SLAM을 이용한 장애물 탐지 및 맵핑 시스템

E-2

목차

- 매핑 로직
- 튜닝 전략
- 맵핑

목적

ROS 2를 활용하여 Turtlebot4가 환경을 탐색하고 장애물 및 미지의 영역을 효율적으로 탐험하도록 설계된 planner 개발

매핑 로직

```
map_data = np.array(msg.data, dtype=np.int8).reshape(height, width)
```

1. [맵 데이터 처리 ; Topic] map 정보 받기 (map 한 행씩 받아와서 2차원 numpy 배열로 변환) ⇒ topic으로 /bot2/map정보 (OccupancyGrid 메시지) subscribe 해서 데이터 가져오기 (0, -1, 100) => 2초 간격으로 맵 업데이트 및 목표 계산
   - 목표점 좌표 변환 (Grid 좌표 → 실제 월드 좌표)

2. [목표점 선정] 좌표값 계산

   - 0인 부분이 맵이 생성된 곳, -1인 부분이 미탐사 영역,100이 테두리(벽)
   - [BFS 탐색] 3x3 픽셀에서 가운데 로봇을 기준으로, 주변에 -1 값이 보이면 목표점 선정. (단, 0과 -1 사이에 100도 있다면, 벽을 뛰어넘는게 되기 때문에 그리지 말도록)

매핑 로직

- [추가로직 ; 통로 중심점 탐색 알고리즘]
  i. 3*3px로 자른 맵에서 [벽, 확인, 미확인] 지역이 모두 있으면서 확인, 미확인 지역이 붙어있는 곳을 벽의 끝으로 판단.
  ii. 모든 벽의 끝점들을 서로 이은 선들의 끝 3px을 제외한 모든 위치에서, 해당 좌표의 3*3px 자른 맵에 벽이 없는지 확인하고, 벽이 없으면 그 선의 중간점을 확인해야할 목표 리스트에 추가.
  iii. 추가된 모든 목표 리스트를 반환

3. [내비게이션 제어 ; Action]

action을 통해 매시간 계산한 좌표를 받아 -1인 지점으로 이동시키기 (NavigateToPose 액션 클라이언트)

튜닝 전략

FollowPath

```
FollowPath:
    ...
    max_speed_xy: 0.26 ⇒ 0.5
```

속도 빠르게 변경

튜닝 전략

local_costmap

```
local_costmap:
      ...
      ...
       cost_scaling_factor: 4.0 ⇒ 5.0
       inflation_radius: 0.45 ⇒ 0.30

      ...
       obstacle_max_range: 2.5 ⇒ 3.5
```

장애물 근처에서 코스트를 급격히 증가시켜
로봇이 장애물 근처로 가지 않도록

장애물 주위의 인플레이션 반경(안전 구역의 거리)을
좁게

로봇 센서가 감지하는 최대 장애물 범위를 증가

튜닝 전략

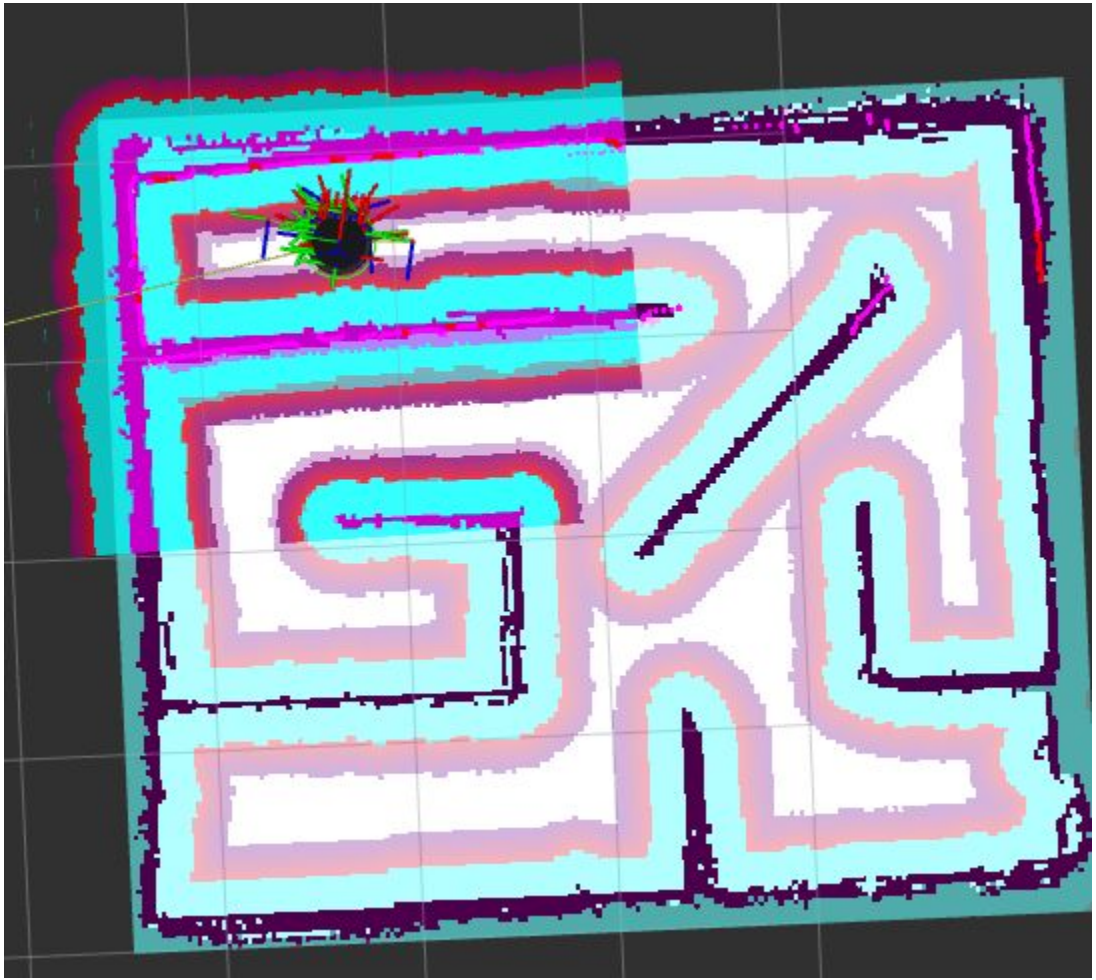
global_costmap

```
global_costmap:
      ...
      ...
        cost_scaling_factor: 4.0 ⇒ 5.0
        inflation_radius: 0.45 ⇒ 0.35
```

