자율주행프로그래밍






팀 장희빈

충돌 후 복귀 시스템 MANUAL

TinyLidarNet & PP 를 활용한 충돌 후 raceline 복귀 시스템

박수빈, 장수인, 최희우

20241966, 20243298, 20241992

제안서
목차

매뉴얼 목적

본 문서는 교수님께서 직접 저희 프로젝트 코드를 문제 없이 빌드하고 실행할 수 있도록 작성된 상세 매뉴얼입니다.

단 한 단계라도 빠지면 빌드 실패 가능성이 있어, 모든 과정을 순서대로 따라주셔야 합니다.

Ubuntu / Docker 환경을 기준으로 작성되었습니다.

0. 컨테이너 생성 및 준비

컨테이너 생성 및 준비

1) 도커 컨테이너 생성 및 실행

[Terminal 1]

subin@subin-16Z90SP-KDOVK:~$ ./dkrun.sh misys:forza_full

misys@subin-16Z90SP-KDOVK:~$ cd forza_ws/race_stack/

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ ./.activate_forza.sh

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ source /opt/ros/humble/setup.bash &&
source install/setup.bash

도커 컨테이너 준비

2) 다른 터미널에서 컨테이너 접속하는 방법

[Terminal 2,3,4]

subin@subin-16Z90SP-KDOVK:~$ docker exec -it [본인 컨테이너 ID] bash

misys@subin-16Z90SP-KDOVK:~$ cd forza_ws/race_stack/

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ source /opt/ros/humble/setup.bash &&
source install/setup.bash

1. 소스 파일 다운로드 및 빌드

소스 파일 다운로드 및 빌드

1) 소스 파일 다운로드

1. collision_detector (폴더)
2. controller_switcher (폴더)
3. inference_node.py
4. pp.hpp
5. pp.cpp
6. base_classes.py
7. localize.yaml
8. particle_filter.py
7. localize.yaml(GPU가 없는 경우 사용)
8. particle_filter.py
9. sim.yaml
10. state_machine (폴더)
11. ofc_enabled_tln.launch.py

3개의 폴더와 8개의 파일을 다운로드 해 준비해주세요.
저는 shared_dir에 모든 파일과 폴더들을 다운받았다는 가정하에 진행하겠습니다.

소스 파일 다운로드 및 빌드

1) 소스 파일(폴더) 이동 방법

[Terminal 1] 파일을 이동시킬 경우

다운로드한 파일 경로

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ cp ~/shared_dir/base_classes.py ~/forza_ws/race_stack/base_system/f110_simulator/f1tenth_gym/gym/f110_gym/envs

이동해야 할 경로

[Terminal 1] 폴더를 이동시킬 경우

다운로드한 폴더 경로 / 이동해야 할 경로

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ cp -r ~/shared_dir/collision_detector ~/forza_ws/race_stack

이런 방법으로 파일과 폴더를 이동시키면 됩니다.
다운받은 파일과 폴더가 이동해야 할 경로는 다음 장에 있습니다.

소스 파일 다운로드 및 빌드

2) 소스 파일(폴더)가 이동해야 할 경로

~/forza_ws/race_stack/base_system/f110_simulator/f1tenth_gym/gym/f110_gym/envs/base_classes.py

~/forza_ws/race_stack/collision_detector

~/forza_ws/race_stack/controller_cpp/include/pp.hpp

~/forza_ws/race_stack/controller_cpp/src/pp.cpp

~/forza_ws/race_stack/controller_switcher

~/forza_ws/race_stack/ofc/launch/ofc_enabled_tln.launch.py

~/forza_ws/race_stack/ofc/ofc/inference_node.py

~/forza_ws/race_stack/particle_filter/particle_filter/particle_filter.py

~/forza_ws/race_stack/state_machine

~/forza_ws/race_stack/stack_master/config/SIM/sim.yaml

~/forza_ws/race_stack/stack_master/particle_filter/config/localize.yaml (이 파일은 gpu 가 없는 컴퓨터의 경우 사용해주세요.)

소스 파일 다운로드 및 빌드

3) 빌드 하기

[Terminal 1]

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ colcon build packages-select particle_filter

misys@subin-16Z90SP-KDOVK:~/forza_ws/race_stack$ source install/setup.bash

내용이 수정된 파일의 패키지

이런 방법으로 수정한 파일들이 있는 폴더를 빌드해야 합니다.
빌드 후 source install/setup.bash 반드시 해주세요.

