로봇청소기 시스템

C-1

김교희 | 김유찬 | 박병주 | 박영건 | 오동규

index

시나리오

1. 자동매핑

2.구역 청소

3.청소 완료

Tutlebot4
Hardware
On-board Computer:Raspberry Pi 4B 4GB
Lidar: RPLIDAR A1M8
Camera: OAK-D-Pro

ROS 2 Resources
Nav2 Github: https://github.com/ros-planning/navigation2
SLAM Toolbox: https://github.com/SteveMacenski/slam_toolbox

Slam.yaml

ROS__parameter

```
resolution: 0.05 -> 0.03
scan_buffer_size: 20 -> 30
```

resolution: 1 픽셀이 나타내는 실제 거리 (m/pixel), 격자맵 해상도

- 1 pixel 당 3cm 로, 해상도를 높여 더 자세한 경로 계획 가능
- 해상도 너무 높으면 CPU, RAM 사용량이 증가, 속도 하락
- 맵 크기와 복잡함 정도에 따라 타협

scan_buffer_size : LiDAR 데이터 저장 버퍼 크기

- 기존 20->30개로 증가시켜 SLAM 성능 향상, 하지만 실시간 성능 하락으로 이어지기에 타협

Slam.yaml

Loop Closure Parameters

```
# Correlation Parameters - Loop Closure Parameters
loop_search_space_dimension: 8.0 -> 5.0
loop_search_space_resolution: 0.01 -> 0.05
loop_search_space_smear_deviation: 0.1 -> 0.05
```

loop_search_space_dimension:루프 클로저 탐색 공간 크기

- 좁은 실내에선 루프클로저 감지 넓이를 줄임

loop_search_space_resolution : 루프클로저 탐색 해상도

- 기존 1cm에서 점점 증가하며 성능 하락 차이 관찰, 반비례하는 탐색 속도와 정밀도 사이에서 타협

loop_search_space_smear_deviation:루프 클로저 감지

실내공간 자체가 노이즈가 많지 않아 기존보다 높임 노이즈 허용 범위

nav2.yaml:경로 최적화

global_costmap

```
cost_scaling_factor: 4.0 -> 2.0
inflation_radius: 0.45 -> 0.20
resolution: 0.05 -> 0.03
```

cost_scaling_factor : inflation_radius 진입시 발생, 증가하는 비용 계수

- 장거리 경로 계획 시 좁은 공간에서 로봇 반경 만큼의 공간이 없어 비용 적절히 낮추는 것 중요
- 정밀한 경로탐색(global)vs장애물 탐지능력(local)

inflation_radius : 전체 경로 상 장애물 반경 확산 반경

- 넓은 범위에서 안정적 경로 설정을 위해 Global>local

resolution: 맵 정확도를 위해 증가

nav2.yaml

local_costmap

```
width: 3 -> 1
height: 3 -> 1
resolution: 0.06 -> 0.01
robot_radius: 0.175 -> 0.165
```

```
cost_scaling_factor: 4.0 -> 3.0
inflation_radius: 0.45 -> 0.20 -> 0.13
```

width, height: 장애물을 감지하는 가로 세로 범위(m)
- 좁은 공간에서 1m로 줄여 장애물 회피 반응속도 최적화

resolution: 로컬 맵 해상도
- 로컬 경로 생성 정확도 ↑
- 정밀한 장애물 감지

robot_radius: 로봇 반경, 실제로 turtlebot4는 가로보다 세로가 작음

cost_scaling_factor: 로컬 장애물 비용 증가 계수
- 실시간 장애물 회피 능력에 중점, 로컬 환경에서 증가하여 충돌 위험↓

inflation_radius: 로컬 경로 상 장애물 확산 범위
- 실시간 장애물 회피 반응↑, 로컬 플래너의 장애물 회피 경로 유연한 변경

nav2.yaml

timestamp

```
raytrace_max_range: 3.0 -> 2.5
raytrace_min_range: 0.0
obstacle_max_range: 2.5 -> 2.0
```

#obstacle< raytrace

raytrace_max_range: 센서가 장애물을 감지하는 최대 거리(m)

- 실내 환경에서↓반응속도↑
- 값이 크면 오탐 가능성(노이즈)

obstacle_max_range:센서가 장애물을 비용 맵에 반영하는 최대 거리(m)

- local: 너무 먼 장애물은 실제 비용맵에 반영x, 경로계획 최적화
- Global: 비효율적 우회 방지

LiDAR 데이터는 global&local timestamp 포함되어 topic

nav2.yaml

smoother_server

```
tolerance: 1.0e-10 -> 0.01
```

smoother_server.tolerance: Global 경로를 얼마나 부드럽게 조정할지 결정하는 허용 오차

- 기존 직선 형태: 경로 수정이 거의 불가(경로가 직선적이고 급격한 방향 전환이 많아질 가능성)
- 10cm로 증가시켜 경로 일관성&유연한 경로보정

