SLAM 모델 기반 로봇 주행
환경 장애물 인식 모델 개발

B1: 이수형, 이건원, 이대원

목차

파라미터 수정 전략

```
required_movement_radius: 0.0
```

기존 0.3에서 0.0으로 변경

```
resolution: 0.025
```

맵 해상도를 높이기 위해 초기 0.06에서 0.025로 변경

```
local_costmap:
inflation_radius: 0.15
```

```
global_costmap:
inflation_radius: 0.3
```

안전하고 부드러운 경로 생성을 위해 0.3, 실시간 장애물 회피를 더 유연하게 처리를 위해 0.15로 설정

Pseudo code

Algorithm 1 맵 탐색 알고리즘

```
1: procedure 맵 데이터 수신(msg)
2:     수신된 맵을 저장
3:     if 동작중이 아니고 목적지 도착 상태 then
4:         plan_nearest_boundary_goal() 실행
5:     end if
6: end procedure
7: procedure 명령 데이터 수신(msg)
8:     if 완전히 정지 then
9:         is_moving ← False
10:        목표 지점 도달로 간주하고 current_x, current_y 업데이트
11:    end if
12: end procedure
13: procedure 목적지 명령 결과 수신(future)
14:    목표 도착 상태를 확인
15:    목적지 도착 상태로 설정
16: end procedure
17: procedure MAIN ALGORITHM
18:    맵 데이터를 2D 격자로 변환
19:    맵의 미확인 영역과 확인 영역의 경계 찾기:
20:    for 격자의 모든 셀 (i, j) (테두리는 제외) do
21:        if 미확인 영역 then
22:            상하좌우에 값이 0 (장애물 없음)인 셀이 있는지 확인
23:            if 있다면 then
24:                (i, j)를 boundary_cells에 추가
25:            end if
26:        end if
27:    end for
28:    if boundary_cells이 비어 있으면 then
29:        알고리즘 종료
30:    end if
31:    가장 가까운 안전한 경계 셀 찾기:
32:    min_distance ← 무한대
33:    for boundary_cells의 모든 셀 (i, j) do
34:        (i, j)를 실제 좌표로 변환
35:        현재 좌표와 거리 계산
36:        if 최소 거리 이상 떨어져있고 셀이 장애물로부터 충분히 거리가 있다면
   then
37:            이동 목표로 설정
38:        end if
39:    end for
40:    if 유효한 목표가 없으면 then
41:        "이동할 안전한 목표 없음. 종료." 메시지 출력 후 노드 종료
42:    end if
43:    네비게이션으로 목적지로 이동 명령 실행
44: end procedure
```

Pseudo code

Algorithm 1 이미지 처리 및 마커 발행 알고리즘

procedure 노드 초기화
    QoS 프로파일 설정: `reliability = BEST_EFFORT`, `history = KEEP_LAST`, `depth = 10`
    압축된 이미지 토픽 `/oakd/rgb/preview/image_raw/compressed` 구독
    포즈 토픽 `/pose` 구독
    CvBridge 초기화
    참조 이미지 읽기 및 회색조로 변환
    SIFT 디텍터 생성 및 참조 이미지 디스크립터 계산
    BFMatcher를 L2 노름으로 초기화
    카메라 내적 행렬 `K`와 왜곡 계수 `dist_coeffs` 정의
    시각화 마커 퍼블리셔 생성
    변환 행렬 `T_maptobaselink`, `T_baselinktocam` 정의
    노드 시작 로그 출력
end procedure
procedure 회전 행렬로 변환(쿼터니언)
    쿼터니언 정규화
    쿼터니언을 사용하여 회전 행렬 계산
    회전 행렬 반환
end procedure
procedure 변환 행렬 생성(회전 벡터, 이동 벡터)
    Rodrigues 공식을 사용하여 회전 벡터를 회전 행렬로 변환
    4x4 변환 행렬 구성
    변환 행렬 반환
end procedure
procedure MAP 좌표계에서 BASELINK 좌표계로 변환
    `T_maptobaselink` 변환 행렬을 사용하여 map 좌표를 baselink 좌표계로 변환
end procedure
procedure BASELINK 좌표계에서 카메라 좌표계로 변환
    `T_baselinktocam` 변환 행렬을 사용하여 baselink 좌표를 카메라 좌표계로 변환
end procedure
procedure 포즈 콜백 함수(msg)
    수신된 포즈를 `posemap`에 저장
end procedure
procedure 포즈로부터 변환 행렬 생성(msg)
    포즈에서 위치 및 쿼터니언 추출
    쿼터니언을 회전 행렬로 변환
    변환 행렬 생성 후 반환
end procedure
procedure 마커 발행
    `baselink` 좌표계에서 카메라 좌표계로 변환한 후,
    `map` 좌표계에서 `baselink` 좌표계로 변환하여 이미지 변환 행렬 계산
    회전 행렬을 쿼터니언으로 변환
    위치, 방향, 크기로 마커 생성
    `/visualization_marker`에 마커 발행
end procedure
procedure 회전 행렬을 쿼터니언으로 변환(회전 행렬)
    회전 행렬의 트레이스를 사용하여 쿼터니언 계산
    계산된 쿼터니언 반환
end procedure
procedure 이미지 콜백 함수(msg)
    CvBridge로 이미지를 OpenCV 형식으로 변환
    이미지를 회색조로 변환
    SIFT를 사용해 키포인트와 디스크립터 계산
    참조 이미지와 매칭하여 3D 좌표 추정
    PnP를 사용하여 포즈 계산
    포즈 추정 성공 시 마커 발행
end procedure

소감

.이미지를 탐색하여 로봇에게 인식시켜주는 부분이 어려웠다.

.터틀봇이 지나가지 못하는 부분이 많아 올바른 값을 찾아 수월하게 움직일 수 있도록 하는것에 어려움이 있었다.

“로봇 시연”