




TurtleBot4

로봇 청소기

A-4조
정승환 김승중 장준하 윤민식

OUTLINE

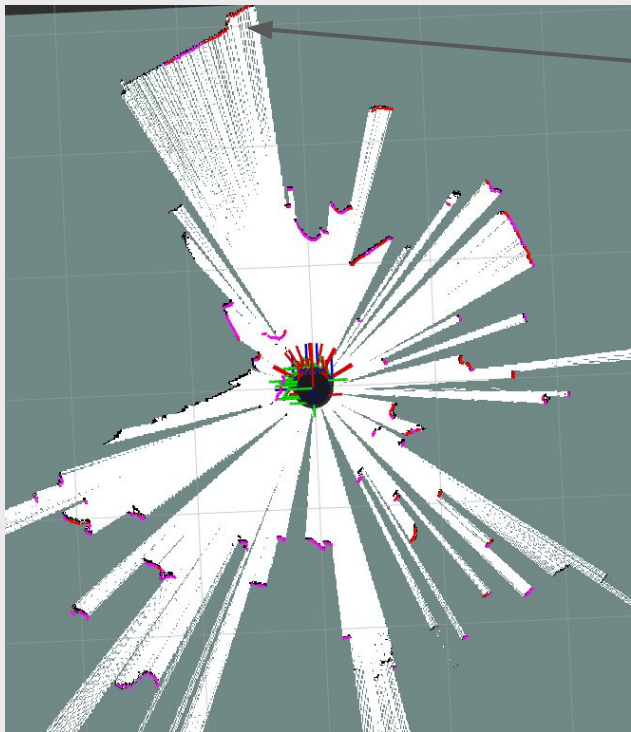
- 
1. Purpose
 2. Logic
 3. Parameter Tuning
 4. Encountered Situations
 5. Overcoming Directions
 6. QnA

Purpose



TurtleBot4를 활용한 자율 주행 로봇청소기 알고리즘
개발
효율적인 경로 계획

Logic



1. 현재 매핑되어있는 map을 기준으로
로봇 좌표상 가장 먼 곳을 찾는다
2. 해당 좌표에 도착 후,
다시 현재 map에서 가장 먼 곳을 찾는다.
3. $[-1,0]$ 이 없을 때까지 1, 2번을 반복한다.

| Logic

[-1, 0] 쌍의 근처에 0이 일정 수 이상 있을 때
탐지

-1	-1	-1
0	-1	0
0	0	-1

근처에 0이 4개

탐지 X

-1	-1	-1
0	-1	0
0	0	0

근처에 0이 5개

탐지 O

| Logic

종료 조건: 0 근처에 -1이 없다면 종료

-1	-1	-1
0	-1	100
0	100	0
0	0	0

종료 X

-1	-1	-1
100	-1	100
100	100	0
0	0	0

종료 O

| Logic

‘/map’ topic callback function:

- : ‘/map’ topic을 받으면 data를 저장**
- : ‘/odom’ topic을 받으면 현재 터틀 봇의 위치를 저장**
- : 터틀 봇의 위치를 map data의 행렬 인덱스로 변환 (좌표 / resolution)**
- : [0, -1] 쌍의 모든 좌표를 찾아서, 각 쌍과 거리를 측정 (값이 0이며 근처에 -1이 있는지 확인)**
- : if [0, -1] 쌍이 하나도 존재하지 않는다면 break**
- : 가장 먼 [0,-1] 쌍을 찾아 터틀 봇 좌표로 변환
(리스트에 좌표와 거리를 저장, pose = poses[dists.index(max(dists))])**
- : ‘/goal_pose’ topic을 보내 터틀 봇을 목적지로 보낸다.**

| Logic (로봇 청소기)

: 0과 100의 경계 픽셀을 모두 감지한다.