	경로 조정 성능	장애물 회피	경로 부드러움
1.0e-10 (기존)	거의 불가	회피 능력 저하	직선적, 방향 틀때 부자연스러움
0.1 (새로운 값)	구간 수정 가능	회피 가능	곡선 주행경로
0.1 이상(높은 값)	경로 일관성 저하	회피성능↑	자연스러운 주행 경로

nav2.yaml

behavior_server

```
max_rotational_vel: 1.0 -> 2.0
```

max_rotational_vel: (rad/s)

- 목표 방향 조정, 장애물 회피 시 회전 반응 속도

알고리즘

map - 미탐색 : -1, 탐색 : 0, 벽 : 100

-1인 지점을 찾고
그 중에 0과 접한곳을 찾고
100과는 떨어진 곳 중
가장 가까운 곳으로 가자

mapping

문제

- goal이 맵 밖으로 나감
- 원하는 위치에 goal이 지정되지 않음
- 코너 부근에서 선회를 하지 못함

원인 추정

- 좌표 변환에 문제 발생
- 충돌 영역이 크게 설정

해결 방안

- resolution, position을 사용하여 좌표 변환
- costmap inflation radius를 조정

Pseudo code

```
노드 초기화:
        /map 구독자
        /goal_pose 발행자

        초기 위치 초기화
        목표점 초기화


맵을 구독:
        맵 데이터 가져오기(위치, 해상도)
        초기위치 설정
        맵 데이터 리스트화
        맵 리스트 패딩
        위치리스트 초기화
        맵에서-1 인 지점 탐색
        for -1 인 곳:
                if 근처에 0이 있고 100이 없으면:
                        거리를 계산해서 위치리스트에 추가

        if 위치리스트가 있으면:
                가장 가까운 곳으로 goal_pose 발행
        else:
                초기 위치로 복귀
                if 현재 위치가 초기 위치면:
                        복귀 완료
```

개선 사항

stuck

ㄴ내 위치가 일정 시간 변하지 않으면 탈출하는 알고리즘 추가

선회시 걸리는 경우

ㄴ더 부드러운 선회를 위해 추가적인 parameter tuning

cleaning

무엇을 해야할까?

두가지 방식으로 접근

1. 전체 맵을 사용해서 구역을 분할하고 경로를 미리 생성해서 그 경로를 따라가게 만들자
2. ㄹ자로 청소를 하는데 미청소인 영역도 찾아서 가자

방법 1

청소 알고리즘 개요

목표

맵핑 완료된 맵 호출
맵의 무결성 해소
맵을 맵의 특성에 따른 구역으로 분할
각 구역의 최적 경로를 계산

시간 간소화를 위해 시뮬레이션 이용

청소 알고리즘 개요

알고리즘 단계: 안전 영역 확보 및 전처리

- 장애물(값 100) 영역은 로봇이 실제로 이동할 때 충분한 안전 거리를 두기 위해 dilation(확대)을 적용합니다.
- 여기서 dilation은 로봇 반지름(약 0.171 m)과 안전 마진(0.1 m)을 합산한 약 0.271 m를 기준으로, 해상도에 따라 커널 크기를 산출합니다.
- 전처리 단계에서는 모폴로지 클로징(closing)과 추가로 erosion(수축)을 적용하여, 분산된 free 영역들을 병합하고 실제 로봇이 안전하게 이동할 수 있는 영역(안전 free 영역)을 확보합니다.

청소 알고리즘 개요

알고리즘 단계: 청소 영역 분할

- 전처리된 free 영역 이미지에서 connected components(연결 요소 분석)를 사용하여 청소 가능한 영역들을 추출합니다.
- 최소 영역 크기는 실제 1m×1m 크기에 해당하는 픽셀 수 (예: $(1/\text{resolution})^2$)를 기준으로 하였지만, 환경에 따라 이 값이 너무 높아서 영역이 전혀 나오지 않는다면 조정할 수 있습니다.

