

제5회 국민대 자율주행 경진대회 예선경기 #과제3

자율주행 SW 설계서

1. 참가팀 일반사항

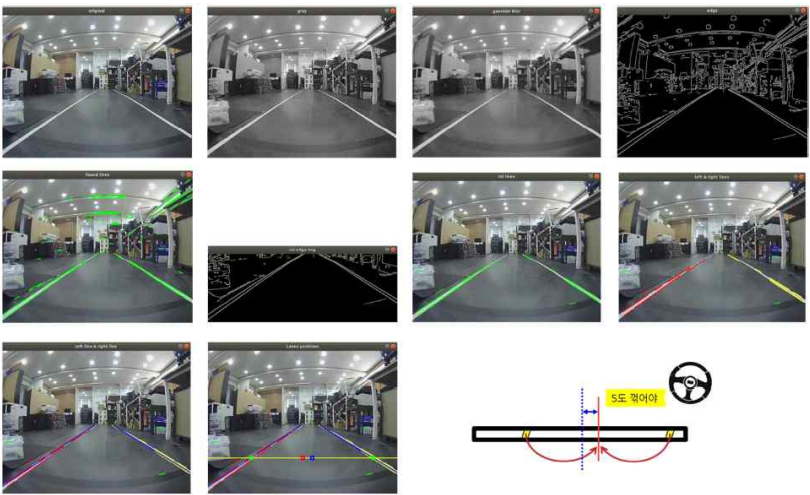
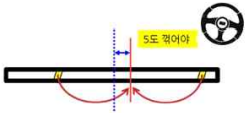
참가팀	팀명	천마력
	팀장	박윤아(영남대)
	팀원	김태환(영남대), 정원호(영남대), 양재석(영남대), 손동민(영남대)

2. 미션 완수를 위한 자율주행 SW 설계서

111111

수행 미션 (1번)	차선을 벗어나지 않고 주행 <ul style="list-style-type: none">주행트랙에서 차량이 차선을 벗어나지 않고 주행해야 함. 코너와 곡선구간 차선 위에 세워둔 차선이탈 판정용 블록을 쓰러뜨리지 않아야 함.	
① 사용하는 센서별 용도	카메라	영상처리를 통해 차도에서 양쪽 차선의 위치를 파악하기 위해 사용
	초음파센서	주행 중에 전방에 있는 장애물과 충돌하는 것을 막기 위해 사용
	라이다	SLAM 기술로 차량이 트랙의 어디쯤 왔는지 위치를 파악하기 위해 사용
	IMU	차량의 현재 속도를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적인 구현방안	(1)	트랙에 그려진 차선을 인식하고 차량이 차선의 중간쯤에서 달려야 함.
	<ul style="list-style-type: none">- 카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.- 칼라 이미지를 회색톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.- 화면의 아래부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.- 흰색점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.- 왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.- 화면의 중심을 기준으로 왼쪽차선과 오른쪽 차선이 동일한 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.- 화면의 중심 기준선에서 왼쪽차선이 오른쪽 차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.	

	<ul style="list-style-type: none"> - 화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다. - 트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.
(2)	직선 구간에서는 빨리, 곡선 구간에서는 천천히 달려야 함.
	<ul style="list-style-type: none"> - 핸들의 조향각을 살펴서 좌우 꺾임 각도가 5도 이하이면 직선이라고 간주하고 속도를 크게 높인다. - 꺾임 각도가 5도 이상이면 곡선 구간이라고 간주하고 속도를 늦춘다. - S자 곡선처럼 중간에 회전방향의 좌우가 바뀌면서 직선구간이 잠깐 등장하는 경우가 있으므로 약간의 시간을 두고 여러 번 관찰하는 방식으로, 핸들의 꺾임 정도를 정확하게 파악하여 반응하도록 한다. - 갑작스러운 가속보다는 속도를 점진적으로 올리거나 내리는 방식으로 차량의 속도를 제어한다. - 모터 배터리의 잔량에 따라 모터의 회전속도가 하드웨어적으로 변화하는 경우가 있으므로 IMU센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 체크하여 속도제어에 반영한다.
(3)	핸들을 너무 자주, 그리고 급격하게 꺾으면 안됨. (안정적으로 주행하게끔)
	<ul style="list-style-type: none"> - 속도가 빠른 고속도로 주행에서는 핸들을 조금만 꺾어도 차량이 휘청이게 되며, 잘 못하면 차체가 중심을 잃어 사고가 날 수도 있다. - 차량의 속도가 빠를 경우에는 조금씩 핸들을 좌우로 계속 왔다갔다 꺾는 방식의 운전은 피한다. 특정 값 이하의 작은 핸들조작은 Skip 하고, 그 이상의 핸들조작만 반영하는 방식으로 핸들을 조작한다. - 차량의 속도가 빠른 경우에 급격한 핸들 꺾임은 차량을 휘청이게 하므로 피한다. 기존 핸들 각도와 새로운 핸들 각도의 차이가 큰지 체크하여 임계값 이상으로 크면 (또한 차량의 속도가 빠르면) 적당한 가중치 값을 곱하여 (예를 들면 0.5) 핸들조작 각도를 일부러 작게 만든다. - 이런 상황이 일정 횟수 이상 반복되면 차량의 속도를 빨리 줄인다. 차량의 속도가 줄면 가중치 값을 곱할 필요가 없으므로 핸들 조작량이 정상으로 되돌아 와서 차량이 차선을 벗어나지 않도록 신속한 핸들링이 가능해질 것이다.
(4)	차선 인식에 실패하면 원래 가던 방향으로 계속 주행해야 함.
	<ul style="list-style-type: none"> - 여러 가지 이유로 차선의 인식에 실패하면 일단은 차량이 원래 가던 방향으로 그대로 갈 수 있도록 앞서의 핸들 조향각과 속도를 그대로 사용한다. 이를 위해 핸들 조향각과 속도 값은 특정 변수에 저장하고 있어야 한다. - 하지만 정해진 시간이 지난 후에도 계속 차선인식을 하지 못하는 경우에는, 아마도 차선을 이탈했을 가능성이 높으므로, 이때에는 핸들을 정면으로 하고 속도를 0 값으로 하여 차량을 정차시킨다.
(5)	코너에 진입하기 전에 속도를 늦추고 코너에 진입한 후엔 속도를 높여야 함.
	<ul style="list-style-type: none"> - 고속 주행시에는 코너에 진입하기 전에 속도를 늦추는 것이 타당하다. 그리고 일단 코너에 진입하여 핸들이 꺾인 상태가 되면 속도를 다시 높여도 된다.

