트랙 주행을 위한 경로 생성

현재 상황
맵리스

Mapless Navigation using Onboard Sensors
특징 : 특징 : HD Map x, GPS x, 절대 좌표계 x, **차량 기준 상대 좌표계 (local frame) 만 사용가능** → 연속적인 주행 경로 (trajectory) 만 생성

개요
센서 관측 → 즉시 판단 → 제어

사용하는 센서 : 카메라 , 라이더 , 초음파
센서 입력 → 센서 융합 (Perception) → 자차 기준 환경 표현 (Local Map) → 주행 가능 영역 추정 (Free Space) → 로컬 경로 생성 (Local Planning)

트랙 주행을 위한 경로 생성

Step 1

센서입력

카메라 : 주행 가능 영역에 대한 시각정보 -> 차선 곡선 , 주행 가능 영역 마스크

라이더 : 트랙 경계에 대한 정보 -> x,y 의 좌표정보

초음파 : 라이더 사각지대 보완 -> 근접한 거리에 대한 정보

Step 2

센서 융합

상대 좌표계 (local frame)
차량중심 (0,0), 전방 : +x, 좌측 : +y
이렇게 차를 기준으로 상대적인 위치값만을 이용하여 나타냄

Step 3

자차 기준 환경 표현

지도 없이 센서로 본 길의 모양을 차 기준 좌표계에서 정리

Step 4

주행 가능 영역 추정

차 기준 좌표계를 통해
/ 현재 차량 폭 + 여유 > 트랙 폭
→ 이 구간은 주행 불가

Step 5

로컬 경로 생성

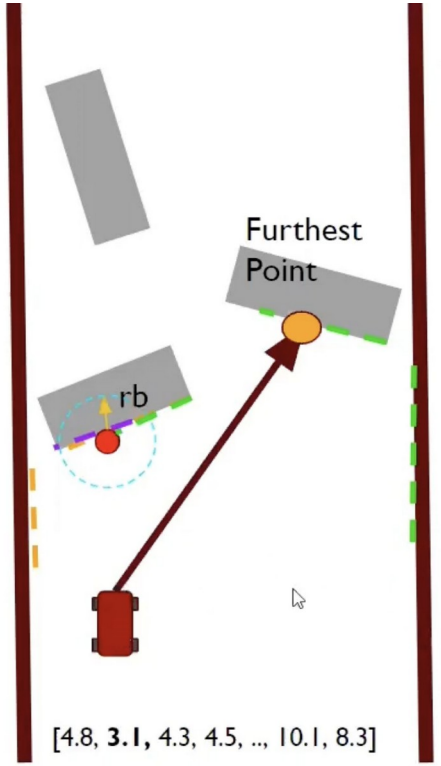
자차 기준 환경 표현과 주행 가능 영역을 바탕으로 차량이 지금부터 짧은 거리 동안 실제로 따라갈 연속적인 곡선 제작

장애물 회피 - Follow the Gap

Step 1
Find nearest LIDAR point and put a “safety bubble” around it of radius *rb*

Step 2
Set all points inside bubble to distance 0. All nonzero points are considered ‘free space’

Step 3
Find maximum length sequence of consecutive non-zeros among the ‘free space’ points - The **max-gap**



Step 4
Find the ‘best’ point among this maximum length sequence

Naive: Choose the furthest point in free space, and set your steering angle towards it

Changing speed results in you losing velocity

Better Idea Intuition
If you’re 3-4m away from your closest obstacle, should you immediately make a sharp turn to avoid it?

1. 전방 스캔만 사용

2. 가장 가까운 장애물 (closest point) 탐색

3. 스캔에서 최소 거리 지점이 “가장 위험한 장애물”이라고 간주

4. Safety Bubble(버블) 생성
- 가장 가까운 장애물 주변을 안전거리 반경 r 만큼 통째로 “금지 영역”으로 마스킹
- 구현 : 버블 구간의 range 값을 0 으로 설정

5. Max Gap(찾기 갈 수 있는 공간)
- 연속으로 가장 길게 이어진 구간 (Gap)을 선택

6. Gap 내부 목표점 선택 - gap 중앙 또는 gap 안에서 가장 먼 점 (farthest point) 선택

7. 간단한 조향 - 정면 인덱스 기준으로 좌 / 우 판별 하여 조향

장애물 회피 - Follow the Gap

$$\delta_{cmd} = \alpha\delta_{lane} + (1-\alpha)\delta_{gap}$$

제어 입력 예시

. 라이다에서 인식되는 장애물의 거리에 따라 를 서서히 감소 (장애물에 가까울 수록 0 에 가깝게 보냄).