청소 알고리즘 개요

알고리즘 단계: 청소 경로 생성

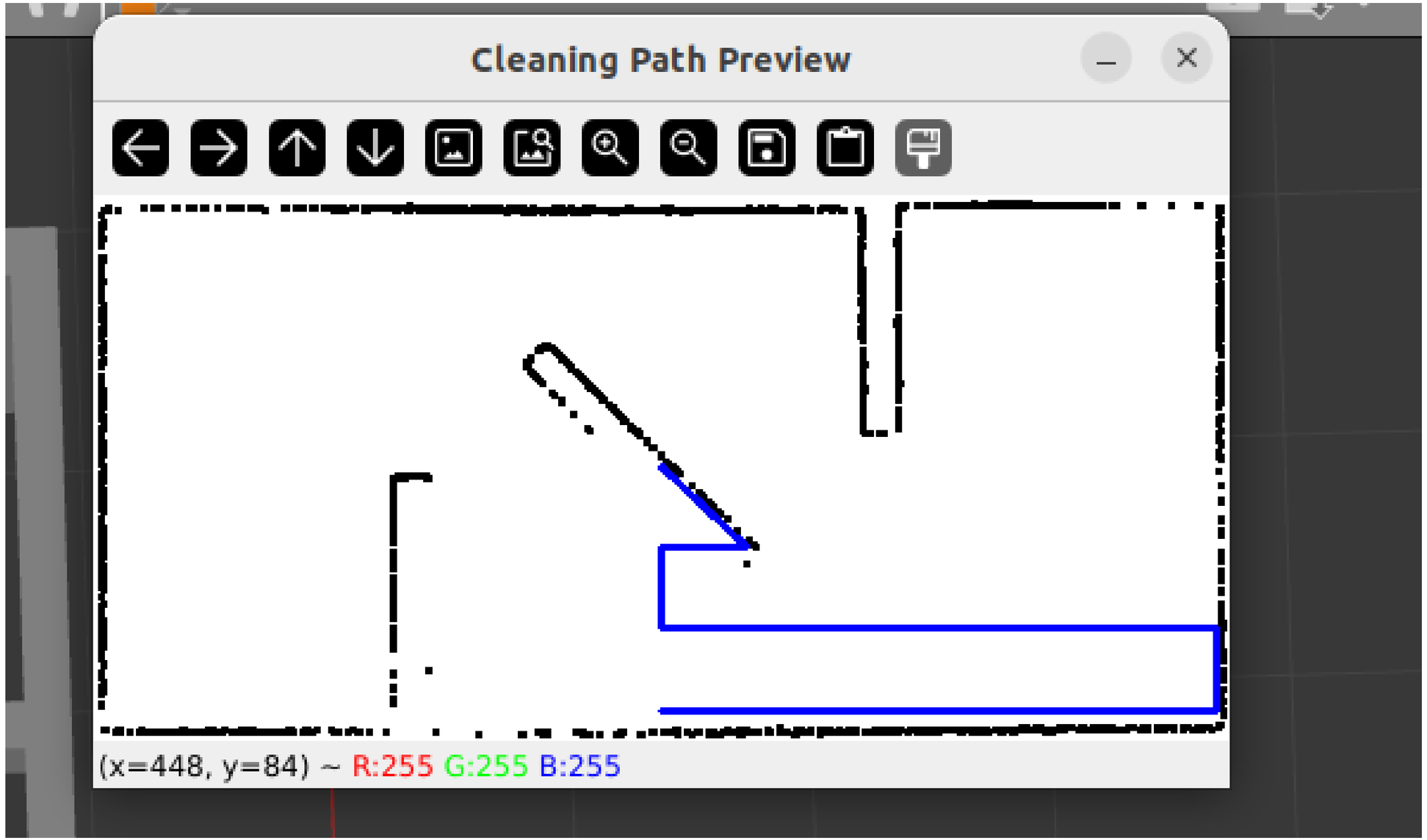
- 각 청소 영역 내에서 로봇의 크기(0.342m)를 고려한 spacing(간격)으로 스위핑(sweeping) 방식의 경로 후보를 생성합니다.
- 생성된 경로 후보에 대해 Bresenham 선 알고리즘을 사용하여, 연속적인 두 waypoint 사이에 장애물이 없는지 확인(즉, 실제 free 영역 내에 있는지 검사)합니다.
- 이후, Douglas-Peucker 알고리즘을 적용하여 경로를 단순화(직선화)함으로써, 울퉁불퉁한 경로를 보다 매끈한 경로로 보정합니다.

청소 알고리즘 개요

알고리즘 단계: 청소 실행 (경로 추종):

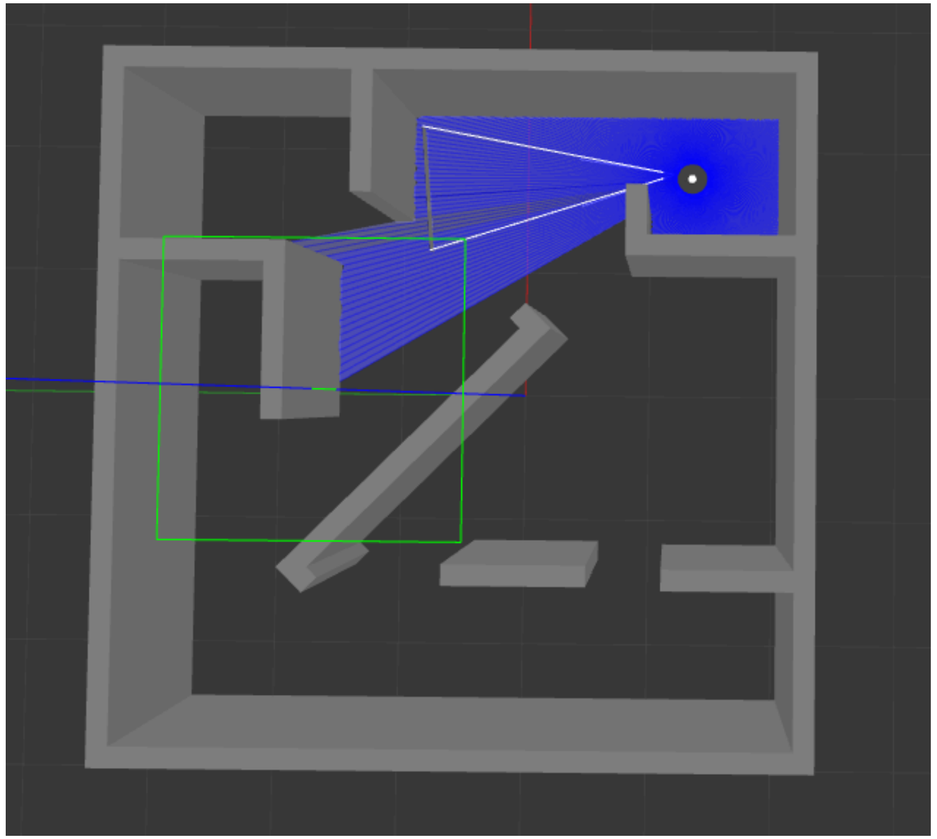
- nav2로 사용시param 값에 따라 경로가 계속 틀어지는 경우가 있었습니다.