	<ul style="list-style-type: none"> - 카메라 영상에 윗부분을 살피서, 즉 조금 먼 곳에 있는 차선을 살피서 코너구간으로 진입하게 될 것인지를 파악하고, 이를 통해 코너에 진입하기 전에 차량의 속도를 늦췄다가 핸들을 꺾은 후 곧바로 다시 속도를 높인다. - 이 운전법은 일정 속도 이상의 고속에서만 의미가 있으므로 차량의 현재 속도를 파악하여 적절히 적용한다. <div style="background-color: yellow; padding: 5px;">(6) 장애물과 충돌하면 안됨.</div> <ul style="list-style-type: none"> - 전방에 있는 장애물과의 충돌은 어떤 경우에도 피해야 하므로 초음파센서를 사용하여 전방 30센치 안에 장애물이 있는지 항상 체크한다. - 장애물 체크 시점은, 영상처리 작업을 통해 핸들 조향각과 차량의 속도를 결정한 후 모터제어 토픽을 발행하기 직전에 수행하면 된다. - 전방 장애물이 감지되는 경우에는 모터제어 토픽에서 속도 값을 0으로 세팅하는 방법으로 장애물 충돌을 방지함. <div style="background-color: yellow; padding: 5px;">(7)</div> <div style="background-color: yellow; padding: 5px;">-</div> <div style="background-color: yellow; padding: 5px;">(8)</div> <div style="background-color: yellow; padding: 5px;">-</div> <div style="background-color: yellow; padding: 5px;">(9)</div> <div style="background-color: yellow; padding: 5px;">-</div> <div style="background-color: yellow; padding: 5px;">(10)</div> <div style="background-color: yellow; padding: 5px;">-</div>
<p>③ 기타 구현 기술과 관련된 내용 자유롭게 기술</p> <p>(구현 기술과 관련된 그림 포함)</p>	<ul style="list-style-type: none"> • OpenCV 영상처리 기반 허프변환 기법을 기반 차선인식 (참조사이트 https://machinelearningknowledge.ai/lane-detection-tutorial-in-opencv-python-using-hough-transform/) <div style="display: flex; flex-wrap: wrap;">  </div> <ul style="list-style-type: none"> • 

- OpenCV 영상처리 기반 원근변환과 슬라이딩윈도우 기법을 이용한 차선인식

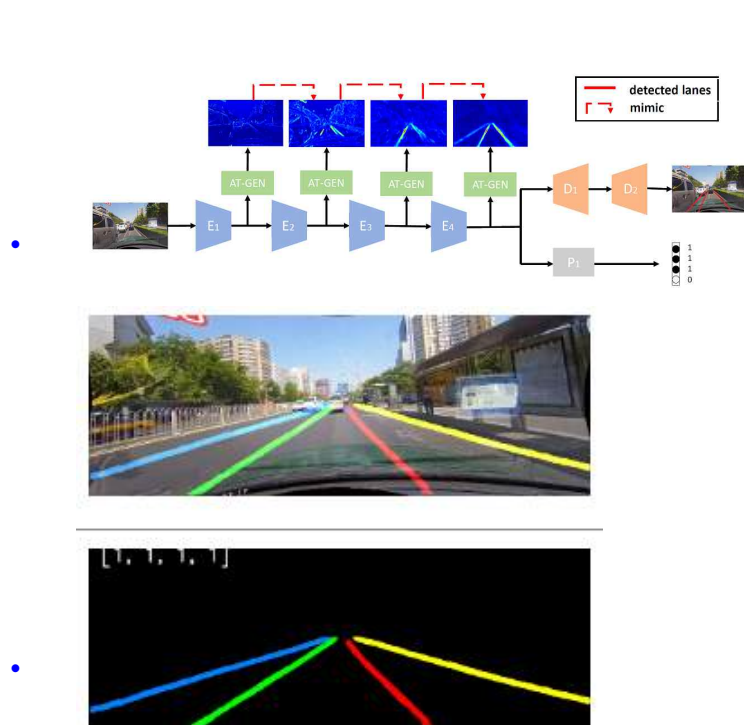
(참조사이트 <https://medium.com/geekculture/advanced-techniques-for-lane-finding-self-driving-cars-fc1a147fc49>)

a)



- 머신러닝 기반의 객체인식 모델을 이용한 차선인식

(참조사이트 https://github.com/InhwanBae/ENet-SAD_Pytorch)

