로봇네비게이션 중간 프로젝트

조교 한창완

프로젝트 소개

아래의 세 가지 알고리즘을 구현하는 것이 이번 프로젝트의 목표입니다.

1. **AStar 구현**
2. **RRT 구현**
3. **Follower the gap**

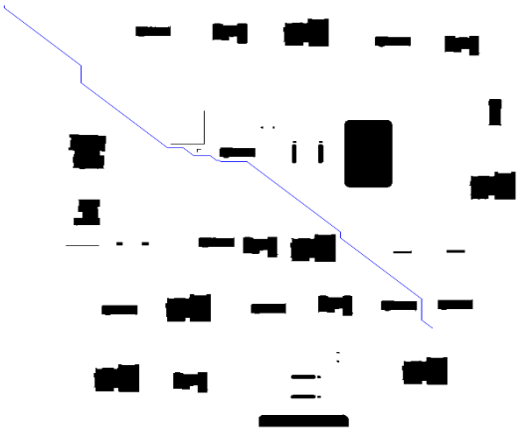
AStar 구현

제공한 압축 폴더에서 astar_planner.py 을 이용해 진행합니다. #TODO가 써 있는 부분에서 주어진 함수들을 이용해 AStar 알고리즘을 구현하면 됩니다.

사용하는 파일은 track.npy로 결과를 확인하시기 위해서는 아래의 명령어를 입력하면 됩니다.

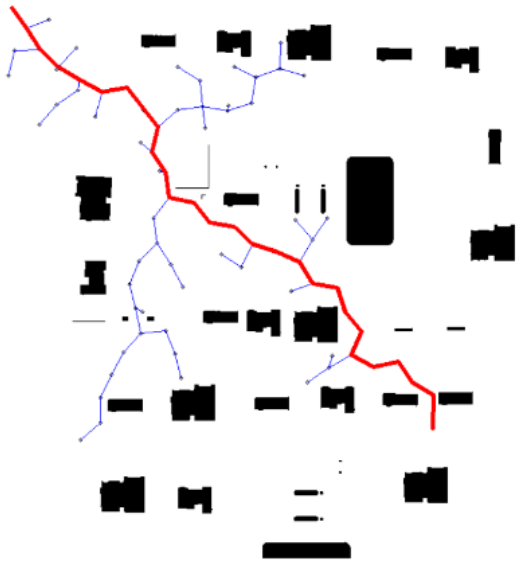
1. **astar_planner.py 이 있는 폴더에서 터미널을 생성합니다.**
2. **python astar_plannar.py track.npy 또는 python3 astar_plannar.py track.npy**

위의 명령어를 입력하시면 아래와 같은 결과를 획득할 수 있습니다. 단, 아래의 결과는 track에서 진행한 것이 아닌 map.pkl에서 진행한 결과입니다.



RRT 구현

제공한 압축 폴더에서 rrt_plannar.py를 이용해 진행합니다. #TODO가 써 있는 부분에서 주어진 함수를 구현하는 것으로 RRT를 생성할 수 있습니다. 이번에는 map.pkl을 이용해 구현하기 때문에 최종 결과는 아래와 같습니다.



아래의 명령어를 통해서 결과를 확인할 수 있습니다.

1. rrt_plannar.py 이 있는 폴더에서 터미널을 생성합니다.
2. python rrt_plannar.py map.pkl 또는 python3 rrt_plannar.py map.pkl

구현이 필요한 함수

 # sample_state(self)

지도에 임의로 포인트를 생성할 때, 사용하는 함수입니다. 이미지의 크기를 넘지 않도록 설정하는 것이 중요합니다. (함수의 입력을 바꾸셔도 상관없습니다.)

 # steer_towards(self, s_nearest, s_rand, max_radius)

경로를 찾기 위해 새로운 포인트를 생성하는 함수입니다. s_rand을 이용해 새로운 합수를 생성하고, s_nearest에서 max_radius보다 먼 경우, 최대 거리에 있는 점을 이용합니다.

 # plan

알고리즘을 진행하는 함수입니다. 위에서 생성한 함수와 기존의 함수를 이용해 진행하면 됩니다. Steer_towards를 이용해 새로운 점을 생성하고 해당 점까지의 경로에 장애물이 있는지 없는지를 파악해야 합니다. 해당 경로에 장애물이 없는 경우, 경로를 저장합니다. 위를 반복해 도착 지점까지의 경로를 생성하면 됩니다.