- 청소 단계에서는 odometry를 통해 현재 로봇의 위치를 계속 업데이트하고,
- 청소 경로의 각 waypoint까지의 오차를 계산한 후, cmd_vel(Twist 메시지)를 발행함으로써 로봇이 경로를 따라 이동하게 합니다.
- 모든 영역에 대한 청소 경로를 순차적으로 완료하면, 로봇은 그 자리에서 0 Twist 메시지를 발행받아 정지합니다.


Cleaning Path Preview
(x=448, y=84) ~ R:255 G:255 B:255

시간 부족 문제

배터리 및 순서상 실제로 테스트 해볼 시간이 적음

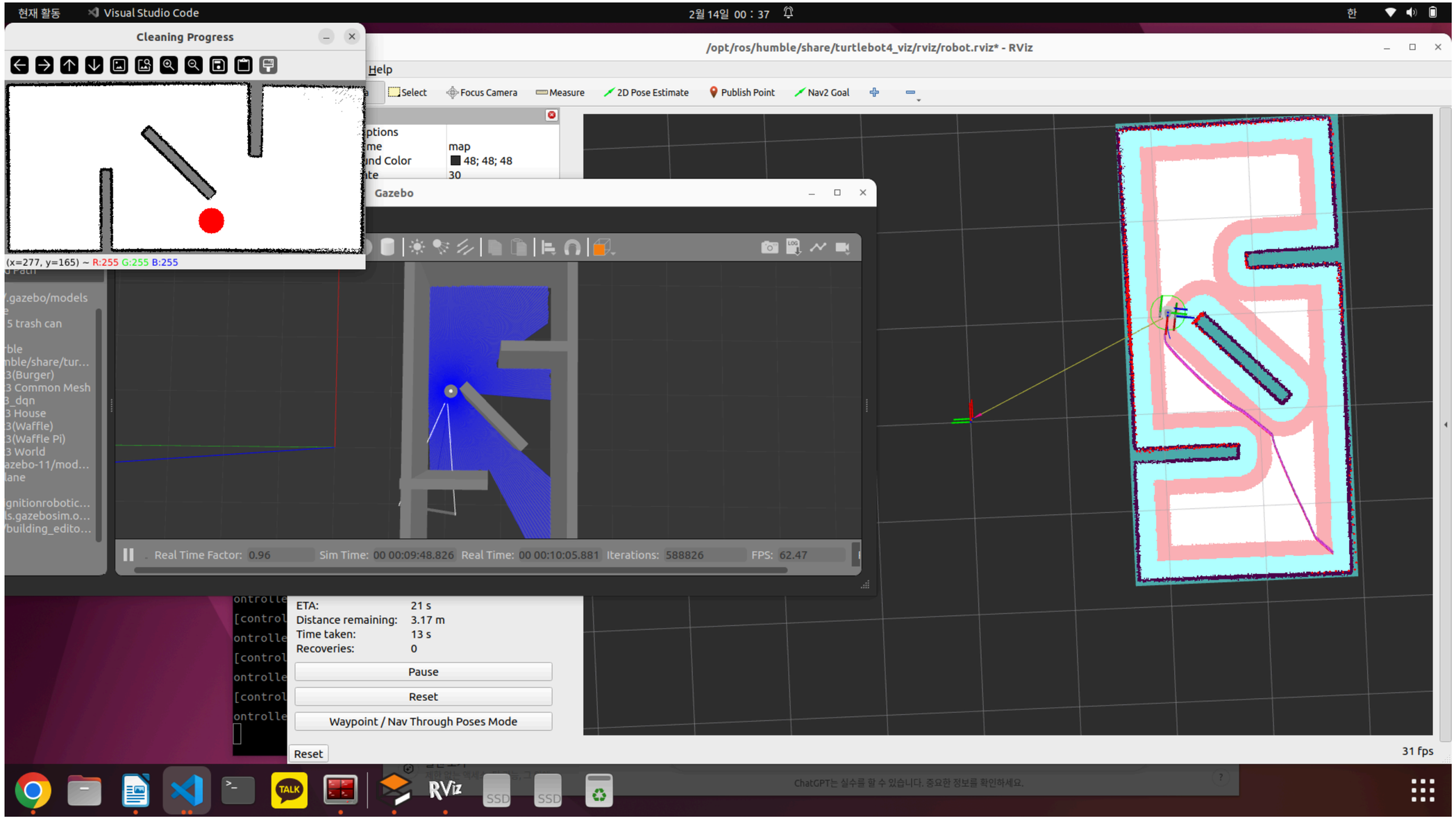


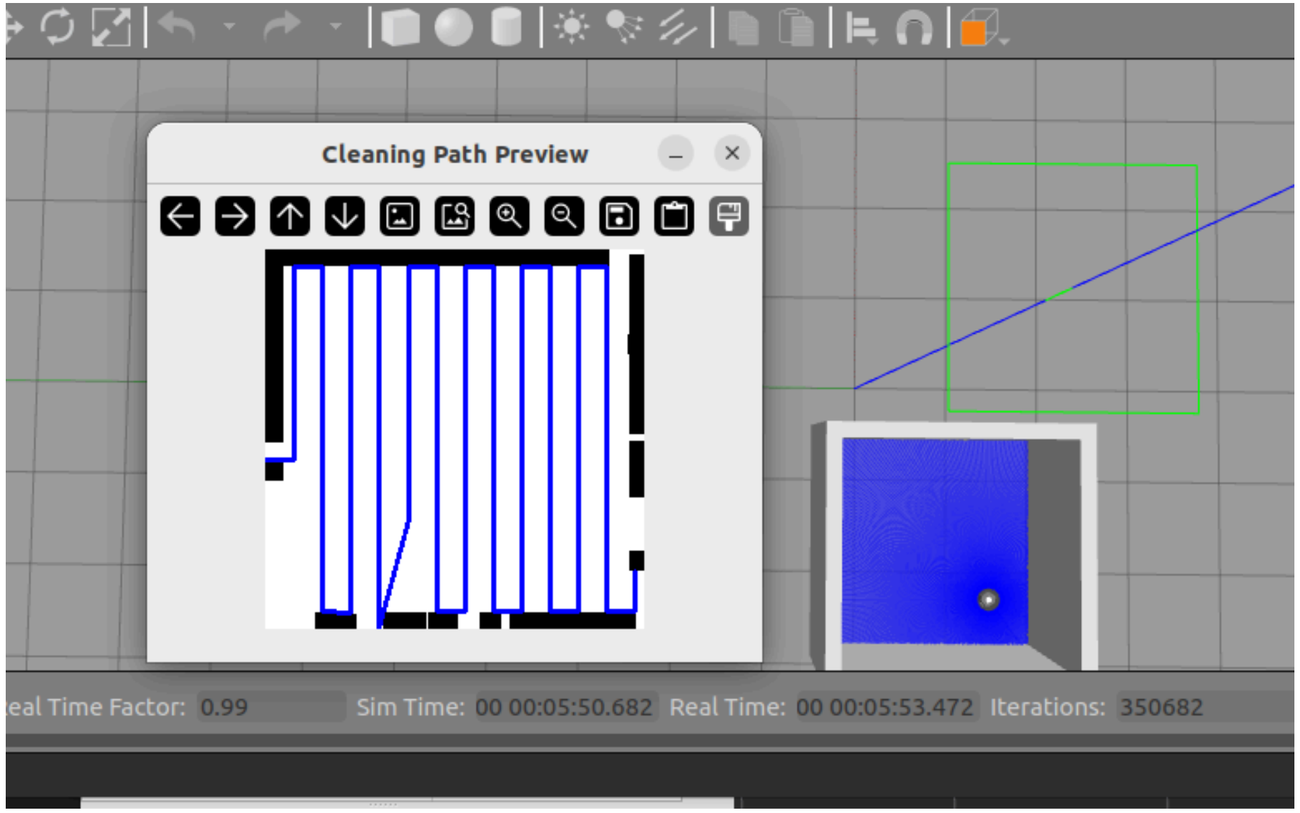
해결법:

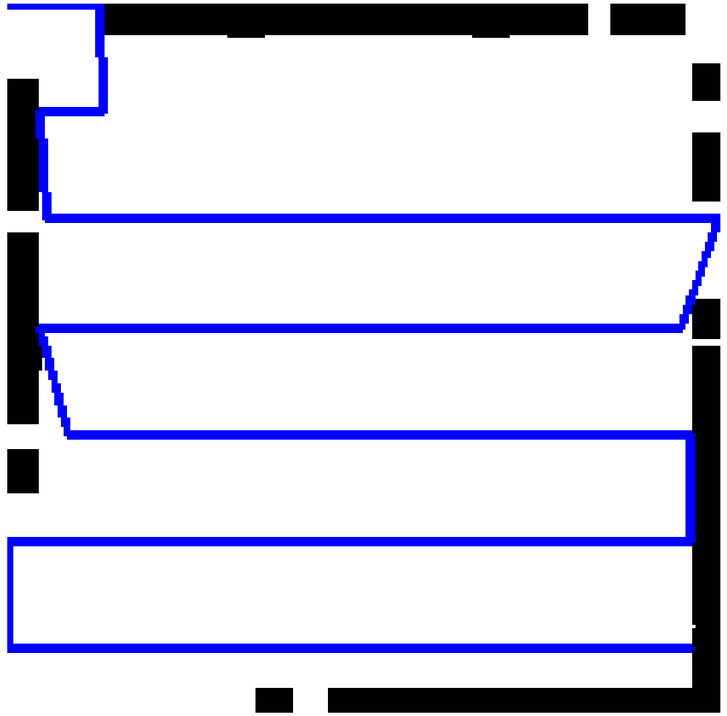
가제보 시뮬레이터에서 모델 직접 생성

아쉬운점:

