

# 제5회 국민대 자율주행 경진대회 예선경기 #과제3

## 자율주행 SW 설계서

### 1. 참가팀 일반사항

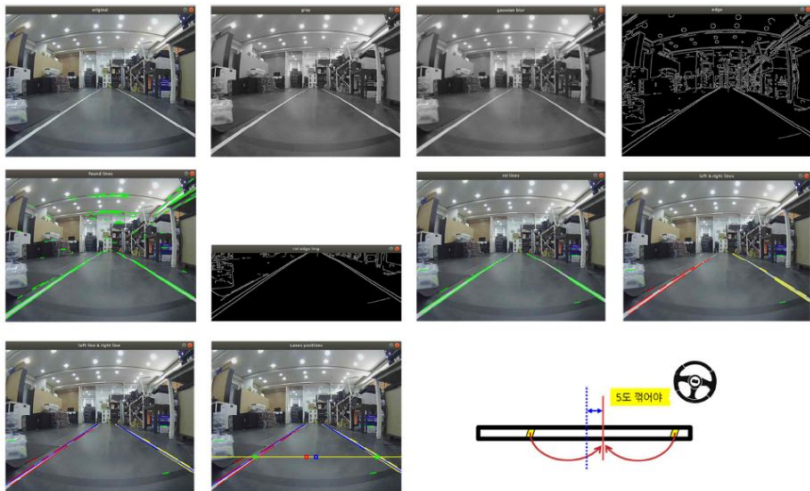
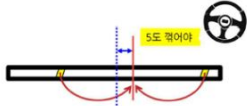
참가팀	팀명	코봇로봇
	팀장	민경서(국민대)
	팀원	윤영수(국민대), 이성빈(국민대), 임병준(국민대)

### 2. 미션 완수를 위한 자율주행 SW 설계서

1 1 1 1 1

수행 미션 (1번)	<b>차선을 벗어나지 않고 주행</b> <ul style="list-style-type: none"> <li>주행트랙에서 차량이 차선을 벗어나지 않고 주행해야 함. 코너와 곡선구간 차선 위에 세워둔 차선이탈 판정용 블록을 쓰러뜨리지 않아야 함.</li> </ul>	
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"> <li>카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.</li> <li>칼라 이미지를 회색톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.</li> <li>화면의 아래부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.</li> <li>흰색점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.</li> <li>왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.</li> <li>화면의 중심을 기준으로 왼쪽차선과 오른쪽 차선이 동일한 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.</li> <li>화면의 중심 기준선에서 왼쪽차선이 오른쪽 차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로</li> </ul>	