: 경계 안쪽 픽셀 테두리를 만난다.

while :

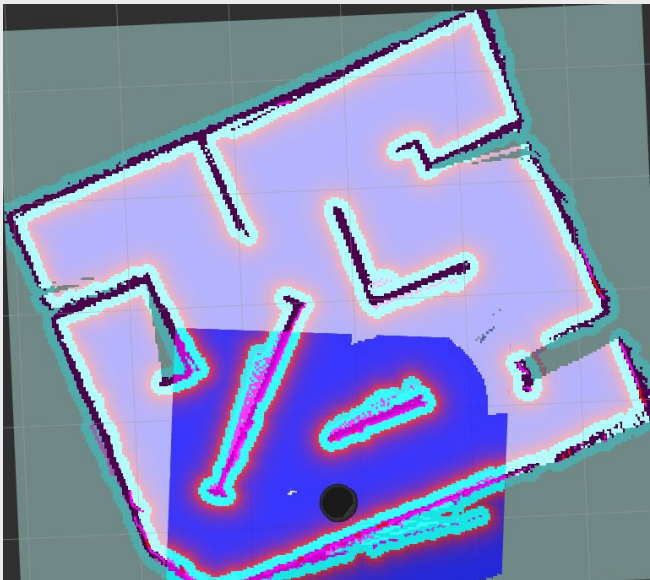
: 가로 단위로 달힌 구간의 경계 값을 구한다.

: 오른쪽 -> 왼쪽 -> 위 -> 아래 순으로 달힌 구간을 탐색한다.

: 가장 가까운 달힌 구간으로 로봇을 이동시킨다.

: if 달힌 구간이 남지 않았다면 **break**

Parameter



1. map 이동 경로에 **cost**가 하늘색 수준으로
설정되게 튜닝
2. 코너링할 때, 장애물에 너무 타이트하게
붙지
않도록 튜닝

Parameter

```
slam_params = RewrittenYaml(  
    source_file=LaunchConfiguration('params'),
```

```
nav2_params = LaunchConfiguration('params_file')
```

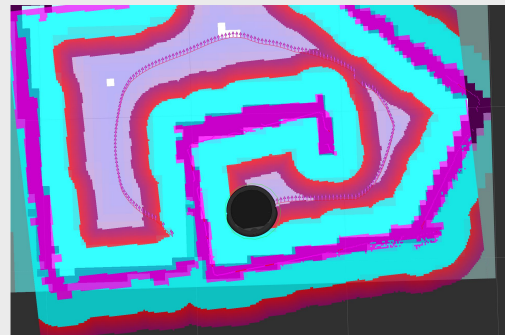
resolution: 0.05



resolution: 0.01

[[[nav2 실행 순서]]]

```
ros2 launch turtlebot4_navigation slam.launch.py params_file:=slam.yaml  
ros2 launch turtlebot4_viz view_robot.launch.py  
ros2 launch turtlebot4_navigation nav2.launch.py params_file:=nav2.yaml
```



Parameter

Tuning

local_costmap: 로봇 주변의 짧은 경로를 실시간으로 계획하고, 장애물 회피

global_costmap: 전체 맵을 기반으로 목표 지점까지의 경로를 계획

cost_scaling_factor

장애물에서 떨어지는 거리에 따라 위험도(cost)를 얼마나 빨리 증가시킬지를 결정하는 값

>> 10.0

inflation_radius

장애물로부터 얼마나 멀리 떨어져야 하는지를 결정하는 거리

>> 0.55

| Parameter

Tuning

general_goal_checker:

xy_goal_tolerance

>> 0.3

yaw_goal_tolerance

>> 6.28

목표 위치에 도달했는지 확인하는 값

FollowPath:

