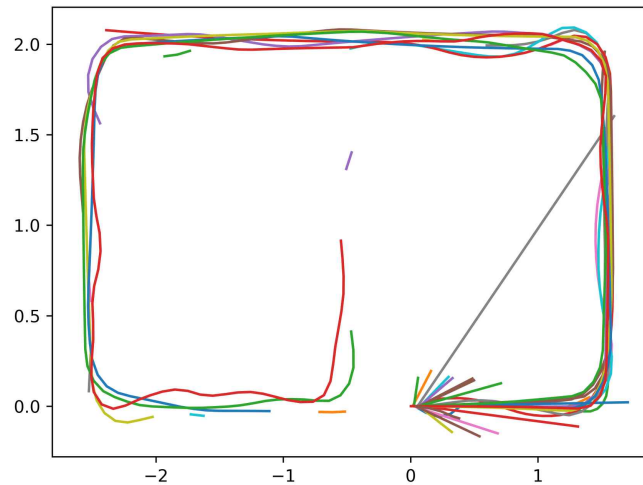
설정된 목표점 (0.5, 0) (1.0, 0) (1.5, 0) (1.5, 0.25) (1.5, 0.5) (1.5, 0.75) (1.5, 1.0)....에 대해 성공하면 trajectory를 뽑아주는 코드이다. 시뮬레이션 상의 가제보는 벽에 충돌하게 되면 다시 (0, 0)으로 돌아가므로 경로가 길게 늘어서 있는 경우는 터틀봇이 리스폰된 (0, 0)에서부터 도달한 경로이고, 짧게 끊겨있는 경로는 목표점에 도달한 이후 벽에 충돌하지 않고 바로 도달한 경로이다.



**Fig. 11** Turtlebot trying to reach the target box through learning

이후 굉장히 많은 episode를 통해 학습한 터틀봇이 마지막 목표상자를 타겟으로 접근하는 모습이다. 학습하면서 생긴 경로들은 1번 목표점부터 총 26번 목표점까지 모두 누적하여 아래의 그래프와 같이 나타내었다.





**Fig. 12** Final trajectory of tennis court created by DQN learning

위 시뮬레이션 사진에서  $(-0.5, 1.0)$  점과  $(-0.5, 1.5)$  목표점을 한번에 통과하여 그 경로가 이산적으로 나타났다. 경로가 연속적이지 못한 한계가 있었지만, 원하는 목표점을 학습을 통해 모두 경유할 수 있었던 점에 큰 의의가 있었다.

위는 DQN학습을 통한 field painting으로 대략적인 테니스 코트를 trajectory를 통해 그려본 모습이다. 40000번이 넘는 episode를 통해 목표점에 도달한 경로들만 골라 뽑은 그래프이다. 학습하는 모델 특성상 굉장히 많은 시도를 해야하는데 실제 터틀봇에서 여러번의 시도에는 제한이 있어 시뮬레이션으로만 구현하였다.

### (3) 실제 터틀봇 구현



**Fig. 13** The actual Turtlebot drawing a field

### 나. 최종결과물 주요특징 및 설명

#### (1) 영상 제출물

<https://youtu.be/1fdHOB573G4>

00:04~04:43 gazebo 시뮬레이션 환경에서 PID제어로 축구장 그리기

04:44~08:29 gazebo 시뮬레이션 환경에서 DQN 학습모델이 학습하는 모습

08:30~11:24 실제 turtlebot3 모델=burger가 사각형 트랙을 그리는 모습과 장애물 회피영상

#### (2) 발표자료

power point로 요약함.

### (3) 터틀봇3

Raspberry Pi PC 및 360 Laser Distance Sensor LDS-01가 부착된 작동 가능한 터틀봇 하나의 고정된 IP에서 잘 작동하기 때문에 학교 wi-fi와 같은 유동 IP에서는 잘 작동하지 않음.

## 5. 기대효과 및 활용방안

### 가. 기대효과

아스팔트의 차선 그리기 뿐 아니라 축구장, 야구장, 오징어게임장, 테니스 코트장 등 여러 방면에서 터틀봇을 이용해 선을 그릴 수 있다. 또한 특정 상황에서 사람이 지나가거나 갑작스럽게 지나가는 장애물을 라이다 센서로 감지하여 멈추는 기능을 추가함으로 높은 안정성도 기대할 수 있다.

### 나. 활용방안

본 과제에서 수행한 축구장, 야구장, 오징어게임장, 테니스 코트장 뿐만 아니라 아스팔트 도로에 차선 그리기 등 원하는 규칙만 입력해주면 스스로 학습 또는 제어를 통해 원하는 field를 만들어주는 로봇을 제작할 수 있다. 만들어진 코드에 waypoint만 입력해주면 되므로 그 활용도는 무궁무진할 것으로 예측된다.

## 6. 결론 및 제언

원하는 waypoint를 따라 trajectory를 뽑아주는 터틀봇을 2 가지 방법 PID제어와 DQN학습모델로 구현하였다. 처음으로, PID제어는 터틀봇의 속도 명령만으로 위치 제어를 하므로 간단한 field를 구현하는 데는 굉장히 효율적이다. 복잡한 개형의 경로라도 정확한 위치값과 속도값을 odometry값으로 잘 얻을 수 있다면 어떠한 field를 구현하는데도 문제가 없다. 다음으로는 DQN 학습모델을 활용한 머신러닝 방법이다. 학습할 수 있는 환경이라면 어떠한 map도 최적경로를 학습하여 그려낼 수 있다는 장점이 있다. 하지만, 굉장히 많은 학습을 해야하므로 실제 구현하기 어려운 문제점이 있었으며 충분한 학습이 되지 않으면 그마저도 깔끔한 경로가 나오지 않는다는 한계점이 있었다.

위 학습을 기반으로 실제 터틀봇에서도 waypoint에 따라 field를 그려주도록 구현하였으며 머신러닝의 한계에 따라 PID제어 기반으로 완성하였다. 결과적으로는 터틀봇으로 2m x 1m의 축구장을 큰 오차 없이 그려낼 수 있었다.

### [참고자료]

- (1) Garrido, Alvarez L. (2019). ROBOTIS e-Manual. [online] ROBOTIS e-Manual. Available at: <http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>
- (2) Garrido Alvarez, L. (2019). Documentation - ROS Wiki. [online] Wiki.ros.org. Available at: <http://wiki.ros.org/Documentation>
- (3) YoonSeok Pyo, [git-hub] <https://github.com/robotpilot>
- (4) Biswaw Arijit, 2009/01/01, Design of fractional-order PID m controllers with an improved differential evolution, Eng. Appl. of AI, 22, p. 346
- (5) Nes,et Un`ver Akmandor, Hongyu Li, Gary Lvov, Eric Dusel and Tas,kin Padır (2022) Deep Reinforcement Learning based Robot Navigation in Dynamic Environments using Occupancy Values of Motion Primitives
- (6) Leonardo Garrido, Carlos Aguirre, Cameron Thompson, M. F. De La Torre, V. Behzadan, W. Hsu, (2019) DQN: Deep Q-Learning for Autonomous Navigation

※ 본 양식은 요약보고서이며, 최종결과물을 추가제출 필수

팀 대표 : \_\_\_\_\_ 김정윤 