2. 터미널별 명령어 실행

터미널1 명령어

1) 터미널 1 명령어 및 주의사항

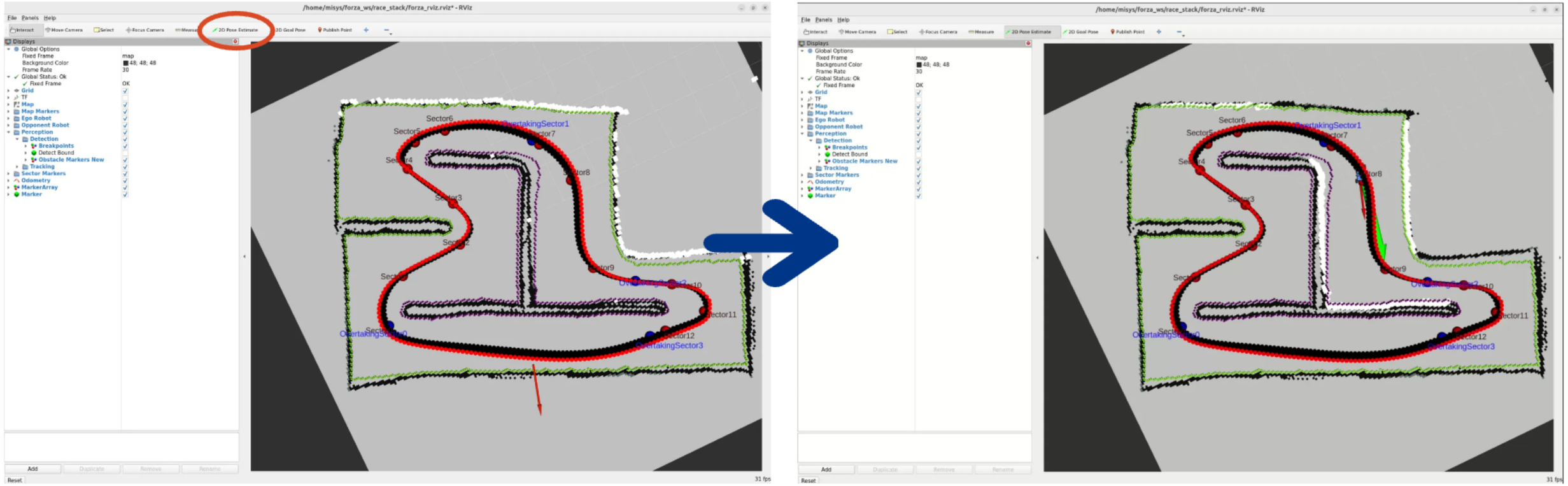
[Terminal 1] 시뮬레이터와 base_system을 실행해 차량 환경을 초기화

misys@subin-16Z90SP-KDOVK:~$ cd forza_ws/race_stack$ ros2 launch stack_master base_system_launch.xml racecar_version:=OrinNano map_dir:=small_hall map_name:=small_hall_orig sim:=true

반드시 가장 먼저 실행해야 합니다.

터미널1 명령어

1) 터미널 1 명령어 및 주의사항



2D Pose Estimation 으로 지도 화면을 그어주면 화살표와 함께 원하는 시작 위치와 방향을 설정할 수 있습니다.

터미널2 명령어

2) 터미널 2 명령어 및 주의사항

[Terminal 2] 주행 알고리즘과 state machine을 실행해 차량의 주행 로직을 담당

misys@subin-16Z90SP-KDOVK:~$ cd forza_ws/race_stack$ ros2 launch stack_master time_trials_launch.xml racecar_version:=OrinNano ctrl_algo:=PP

터미널3 명령어

3) 터미널 3 명령어 및 주의사항

[Terminal 3] LiDAR·IMU 데이터를 기반으로 충돌 여부를 판단해 publish

misys@subin-16Z90SP-KD0VK:~$ cd forza_ws/race_stack$ ros2 run collision_detector collision_detector_node

터미널4 명령어

4) 터미널 4 명령어 및 주의사항

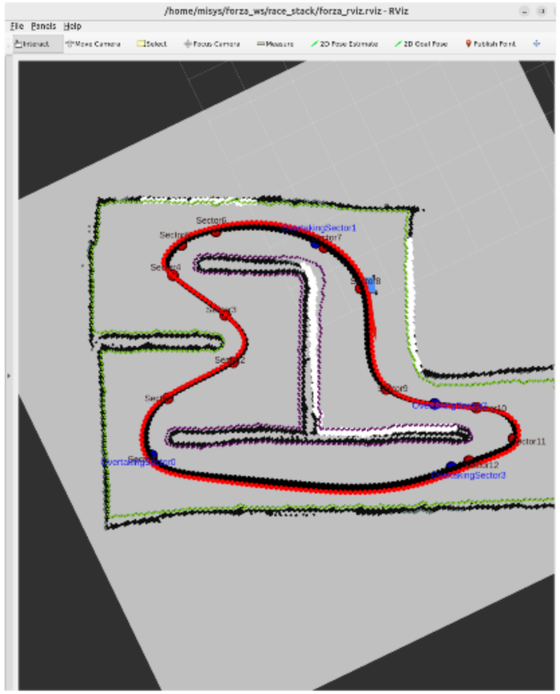
[Terminal 4] 상태에 따라 적절한 제어기를 선택해 최종 차량 제어 명령을 출력