마지막날 되서야 제작방법을 깨우침


현재 활동
Visual Studio Code
2월 14일 00 : 37
한
Cleaning Progress
(x=277, y=165) ~ R:255 G:255 B:255
/opt/ros/humble/share/turtlebot4_viz/rviz/robot.rviz* - RViz
Help
Select
Focus Camera
Measure
2D Pose Estimate
Publish Point
Nav2 Goal
map
48; 48; 48
30
Gazebo
Real Time Factor: 0.96
Sim Time: 00 00:09:48.826
Real Time: 00 00:10:05.881
Iterations: 588826
FPS: 62.47
ETA: 21 s
Distance remaining: 3.17 m
Time taken: 13 s
Recoveries: 0
Pause
Reset
Waypoint / Nav Through Poses Mode
Reset
31 fps
ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.


Cleaning Path Preview
eal Time Factor: 0.99
Sim Time: 00 00:05:50.682
Real Time: 00 00:05:53.472
Iterations: 350682



문제 발생 및 수정 과정

청소 경로 생성을 위한 매핑 정확도

- 각 청소 영역 내에서 로봇의 크기(0.342m)를 고려한 spacing(간격)으로 스위핑(sweeping) 방식의 경로 후보를 생성
- 생성된 경로 후보에 대해 Bresenham 선 알고리즘을 사용하여, 연속적인 두 waypoint 사이에 장애물이 없는지 확인(즉, 실제 free 영역 내에 있는지 검사)
- Douglas-Peucker 알고리즘을 적용하여 경로를 단순화(직선화)함으로써, 울퉁불퉁한 경로를 보다 매끈한 경로로 보정합니다.

문제 발생 및 수정 과정

청소 영역(Region) 수가 너무 적게 나옴

원인:

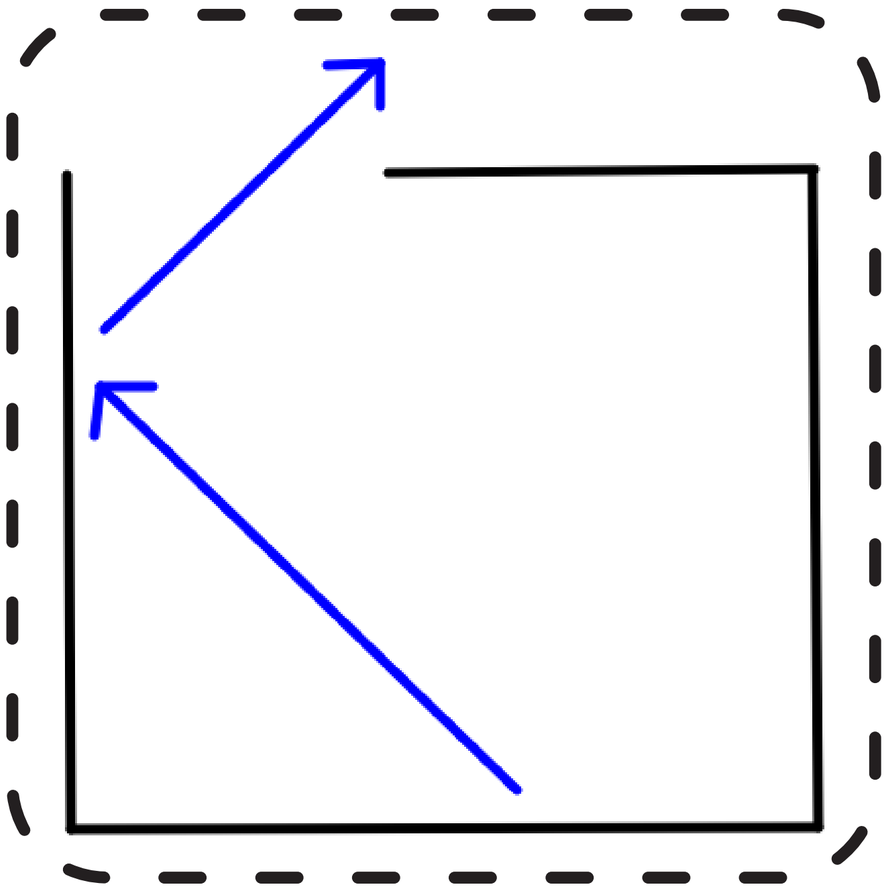
- 초기에는 dilation 커널 크기가 로봇 크기만 반영하거나 너무 크게 설정되어, 실제로 로봇이 이동할 수 있는 공간이 전부 장애물로 처리되는 경우가 있었습니다.