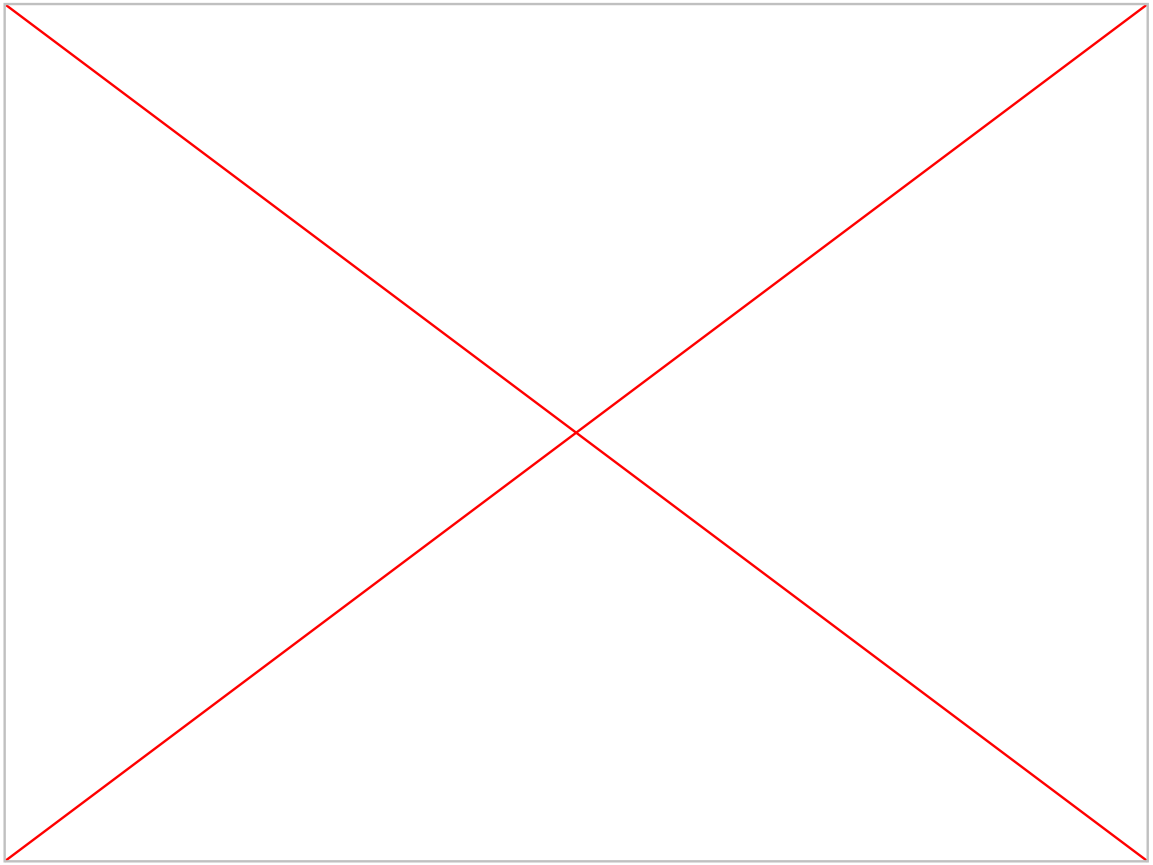
장애물에서 가까워질수록 코스트를 급격히 증가시켜 근처로 접근하지 못하도록

장애물 주위의 인플레이션 영역(안전 거리)을 설정하여 로봇이 충돌 없이 통과할 수 있도록 보호 구역을 생성

매핑 결과



매핑 rviz



매핑 현장