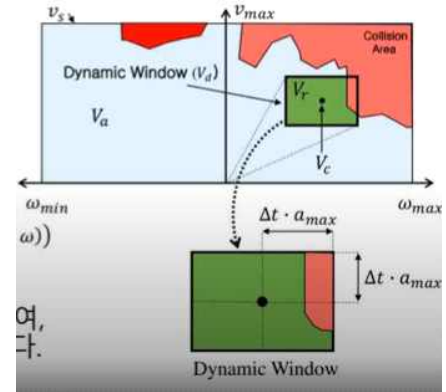


수행 미션 (2번)	어두운 터널을 통과 <ul style="list-style-type: none"> 카메라 영상으로 차선을 인식하는 것이 불가능한 어두운 터널 안에서 거리센서(초음파센서 또는 라이다)를 이용해서 벽과 충돌하지 않으면서 주행하여 터널을 빠져나와야 함. 	
① 사용하는 센서별 용도	카메라	영상처리를 통해 차도에서 양쪽 차선의 위치를 파악하기 위해 사용
	초음파센서	주행 중에 전방에 있는 장애물과 충돌하는 것을 막기 위해 사용
	라이다	SLAM 기술로 차량이 터널 벽과 충돌 하지 않게 하기 위해 사용
	IMU	현재 자세에서 지도의 지정된 목표 자세로 이동하기 위해 사용
② 구현할 기능 요약, 그리고 구체적인 구현방안	(1)	카메라 영상으로 차선을 인식하는 것이 불가능함
		<ul style="list-style-type: none"> 터널에서는 라이다를 사용하여 차량 주변의 장애물을 파악하여 피하고 방향 설정을 통하여 출구로 나가게끔 알고리즘을 구성했다. 라이다에서 값을 받아와서 원하는 값으로 가공하는 노드 주로 필요한 정면의 거리값을 배열로 받아서 정면, 왼쪽, 오른쪽으로 나누어서 라벨링한다. 전의 값과의 차이가 임계치를 넘어가는 경우 이전과 같은 값을 사용
	(2)	출구 방향으로 주행하며 벽을 감지해야함
		<ul style="list-style-type: none"> SLAM으로 맵을 만들고 네비게이션을 사용해 초기자세를 잡고 진행하는 방향을 설정해준다. 전방에 만약 벽이 있을 경우 초음파 센서를 이용해 일정 거리 가까워지면 반응하여 모터 제어를 멈추고 차량 방향을 전환한다.
	(3)	가끔 라이다 값이 0으로 튕는 일이 발생함
		<ul style="list-style-type: none"> 아래 튕는 값 0을 제외하고 평균값을 내는 함수를 만들고 평균 낸 값을 다른 노드로 보내기 위해 퍼블리쉬를 한다 이후 값의 배열번호를 다른 노드로 퍼블리쉬 한다. data는 전반적인 각도들의 묶음, sharp는 소수개수의 각도의 묶음이다. 다음으로 data2는 왼쪽, 가운데, 오른쪽 중에서 가장 먼 곳을 가려내기 위해서 구성된다. Sort로 내림차순으로 배열을 정리하여 가장 큰 값을 맨 앞으로 보낸다. 그 다음 기존의 data2 배열에서 몇번째 값이였는지 가려내기 위해 find_same 함수를 쓴다. 값을 찾아내면 그 값의 배열번호를 다른 노드로 퍼블리쉬한다. (아래 사진 자료참고)
	(4)	네비게이션 사용 시 주변 환경을 쾌적하게 해야함
		<ul style="list-style-type: none"> 주변 환경에 따라 네비게이션이 장애물과 일정한 거리를 유지하기 위해 지속적으로 경로를 재생성하는 경우 부자연스러운 움직임이 발생할 수 있다 가급적 장애물이 많지 않은 넓은 공간에서 테스트를 진행하고 좁은 공간에서 네비게이션이 사용 시 네비게이션 파라미터 설정을 진행해야 한다.

(5) Dynamic Window Approach(DWA)

- local plan에서 주로 사용하는 기법으로, 로봇의 속도 탐색 영역(Velocity Search Space)에서 로봇과 충돌 가능한 장애물을 회피하면서 목표점까지 빠르게 다다를 수 있는 속도를 선택하는 방법입니다. 기존의 지도를 위치 기반으로 접근하던 방식에서 지도 자체를 속도 영역으로 접근합니다.

- v (병진속도), w (회전속도)
- V_s : 가능 속도 영역
- V_a : 허용 속도 영역
- V_r : 다이내믹 윈도우 안의 속도 영역



목적함수

- 목적함수 G 는 로봇의 방향, 속도, 충돌을 고려하여, 목적함수가 최대가 되는 속도 v, w 를 구하게 됩니다.
- 로봇 이동에 실제 사용되는 병진속도, 회전속도를 이용하기 때문에 좌표를 전달하는 것보다 효율적입니다.

(참고자료)

[https://roomedia.tistory.com/entry/14%EC%9D%BC%EC%B0%A8-SLAM%EA%B3%BC-Navigation-%EB%85%B8%EB%93%9C-%EC%B2%98%EB%A6%AC-%EA%B3%BC%EC%A0%95#toc-Dynamic%20Window%20Approach\(DWA\)](https://roomedia.tistory.com/entry/14%EC%9D%BC%EC%B0%A8-SLAM%EA%B3%BC-Navigation-%EB%85%B8%EB%93%9C-%EC%B2%98%EB%A6%AC-%EA%B3%BC%EC%A0%95#toc-Dynamic%20Window%20Approach(DWA))

(6)

-

(7)

-

(8)

-

(9)