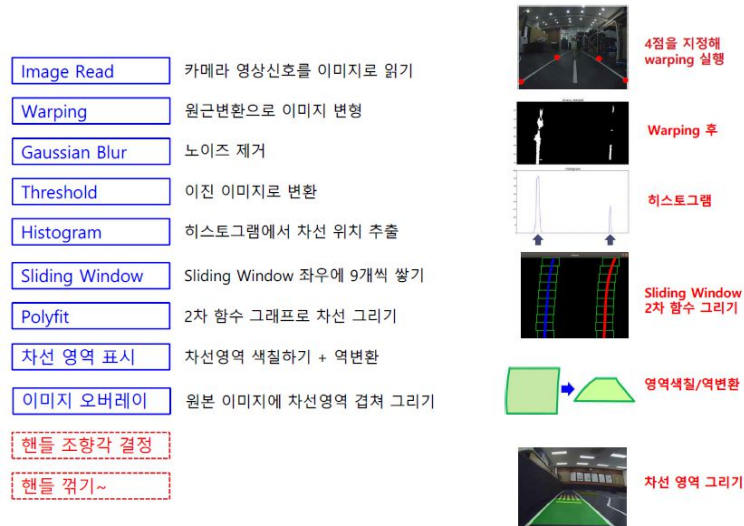
	<p>로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다.</p> <ul style="list-style-type: none"> <li>- 트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.</li> </ul>
(2)	<p>직선 구간에서는 빨리, 곡선 구간에서는 천천히 달려야 함.</p> <ul style="list-style-type: none"> <li>- 핸들의 조향각을 살펴서 좌우 꺾임 각도가 5도 이하이면 직선이라고 간주하고 속도를 크게 높인다.</li> <li>- 꺾임 각도가 5도 이상이면 곡선 구간이라고 간주하고 속도를 늦춘다.</li> <li>- S자 곡선처럼 중간에 회전 방향의 좌우가 바뀌면서 직선 구간이 잠깐 등장하는 경우가 있으므로 약간의 시간을 두고 여러 번 관찰하는 방식으로, 핸들의 꺾임 정도를 정확하게 파악하여 반응하도록 한다.</li> <li>- 갑작스러운 가속보다는 속도를 점진적으로 올리거나 내리는 방식으로 차량의 속도를 제어한다.</li> <li>- 모터 배터리의 잔량에 따라 모터의 회전속도가 하드웨어적으로 변화하는 경우가 있으므로 IMU센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 체크하여 속도제어에 반영한다.</li> </ul>
(3)	<p>핸들을 너무 자주, 그리고 급격하게 꺾으면 안됨. (안정적으로 주행하게끔)</p> <ul style="list-style-type: none"> <li>- 속도가 빠른 고속도로 주행에서는 핸들을 조금만 꺾어도 차량이 휘청이게 되며, 잘못하면 차체가 중심을 잃어 사고가 날 수도 있다.</li> <li>- 차량의 속도가 빠를 경우에는 조금씩 핸들을 좌우로 계속 왔다갔다 꺾는 방식의 운전은 피한다. 특정 값 이하의 작은 핸들 조작은 Skip 하고, 그 이상의 핸들 조작만 반영하는 방식으로 핸들을 조작한다.</li> <li>- 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) 핸들 조작 각도를 일부러 작게 만든다.</li> <li>- 이런 상황이 일정 횟수 이상 반복되면 차량의 속도를 빨리 줄인다. 차량의 속도가 줄면 가중치 값을 곱할 필요가 없으므로 핸들 조작량이 정상으로 되돌아와서 차량이 차선을 벗어나지 않도록 신속한 핸들링이 가능해질 것이다.</li> </ul>
(4)	<p>차선 인식에 실패하면 원래 가던 방향으로 계속 주행해야 함.</p> <ul style="list-style-type: none"> <li>- 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.</li> <li>- 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.</li> </ul>
(5)	<p>코너에 진입하기 전에 속도를 늦추고 코너에 진입한 후엔 속도를 높여야 함.</p> <ul style="list-style-type: none"> <li>- 고속 주행시에는 코너에 진입하기 전에 속도를 늦추는 것이 타당하다. 그리고 일단 코너에 진입하여 핸들이 꺾인 상태가 되면 속도를 다시 높여도 된다.</li> <li>- 카메라 영상에 윗부분을 살펴서, 즉 조금 먼 곳에 있는 차선을 살펴서 코너구간으로</li> </ul>

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

- OpenCV 영상처리 기반 원근변환과 슬라이딩윈도우 기법을 이용한 차선인식

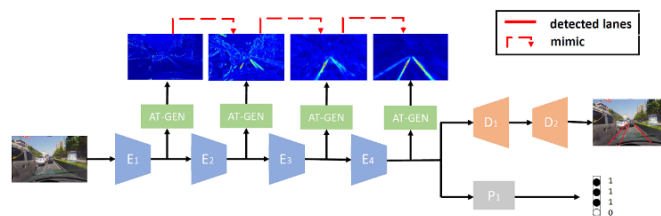
(참조사이트 <https://medium.com/geekculture/advanced-techniques-for-lane-finding-self-driving-cars-fc1a147fc49>)

a)



- 머신러닝 기반의 객체인식 모델을 이용한 차선인식

(참조사이트 [https://github.com/InhwanBae/ENet-SAD\\_Pytorch](https://github.com/InhwanBae/ENet-SAD_Pytorch))