. 평소 장애물이 인식 안됨 -> = 1 에 가깝고 기존의 조향각 입력 () 을 주로 사용

장애물 정지 - 라이다 , 초음파 사용

라이다 , 초음파를 사용하여 특정 거리 (매우 근접한 거리) 내에 감지가 된다면 STOP 하도록 제어 명령

회피도 해야할 수 있으므로 충돌 임박 상황에서만 멈추도록

주차

현재 상황

GPS 가 없어 차량의 절대 위치를 알 수 없음
차량이 이동할수록 센서 기준이 계속 바뀜

전체 시스템 흐름

1. 주차장 탐색 및 주차 공간 설정
2. 목적지 추적
3. 지역 경로 설정

주차장 탐색 및 주차 공간 설정

빈 공간 탐색 : 라이다 (장애물 인식) 를 사용해 비어 있는 공간을 찾는다 .

공간 검증 : 탐색한 공간이 내 차의 크기보다 좌우로 약 40~60cm 이상의 여유가 있는지 확인 .

도착점 (Goal) 설정 : 카메라로 인식한 주차선 4 개의 모서리 점을 찍어 가상의 직사각형을 만듭니다 .
중심선 추출 : 직사각형의 한가운데를 가로지르는 중심선을 긋습니다 .

최종 위치 (x,y): 주차 칸 안쪽 벽면에서 내 차 길이의 절반 (1/2) 만큼 떨어진 중심선 위의 지점을 도착점으로 정합니다 .

최종 방향 (θ): 주차 칸의 좌우 선과 내 차가 완벽히 평행이 되는 각도를 목표 방향으로 정합니다 .

목적지 추적

기준점 고정 : 주차 공간을 처음 발견한 순간의 내 차 위치를 (0, 0, 0) 으로 설정 .

상대 좌표 업데이트 (ICP): 차가 움직일 때마다 라이다가 주변 환경을 스캔하여 이전보다 얼마나 (x, y) 어느 방향 (θ) 으로 움직였는지를 0.01 초 단위로 계산 .

ICP: 라이다를 통해 얻은 두 개의 Point Cloud 데이터를 가장 잘 맞도록 겹치는 과정을 통해 " 그사이의 이동량 (거리와 회전)" 을 계산해내는 기술

실시간 추적 : 내가 움직인 만큼을 역산하여 , 아까 정해둔 도착점 (Goal) 이 현재 내 차를 기준으로 정확히 몇 미터 뒤 , 몇 도 방향에 있는지 계속 업데이트한다 .

(지금 이 순간 내가 가야 할 남은 거리와 방향)

지역 경로 생성 (Path Planning)

- 자동차가 갈 수 있는 ' 최선의 길 ' 찾기 (Reeds-Shepp 알고리즘 기반 노드)

- 핸들 각도 계산 : 내 차가 한 번에 꺾을 수 있는 최대 각도를 고려해서 , 현재 위치에서 끝점까지 연결되는 S 자 또는 원호 모양의 곡선을 그립니다 .
-
- 전진 / 후진 결정 : 한 번에 못 들어가는 좁은 공간이라면 , " 어디까지 전진했다가 핸들을 꺾어 후진할지 " 기어 변속 지점을 미리 정합니다
-
- 장애물과 부딪히는지 확인 (Safety Check)
- 라이다 데이터 대조 : 위에서 그린 ' 가상의 길 ' 위로 내 차가 지나갈 때 , 라이다로 찍어둔 옆 차나 벽에 닿는지 시뮬레이션해 봅니다 .
- 길 수정 : 만약 닿을 것 같다면 , 조금 더 크게 돌거나 전진을 더 많이 하는 식으로 안전한 길로 다시 그립니다 .