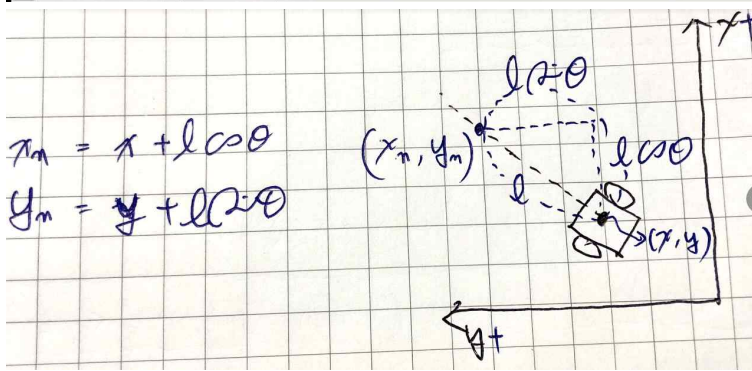
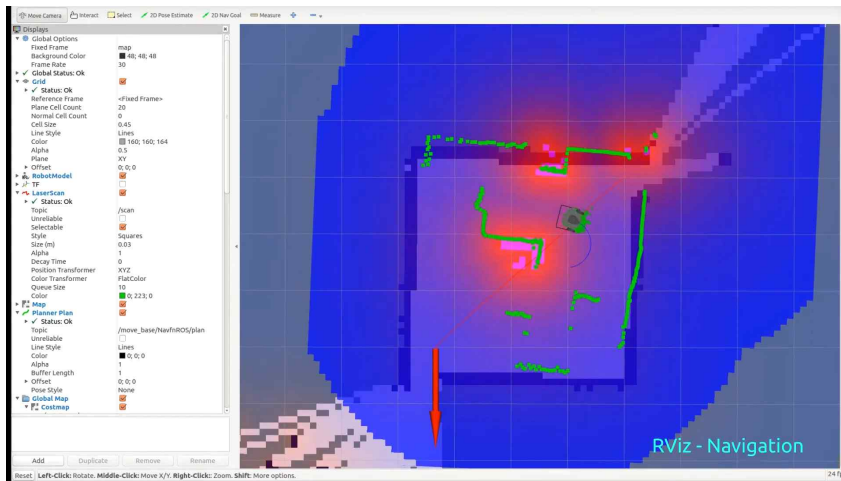
-

(10)

-

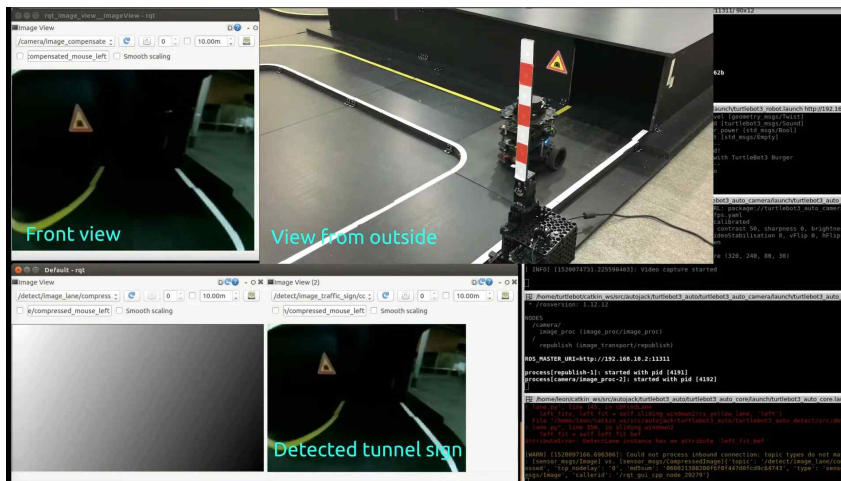
③ 기타 구현 기술과 관련된 내용 자유롭게 기술

(구현 기술과 관련된 그림 포함)



- SLAM과 Navigation 노드 처리 과정

<https://roomedia.tistory.com/entry/14%EC%9D%BC%EC%B0%A8-SLAM%EA%B3%BC-Navigation-%EB%85%B8%EB%93%9C-%EC%B2%98%EB%A6%AC-%EA%B3%BC%EC%A0%95>



- 터틀봇3 오토레이스

<https://github.com/KoG-8/Turtlebot3-RealRiceThief>

```
Front_Left_avg = avg(Front_Left)
...
```

```
def avg(data):
    return sum(data)/(len(data)-data.count(0)+0.01)
```

```
raw.data = [...]
raw.sharp = [...]
raw.front = [...]
```

```
data2 = [Front_Left_avg, Front_avg , Front_Right_avg]
```

```
data2.sort(reverse=True)
```

```
find_same(data2[0])
```

- LiDAR와 UWB를 사용하여 터널과 같은 환경에서 localizability를 추정
<https://www.ri.cmu.edu/wp-content/uploads/2019/06/root.pdf>