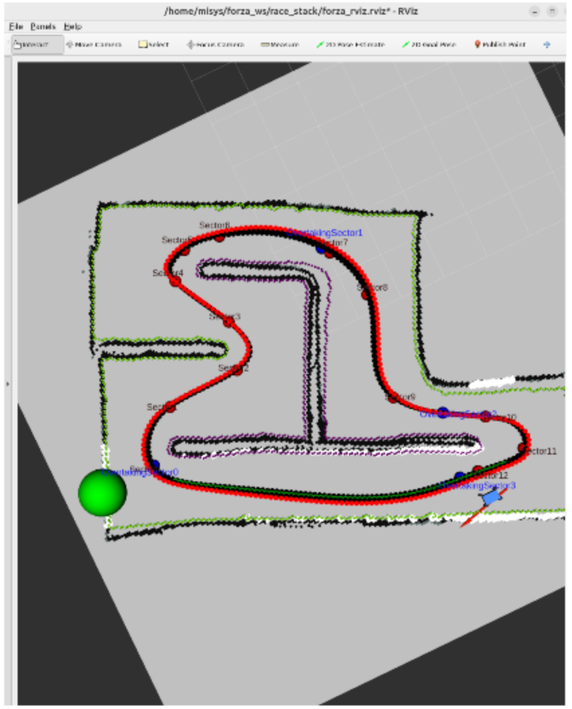
misys@subin-16Z90SP-KDOVK:~$ cd forza_ws/race_stack$ ros2 launch controller_switcher controller_switcher.launch.py

터미널4번은 2,3번 터미널을 먼저 실행시킨 후
차가 출발하고 왼쪽 하단에 초록색 원이 생겨나면 실행시켜주세요.

터미널4 명령어

4) 터미널 4 명령어 및 주의사항





터미널4번은 2,3번 터미널을 먼저 실행시킨 후 차가 출발하고 왼쪽 하단에 초록색 원이 생겨나면 실행시켜주세요.

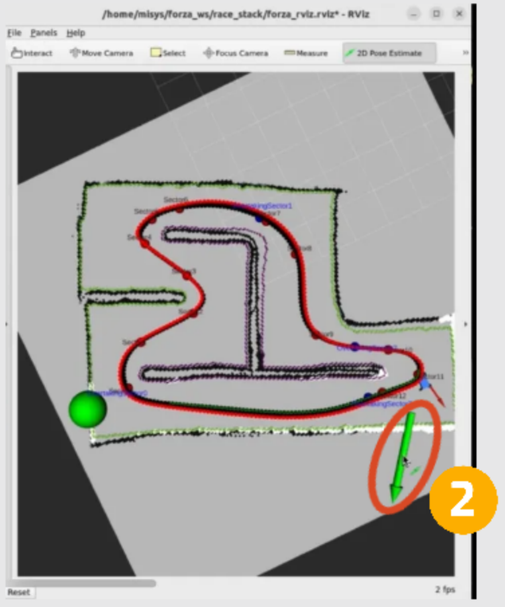
3. 충돌 유발 및 결과 확인

충돌 유발 및 결과 확인

1) 2D Pose Estimation을 벽을 향해 긁어 충돌을 유발

1. 정상적인 주행을 하는 상태

2. 2D Pose Estimation을 의도적으로 벽을 향해 긁는 모습



3. 충돌이 일어나 RECOVERY 모드로 바뀐 상태 (결과 확인)

만약 차가 주행 중 충돌이 일어나지 않는다면 차가 벽 가까이 다가갔을 때 의도적으로 2D Pose Estimation을 벽을 향해 그어 저희의 결과물을 확인해주세요.
저희 프로그램은 충돌 전 주행 상황에서 pose가 정상적으로 잡혀있는 상황을 가정하기 때문에 2D Pose Estimation을 그어주는 과정에서 잘못된 곳에 파티클이 수렴하게 된다면 다시 잡아주셔야 합니다.

감사합니다

Thank you