수정:

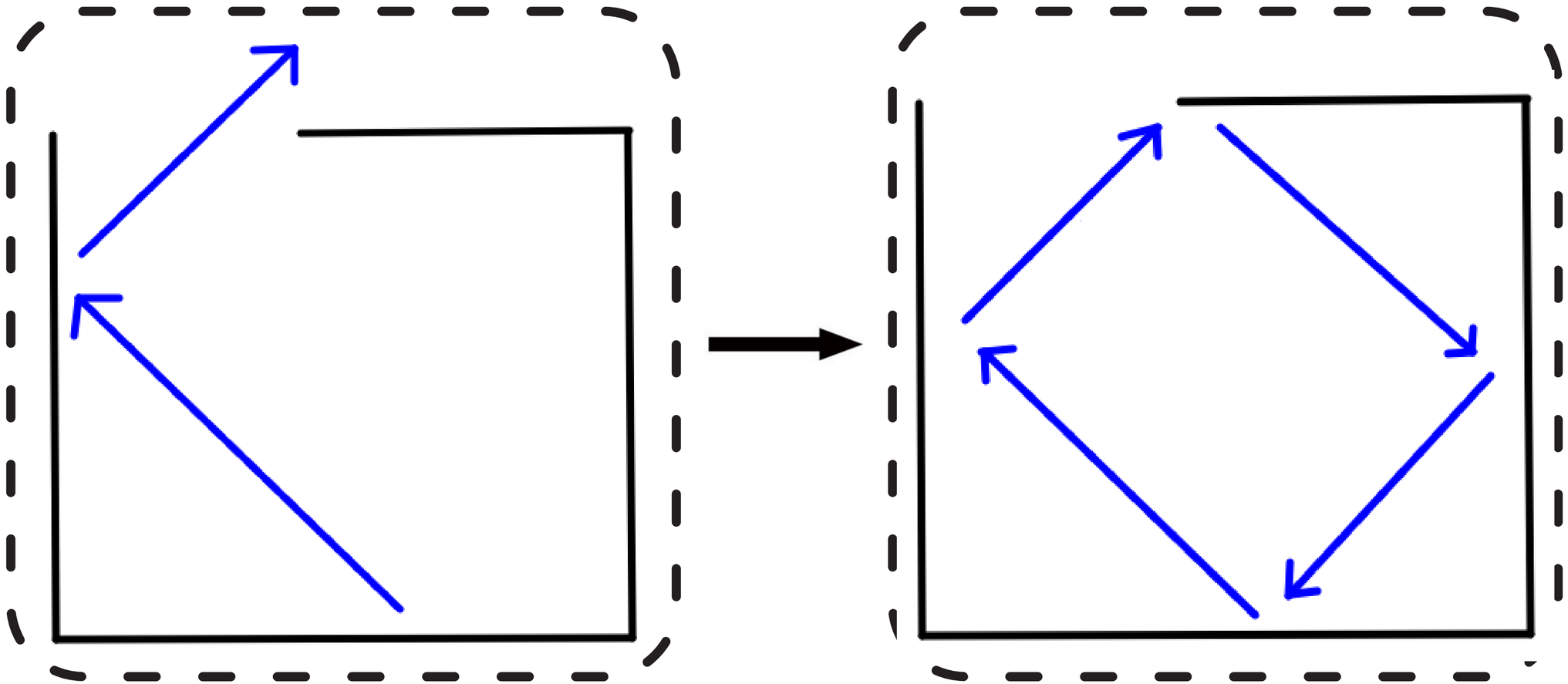
- 로봇 반지름(342 mm의 절반, 약 0.171 m)과 안전 마진(0.1 m)을 합산하여 약 0.271 m의 dilation 거리를 산출하고, 해상도에 따라 커널 크기를 계산하도록 하였습니다.
- 이를 통해 경로 후보가 벽에서 최소 10cm 이상의 안전 거리를 확보하면서 생성되도록 하였습니다.