수행 미션 (3번)	장애물을 피해서 주행 <ul style="list-style-type: none"> 도로 위에 놓인, 차선 안쪽에 놓인 장애물과 충돌하지 않고 피해서 주행해야 함. 차선을 잠시 벗어날 수도 있으나 장애물을 모두 피한 후에는 정상적으로 차선 안쪽으로 되돌아와서 주행해야 함. 	
① 사용하는 센서별 용도	카메라	영상처리를 통해 차선 안쪽에 놓인 장애물을 위치를 파악하기 위해 사용
	초음파센서	주행 중에 전방에 있는 장애물과 충돌하는 것을 막기 위해 사용
	라이다	SLAM 기술로 차량이 트랙의 어디쯤 왔는지 위치를 파악하기 위해 사용
	IMU	자율주행차의 위치 파악을 위해 사용
② 구현할 기 능 요약, 그리고 구체적 구 현방안	(1)	주행 중 차선 안쪽에 놓인 장애물을 인식해야함
	- 미션 1번에 작성된 전방에 있는 장애물과의 충돌은 어떤 경우에도 피해야 하므로 초음파 센서를 사용하여 전방 30cm 안에 장애물이 있는지 항상 체크하는 기능을 사용.	
	(2)	장애물이 있을 때 주행
	- 초음파 센서의 초음파가 물체에 부딪쳐 반사되어 돌아오는 데까지 걸리는 시간을 측정하여 물체의 거리를 계산한다.	
	(3)	장애물 회피 후 차선으로 다시 돌아와야 함
	- 자율적으로 경로를 계획하면서 주행하는 자율주행 로봇의 핵심 기능은 효율적인 충돌 회피와 경로 생성이다. 지역 경로 계획자는 충돌 회피를 위해 짧은 시간 동안 장애물과의 충돌 없는 경로를 지역적으로 생성하고, 이 과정을 목표 지점에 도달할 때까지 반복한다. 이러한 지역 경로 계획자의 충돌 회피 알고리즘은 동적 환경에서도 효율적으로 작동해야 하며 즉각적인 충돌 회피가 가능해야 한다.	
	(4)	장애물이 없을 때 차량의 행동
	- 장애물이 없다면 차량이 나가야 하는 출구 쪽을 바라보고 주행하도록 하였다. odom 토픽이 제공하는 글로벌 좌표와 틀어진 각도와 글로벌 좌표를 사용하여 로컬 좌표를 만든다. 차량이 입장하는 순간 새로운 X-Y 좌표를 만들기 위해서 회전행렬 변환을 써서 앞으로 차량이 이동하는 좌표를 터널 입장 시에 형성되는 로컬 좌표의 단위 벡터에 의거해서 변환하여 보여준다.	
	(5)	장애물이 있을 때 차량의 행동
	- 행동보다 장애물이 있을 때의 행동이 우선순위가므로 장애물을 감지하면 바로 다른 행동을 취한다. - 전면부 라이다의 값들을 받아서 10개의 값들 중에서 6개 이상이 지정한 거리보다 적으면 막다른 벽으로 판단하고 제자리 회전을 수행한다. 정면에 그냥 벽만 있다면 왼쪽 오른쪽 사이드의 거리 값을 비교하여 거리가 먼 쪽으로 회전하도록 하였다. 만약 차량 사이드에 벽이 있다면 벽을 피하기 위해 벽의 반대 방향으로 방향을 설정	

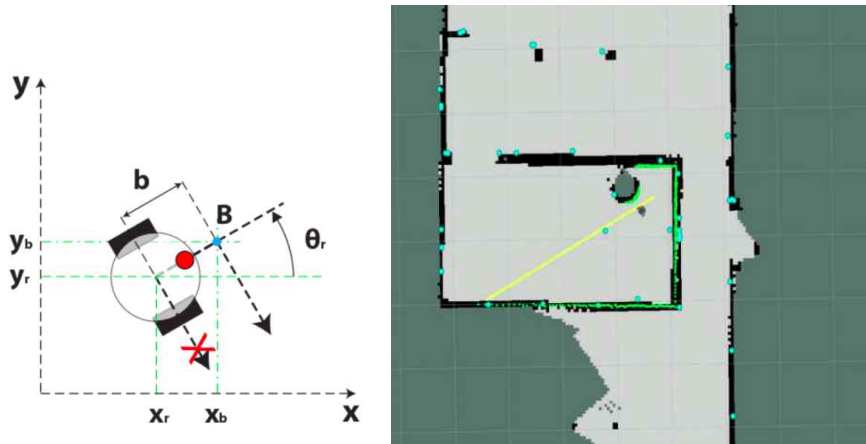
하여 이동과 동시에 회전을 한다.

(6)

-

③ 기타 구현
기술과 관련된
내용 자유롭게
기술

(구현 기술과
관련된 그림
포함)

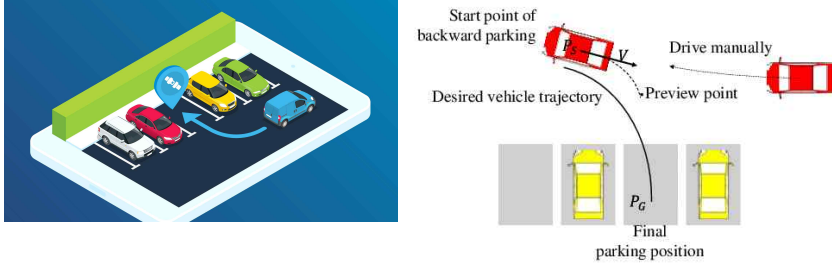
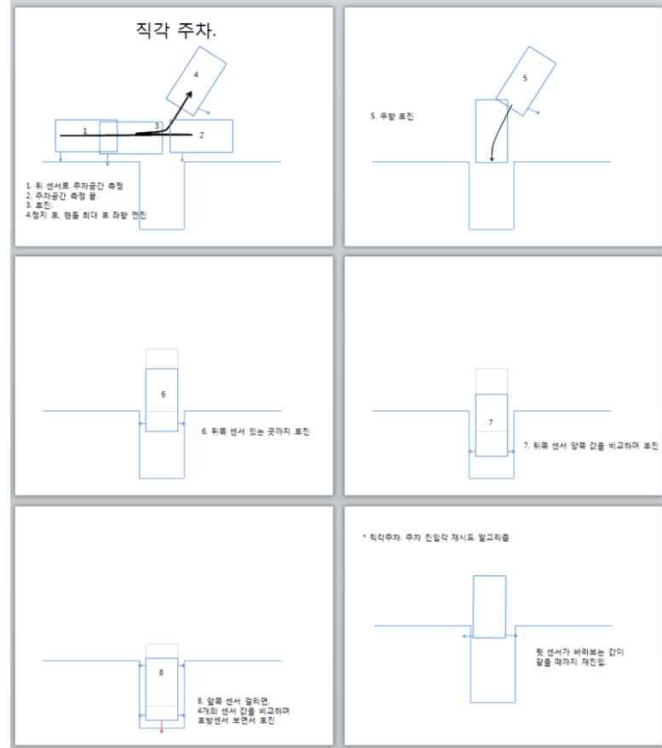