Follower the Gap 구현

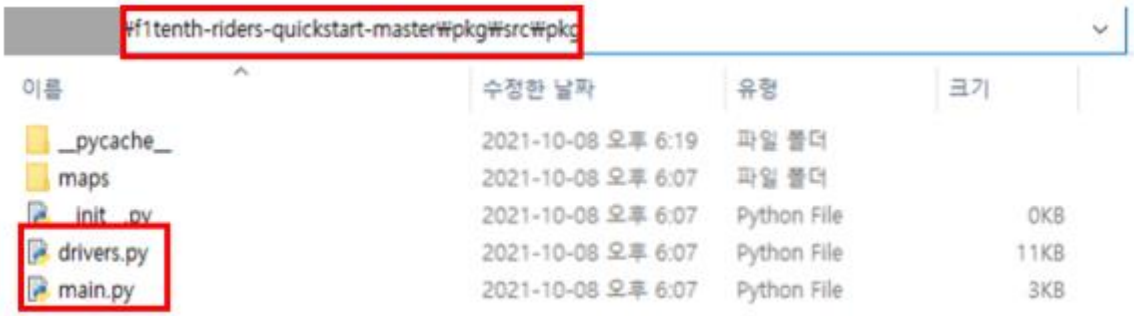
설치 및 확인

주의) 파이썬 및 pip 패키지가 깔려 있다는 가정으로 설명하겠습니다.

1. **다운로드 받은 압축파일을 풀고, f1tenth-riders-quickstart-HW 폴더를 연다.**
2. **해당 폴더에서 터미널을 생성하고 pip install –user -e gym을 입력해 패키지를 설치한다. 안되는 경우, pip을 pip3로 변경해 확인**
3. **현재 경로에서 cd pkg/src를 통해 경로를 변경한다.**
4. **Python -m pkg.main을 통해서 결과를 확인할 수 있다.**
5. **1 바퀴를 완주하는 경우, 종료하게 되며 걸린 시간을 확인할 수 있다.**

과제 진행 방법

1. **'f1tenth-riders-quickstart-master' 폴더에서 'pkg' - 'src' – 'pkg'폴더 아래로 이동한다. 그러면 아래의 사진과 같이 drivers.py와 main.py가 있다.**



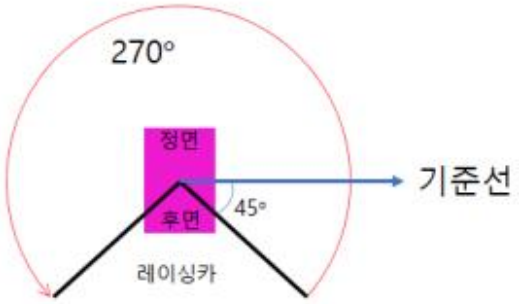
drivers.py는 주행알고리즘들이 담겨있는 파일이고, main.py는 주행알고리즘을 시뮬레이션에

서 테스트하기 위한 파일이다.

2. divers_HW 파일안에는 아래와 같은 기본적으로 process_lidar이라는 함수 를 가진 GapFollower 함수가 있다. 클래스 내에 process_lidar함수는 2D 라이다의 센서값(ranges)을 받는다. 함수의 반환값은 차량의 목표 속도와 스티어링 각이다. (차량은 ackerman steering의 구조를 가진다.

※ **ranges** 값 정의

- 2D라이다의 센서 값은 방위 별 거리값의 배열이며, 리스트 자료형이다.
- 거리값은 우측 방향의 기준선에 대해서 – 45도부터 225도까지 순서대로 리스트에 저장되어 있고, 0(m)에서 35(m)사이이다.
- 라이다 분해능은 0.25도 이며, 리스트의 크기는 1080개이다.



※ **steering_angle** 정의

`steering_angle` : an angle in the range [-π/2, π/2], i.e. [-90°, 90°] in radians, with 0° meaning straight ahead.

3. GapFollower 함수를 완성해 트랙을 한 바퀴 완주하면 된다.

제출 방법

astar_planner.py, rrt_plannar.py, divers_HW.py와 Astar를 진행한 이미지, rrt_plannar를 진행하는 동영상을 압축해서 "**학번_이름.zip**"제출하면 됩니다.

주의 사항

1. 알고리즘 및 함수를 코드를 생성할 때, **작성한 코드의 한 줄, 한 줄마다 무슨** 내용인지 주석을 필수로 달아야 합니다. 확인해서 없으면 감점하겠습니다.
2. 추가적으로 제출할 파일이나 환경설정이 있다면, 그에 따른 사용방법도 같이 보내야 합니다. (pdf형태)
3. 제출 기한은 **11월 2일 23:59** 까지 입니다.
4. 문의 및 코드 제출은 아래의 이메일로 합니다.

   hcw511@naver.com