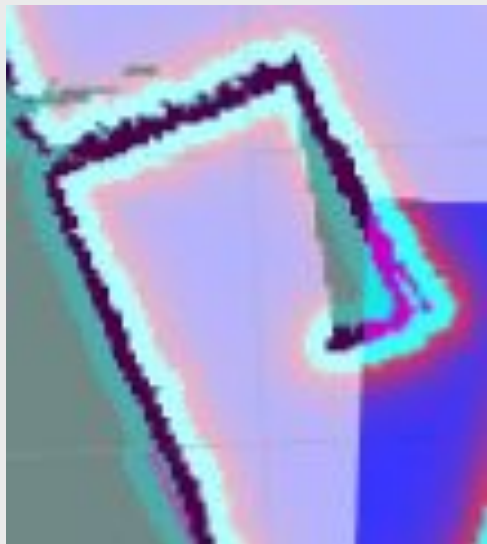
sim_time

>> 2.0

로봇이 앞으로 얼마나 오랫동안 시뮬레이션할지를 결정하는 시간
(더 긴 예측 시간을 사용하면 더 부드러운 경로가 생성)

| Parameter

#1. 좁은 공간에서 회전하지 못함



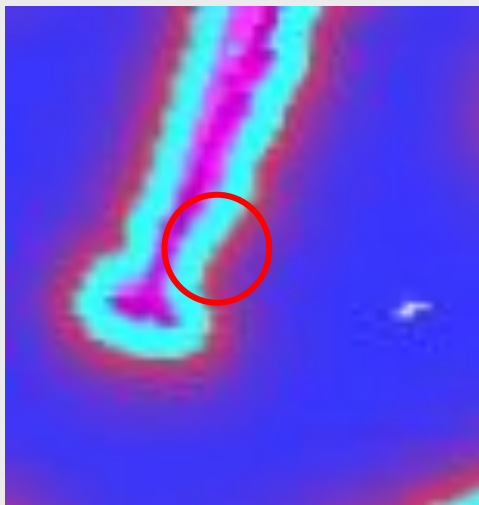
cost map에서 장애물 영역을 최소화

이동 가능 영역을 극대화 시킴

solve

| Parameter

#2. 움푹 파인 곳을 찍으면 도달이 되지 못한다.



goal 인정 반경을 크게 주다 보면 근처에 도달한 순간
목적지에 도달했음을 알리는 topic이 날라온다.

하지만 움푹 파인 곳에 $[-1,0]$ 좌표 쌍이 존재하기
때문에 계속 동일한 좌표를 찍게 되는 로직 문제가
발생

can't
solve

| Parameter

#3. 특정 지점에서 고개를 흔든다.

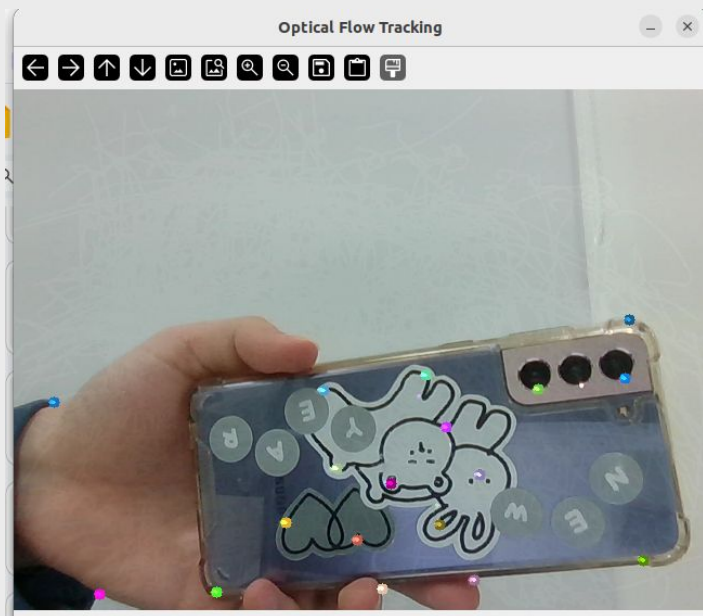


??

이유를 찾지 못했습니다.

**can't
solve**

tracker



| THE END