- turtlebot3_msgs
https://github.com/ROBOTIS-GIT/turtlebot3_msgs
- Mapping & Path-Following For a Two-Wheeled Robot
<https://medium.com/@sarim.mehdi.550/mapping-path-following-for-a-two-wheeled-robot-b8bd55214405>
- A Naive Obstacle Avoidance Technique for Turtlebot 3 implemented in ROS
<https://github.com/enansakib/obstacle-avoidance-turtlebot>
- 운전자의 운전 특성을 고려한 자율주행차의 human-like obstacle avoidance trajectory planning과 tracking model
<https://www.mdpi.com/1424-8220/20/17/4821/htm>

수행 미션 (4번)	정지선 정차 후 일정시간 후에 다시 재출발하여 주행 <ul style="list-style-type: none"> 정지선을 발견하면 지나치지 않고 정지선 앞에 정차해야 함. 정차 후 일정시간(3초)이 지나면 다시 출발하여야 함. 한계시간(5초)이 지나도록 계속 정차해 있으면 안 됨. 	
① 사용하는 센서별 용도	카메라	영상처리를 통해 수평선의 존재를 파악함.
	초음파센서	장애물로 인한 정지선 판별 오류를 방지하기 위해 사용.
	라이다	SLAM 기술로 차량이 트랙의 어디쯤 왔는지 위치를 파악하기 위해 사용
	IMU	차량의 현재 속도를 측정하기 위해 사용
② 구현할 기 능 요약, 그리 고 구체적 구 현방안	(1)	카메라를 통해 받아오는 이미지에서 선을 추출해야 함. <ul style="list-style-type: none"> 카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다. 컬러 이미지를 회색톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다. 잡음으로 인해 잘못된 경계선을 계산하는 것을 방지하기 위한 검출 방식을 사용하여 이미지를 만든다. 화면의 아랫부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다. 정지선의 평균 길이를 반영하여 픽셀을 랜덤하게 선택하는 방법으로 직선을 검출한다.
	(2)	정지선 검출 <ul style="list-style-type: none"> 정지선은 곧 수평선으로, 기울기가 0인 선이다. 검출된 선들 중 기울기가 오차 범위 내에서 0에 가까운 선이 있다면 정지선으로 간주한다. 오차 범위는 카메라에 의해 측정되는 평균적인 정지선들의 기울기에 따라 설정한다.
	(3)	수평선을 발견하면 차량을 멈춰야 함. <ul style="list-style-type: none"> speed, angle=(0, 0)으로 한 topic을 발행한다. 모터 배터리의 잔량에 따라 모터의 회전속도가 하드웨어적으로 변화하는 경우가 있으므로 IMU센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 체크하여 속도제어에 반영한다.
	(4)	일정 시간이 지나면 다시 출발해야 함. <ul style="list-style-type: none"> 정지선을 발견하여 차를 멈춘 후 3초가 지나면 2초 안에 다시 출발하도록 speed 정보를 포함한 topic을 다시 발행해야 한다. 정지하기 시작한 시간을 timestamp 변수로 저장하여 현재 시간과 비교해 3초 동안은 speed를 0으로, 3초가 지나면 speed를 다시 올려서 주행하도록 한다.
	(5)	전방의 장애물로 인하여 정지선이 잘못 검출되었는지 판별

	<ul style="list-style-type: none"> - 초음파 센서를 사용하여 전방에 장애물이 있는지 판단한다. - 장애물로 인해 수평선이 검출된 경우 정지선으로 판별하지 않고 장애물 회피를 실행한다. - 장애물이 있는지 판단하는 기준은 차의 너비, 카메라의 높이 등에 따라 대략 30cm로 설정한다. <div data-bbox="387 495 1474 560" style="background-color: yellow; padding: 2px;">(6)</div> <div data-bbox="387 560 1474 622" style="text-align: center;">-</div>
<p>③ 기타 구현 기술과 관련된 내용 자유롭게 기술</p> <p>(구현 기술과 관련된 그림 포함)</p>	<div style="display: flex; justify-content: space-around; align-items: flex-start;">   </div> <div style="text-align: center; margin: 10px 0;">  </div> <ul style="list-style-type: none"> • Hough Line Transform https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