방법 2

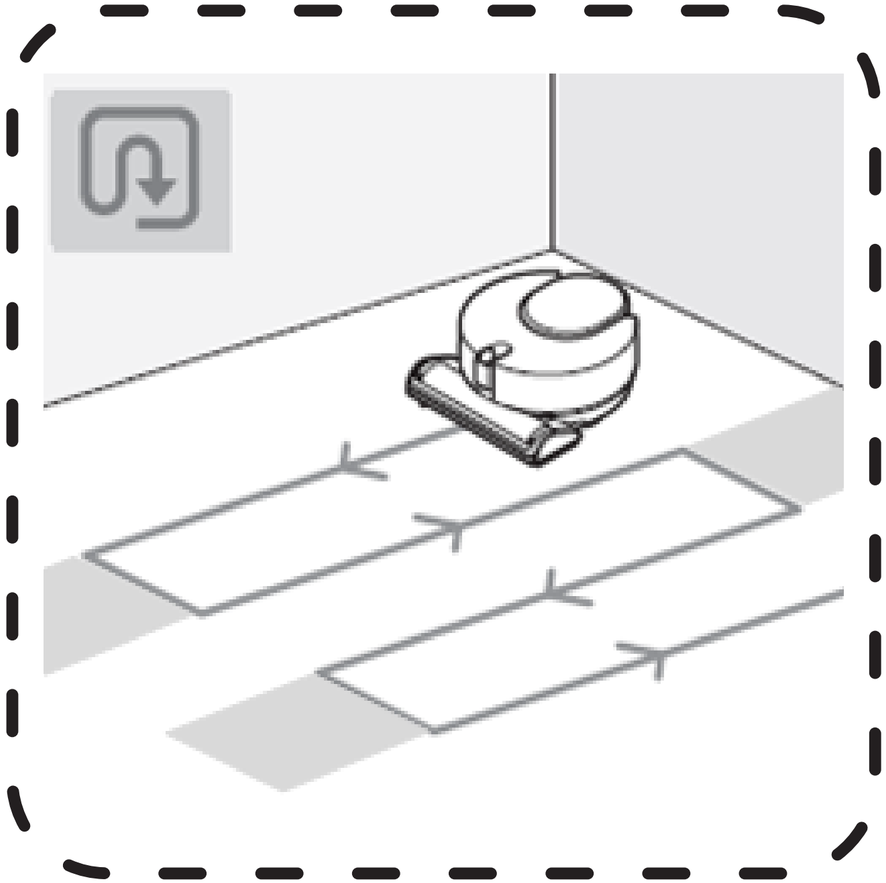
OPTION



OPTION



OPTION



OPTION - PSEUDO CODE

Function map_callback(msg):
- 수신한 맵 데이터를 map_data 변수에 저장

Function is_visited(x, y):
- 맵 데이터가 존재하지 않으면 False를 반환
- (x, y) 좌표를 기반으로 맵의 인덱스를 계산
- 해당 인덱스가 방문한 위치 목록에 포함되어 있으면 True를 반환하고, 그렇지 않으면 False를 반환

Function mark_visited(x, y):
- 맵 데이터가 존재하는 경우:
  - (x, y) 좌표를 기반으로 맵의 인덱스를 계산
  - 해당 인덱스를 방문한 위치 목록에 추가

Function lidar_callback(msg):
- Lidar 데이터를 8개의 섹터로 나눔
- "front-right" 섹터의 거리 정보를 추출
- 측정된 거리를 기반으로 front_right_distance 값을 설정
- 거리가 임계값(threshold) 이하이면 obstacle_detected 변수를 True로 설정

Function move_robot(linear_x, angular_z, duration):
- 로봇의 이동을 위한 Twist 메시지를 생성
- 선속도(linear)와 각속도(angular)를 설정
- 이동 명령을 퍼블리시
- 명령 실행 후 로봇을 정지
- 맵 데이터가 존재하는 경우:
  - 현재 (x, y) 좌표를 맵 데이터에서 가져옴
  - 해당 위치를 방문한 위치 목록에 기록

Function control_loop():
- 만약 장애물이 감지되었다면:
  -장애물 감지를 로그로 출력
  - 로봇을 정지
  - turn_count 값을 기반으로 회전 방향을 결정
  - turn_count 값을 증가
  - 각속도 및 회전 지속 시간을 계산
  - 결정된 방향으로 45도 회전을 두 번 수행
  - 로봇을 6cm 전진
  - 다시 같은 방향으로 45도 회전을 두 번 수행
- 장애물이 없을 경우:
  - 현재 위치가 방문되지 않았다면:
    - 로봇을 전진
  - 이미 방문한 위치라면:
    - "갈 곳이 없습니다."라는 메시지를 출력

OPTION - Video

OPTION - 한계점

직진 시
이동 거리

아래로
이동 불가

회전 각도 조절

cleaning

알고리즘

mapping과 유사하게 진행

1. 벽 확장
2. 청소된 경로 -> 1

   청소안된 경로 -> 0
3. 가까운곳을 목표점으로

발생한 문제

1. 너무 짧은 이동거리
2. 벽 근처에서 멈춤

해결방안

1. 최소 이동거리 도입
2. 1로 변환하는 mask 크기 증가

cleaning

Pseudo code

Code 1

```
노드 초기화:
        /map 구독
        /goal_pose 발행

        초기 위치 초기화
        목표점 초기화
        방문점 초기화
```

Code 2

```
맵을 구독:
        맵 데이터 가져오기(위치, 해상도)
        맵 데이터 리스트화
        맵 리스트 패딩
        위치리스트 초기화

        맵에서 100인 지점 탐색
        for 100인곳:
                근처를 100으로 키움
        for 방문점:
                맵에서 1로 채움

        맵에서 0 인 지점 탐색

        for 0인곳:
                if 최소거리 이상:
                        위치리스트에 추가

        if 위치리스트가 있으면:
                가장 가까운 곳으로 goal_pose 발행
                현재 위치를 방문점에 추가
        else:
                청소 완료
```

Thank you