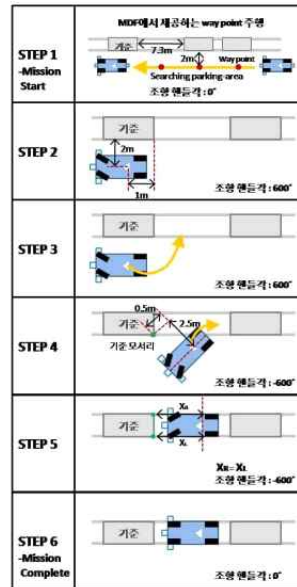
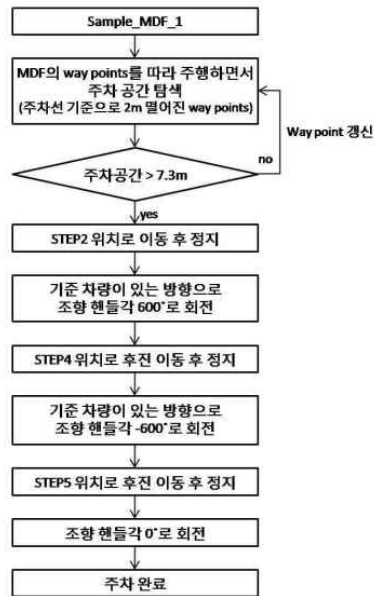
수행 미션 (5번)	T자 수직주차 후 다시 출발 <ul style="list-style-type: none"> 후진으로 2대의 차량 사이에 수직으로 주차함. 주차 과정에서 양쪽 차량과 접촉사고가 나거나 후면 벽과 충돌하면 안됨. 3초 정차한 후 다시 빠져나와 주행트랙으로 복귀하여 계속 주행해야 함. 5초 이상 계속 주차하고 있으면 안됨. 	
① 사용하는 센서별 용도	카메라	-
	초음파센서	차량과 차량 사이의 공간 측정 및 후면 벽과의 거리 측정을 위해 사용
	라이다	SLAM 기술로 차량의 위치를 파악하고 주차구간 측정을 위해 사용
	IMU	차량의 현재 속도를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적인 구현방안	(1)	센서를 통해 주차해야 할 공간을 인식해야 함. <ul style="list-style-type: none"> 라이다를 통해 주차구간을 측정하고 수직 주차 알고리즘을 실행시킨다. 첫 번째 차량과 두 번째 차량을 지나면서 차량 사이 공간을 측정 하고 측정이 끝나면 첫 번째 차량위치로 후진하여 이동한다. 첫 번째 차량과 두 번째 차량 사이 공간을 측정할 때 시간을 측정하여 수직 주차인지 수평 주차인지 판단한다.
	(2)	주차구간 확인 후 차량을 회전시켜서 주차구간 입구로 이동해야 함. <ul style="list-style-type: none"> 차량을 주차구간 입구로 이동시키기 전에 주차구간측정을 시작한 장소로 차량을 이동시킨다. 핸들을 주차구간 반대 방향으로 돌린 후 차를 이동시키고 다시 핸들을 반대 방향으로 돌려 차를 후진시키면서 주차구간 입구로 이동한다. 차량이 주차구간으로 이동을 하기 위해 회전할 때 차량이 트랙 밖으로 나가지 않도록 거리 조절을 해주어야 한다.
	(3)	센서를 통해 양쪽 차량과의 접촉이 없게 주차구간에 진입해야 함. <ul style="list-style-type: none"> 라이다를 사용하여 차량이 주차구간으로 후진할 때 다른 차량과의 거리가 어느정도인지 확인하고, 주차가 가능하다면 후진해서 주차한다. 주차가 불가능하다고 판단하였을 경우 차량을 회전시켜서 다른 차량과의 충돌이 없게 방향을 바꾸고 다시 주차구간에 진입한다.
	(4)	주차구간에 주차 시 후면 벽과의 충돌이 없어야 함. <ul style="list-style-type: none"> 초음파센서를 통해 후면 벽과 차량의 위치를 확인하면서 차량을 주차시킨다. 주차 시 양쪽 차량과의 충돌도 없어야 하므로 차량의 방향을 일직선으로 하고 핸들을 움직이지 않으면서 후진한다. 주차구간에 3초간 정차 후 빠져나와야 하며, 5초 이상 주차하고 있지 않게 해야 하므로 시간을 측정해 주는 코드를 프로그래밍 해주어 정해진 시간에 차량이 빠져나올 수 있도록 해준다.

	<div data-bbox="389 271 1468 367"> <div data-bbox="389 271 456 367">(5)</div> <div data-bbox="469 271 1468 367">주차구간으로 차량 진입 시 각도가 틀어진 상태로 진입할 경우 재진입 해주어야 함.</div> </div> <div data-bbox="389 367 1468 577"> <div data-bbox="389 367 1468 479">- 후진할 때 라이다를 통해 다른 차량과의 위치를 확인하고 차량이 주차를 끝내기 전에 다른 차량과의 거리가 너무 가깝다면 차량을 주차구간에서 나오게 하고, 차량의 각도를 바꾸어 다시 후진하여 주차한다.</div> <div data-bbox="389 479 1468 577">- 양쪽 차량과 충돌하기 전에 차량이 빠져나와야 하므로 판단을 빠르게 할 수 있도록 프로그래밍을 해주어야 한다.</div> </div> <div data-bbox="389 577 1468 651"> <div data-bbox="389 577 456 651">(6)</div> <div data-bbox="469 577 1468 651"></div> </div>
<div data-bbox="167 1171 360 1317">③ 기타 구현 기술과 관련된 내용 자유롭게 기술</div> <div data-bbox="167 1361 341 1473">(구현 기술과 관련된 그림 포함)</div>	<div data-bbox="389 719 1225 981">  </div> <div data-bbox="389 987 1054 1733">  </div> <div data-bbox="389 1778 1460 1928"> <ul style="list-style-type: none"> shared and cooperative control을 통한 운전자 지원 및 운전 실력 향상 <p>https://www.researchgate.net/publication/309476799_Simultaneous_Achievement_of_Supporting_Human_Drivers_and_Improving_Driving_Skills_by_Shared_and_Cooperative_Control</p> </div>

수행 미션 (6번)	평행주차 <ul style="list-style-type: none"> 후진 또는 전진으로 트랙과 수평으로 놓인 주차구역에 주차함. 벽 또는 장애물과 충돌하면 안됨. 주차구역 앞쪽 벽에 붙여 놓은 AR코드를 이용하여 차량이 AR코드를 정면으로 바라보는 위치에 똑바로 정확히 주차해야 함. 	
① 사용하는 센서별 용도	카메라	카메라 센서를 이용해 AR 코드를 읽고 주차구역인지 인식한 후 정확한 주차 위치를 추정한다.
	초음파센서	측면 2개, 후방에 3개의 초음파 센서를 설치하여 주차 공간을 측정한다.
	라이다	-
	IMU	-
② 구현할 기 능 요약, 그리고 구체적 구현방안	(1) 측면의 주차 공간을 인식하고 주차 공간 산출.	
	<ul style="list-style-type: none"> 초음파 센서를 차량의 각각 모서리 부분에 좌측과 우측을 바라보도록 총 2개를 부착한다. 차량이 진행하면서 측정되는 측면 공간의 폭이 평행주차를 위한 최소 폭보다 클 경우 주차 공간으로 인식을 시작하게 되며, 그 이후 차량의 이동 거리가 평행주차를 위한 최소 길이 이상이 될 때까지 측면 공간이 최소 폭 이상을 유지하면 평행주차 가능 공간으로 결정 차량이 전진하면서 측면에 부착된 초음파 센서로 진행 방향 길이와 측면 방향의 폭을 측정하여 공간의 면적을 산출 	
	(2) 후진 주차를 위한 공간 확보.	
	<ul style="list-style-type: none"> 주차 시 후진 공간 확보를 위해 차량의 우측 후면 초음파 센서가 주차 공간을 지나 다시 장애물을 인식한 시점에 모터 제어를 멈추고 차량을 일시 정지시킨다. 	
	(3) 조향각을 조절하여 후진으로 주차(주차 공간과 일직선이 되어야 함)	
	<ul style="list-style-type: none"> 차량 기준의 x축, y축에서 측간거리와 조향각으로 회전반경을 계산하고 회전반경으로 주차 경로를 계산하여 후진을 통해 주차 시킨다. 주차 경로를 설정할 때 후진하며 주차할 경우 차량의 위치와 자세는 뒷바퀴 축의 중심을 기준으로 표현한다. 전체 경로는 후면 직선 이동을 위해 조향각 0일 때와 조향각이 시계방향, 반시계방향일 때 세 가지 경우로 나누어 조건을 세운다. 최소 회전반경을 위하여 조향각을 최대로 설정한다. 	
	(4) 주차 공간의 앞뒤 간격을 맞추어 주차	
	<ul style="list-style-type: none"> -카메라 센서를 통해 전방의 AR 코드를 인식하여 주차 공간 전방과 차량 사이의 공간 좌표값과 자세 정보값을 받아와 전후방의 주차 공간을 인식하고 모터를 제어를 통해 차량 속도를 조절하여 지정된 주차 공간의 간격을 맞추어 준다. 	
	(5)	

(6)

<주차 단계>



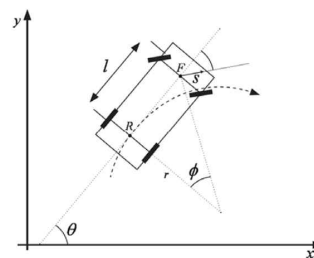
③ 기타 구현
기술과 관련된
내용 자유롭게
기술

(구현 기술과
관련된 그림
포함)

<회전 반경 계산 공식>

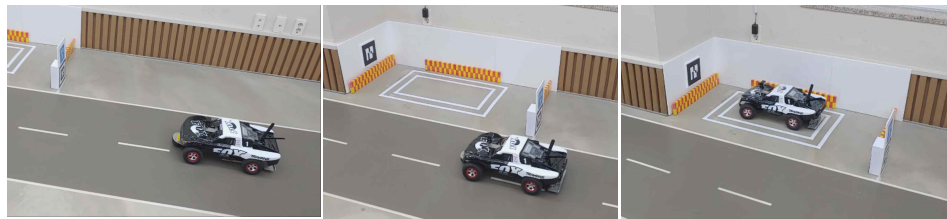
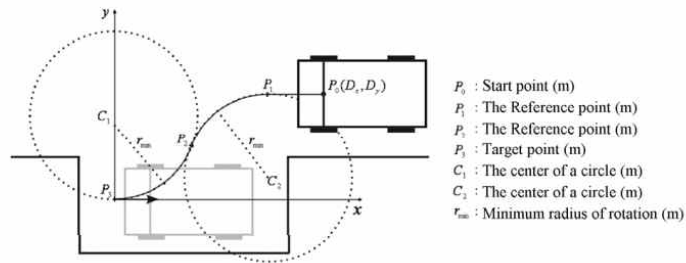
$$\begin{cases} \Delta x_R = s \cos \phi \cos \theta \\ \Delta y_R = s \cos \phi \sin \theta \\ \Delta \theta = \frac{s}{l} \sin \phi \\ r = \frac{l}{\sin \phi} \end{cases}$$

$$\begin{cases} x_R' = x_R + s \cos \phi \cos \theta \\ y_R' = y_R + s \cos \phi \sin \theta \\ x_F = x_R + l \cos \theta \\ y_F = y_R + l \sin \theta \\ \theta' = \theta + \frac{s}{l} \sin \phi \end{cases}$$



ϕ : Wheel angle (deg)
 s : Distance that the vehicle has moved (m)
 F : The center of the front axis (m)
 R : The center of the rear axis (m)
 l : Wheelbase (m)
 r : A radius of rotation (m)

<주차 경로 계산 공식>



- Nonholonomic 이동수단의 자동 평행 주차

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=566343>

- A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles

<https://reader.elsevier.com/reader/sd/pii/S0950705115001604?token=3F3331473B6CC222F801E2C29C0651B4CA3FEBEF284B78467E2C9AC8F4E50FBB61C68980BAD9073758430BD8BDBB8BB6&originRegion=us-east-1&originCreation=20220604062427>