수행 미션 (2번)	<b>어두운 터널을 통과</b> <ul style="list-style-type: none"> <li>카메라 영상으로 차선을 인식하는 것이 불가능한 어두운 터널 안에서 거리센서(초음파센서 또는 라이다)를 이용해서 벽과 충돌하지 않으면서 주행하여 터널을 빠져 나와야 함.</li> </ul>	
① 사용하는 센서별 용도	카메라	어두운 터널 안에서 영상으로 차선 인식 불가능
	레이더	주행 중 전방에 있는 코너 미리 검출을 위해 사용
	라이다	양쪽 벽 인식 및 장애물, 주행 차량을 파악하기 위해 사용
	초음파센서	라이다로 인식할 수 없는 근접 사각지대를 초음파센서를 이용해 물체를 인식하는 데 사용
	IMU	차량의 현재 속도와 자세를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적 구현방안	(1)	터널 진입과 탈출을 인식함.
		<ul style="list-style-type: none"> <li>카메라 영상인식이 빛에 의해 불가능하다고 판단되고, 양쪽 벽면의 라이다 센서 값이 특정 임계값 이하로 유지되면 터널 진입으로 인식한다.</li> <li>카메라 영상인식이 빛에 의해 가능하다고 판단되고, 양쪽 벽면의 라이다 센서 값이 특정 임계값 이상으로 커지면 터널 탈출로 인식한다.</li> </ul>
	(2)	양쪽 벽을 인식하고 차량이 차선의 중간쯤에서 달려야함.
		<ul style="list-style-type: none"> <li>주행 차량 좌우의 라이다 센서 값을 비교하여 일정 오차 이상의 값이 나타나면 벽과 충돌할 우려가 있어 차량의 주행 방향을 조절한다.</li> <li>라이다 센서의 왼쪽 벽과의 거리 값 &gt; 오른쪽 벽과의 거리 값의 경우 차량이 차선의 방향보다 오른쪽으로 주행되고 있음으로 파악하여 핸들 조향각을 살짝 왼쪽으로 꺾어 차량의 방향을 정렬하고, 왼쪽 벽과의 거리 값 &lt; 오른쪽 벽과의 거리 값의 경우</li> <li>차량이 차선의 방향보다 왼쪽으로 주행되고 있음으로 파악하여 핸들 조향각을 살짝 오른쪽으로 꺾어 차량의 방향을 정렬한다.</li> <li>라이다 센서의 왼쪽 오른쪽 센서 값의 차이가 유지되는 경우, 핸들 조향각을 센서 값 차이에 비례하게 키운다.</li> </ul>
	(3)	장애물과 충돌하지 않고 달려야함. (다른 차량이 있다면 안전거리 유지)
		<ul style="list-style-type: none"> <li>라이다 센서를 이용해 차량 진행방향 전후의 장애물을 파악한다.</li> <li>전후방 라이다 센서값이 일정 거리 값 이상으로 유지된다면 현재 주행상태를 유지한다.</li> <li>전방 라이다 센서값이 일정 거리 값보다 작아진다면 감속주행한다.</li> <li>단, 센서값이 특정 임계값 이하로 측정되면 정지한다.</li> <li>후방 라이다 센서값이 일정 거리 값보다 작아진다면 거리에 반비례하게 가속도를 높여 가속주행한다.</li> <li>모터의 배터리 잔량에 따라 모터의 회전속도가 하드웨어적으로 바뀌는 경우가 있음</li> </ul>

	<p>므로 IMU센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 체크하여 속도제어에 반영한다.</p>
(4)	<p>핸들을 너무 자주, 그리고 급격하게 꺾으면 안됨. (안정적으로 주행)</p> <ul style="list-style-type: none"> <li>- 속도가 빠른 고속도로 주행에서는 핸들을 조금만 꺾어도 차량이 휘청이게 되며, 잘 못하면 차체가 중심을 잃어 사고가 날 수도 있다.</li> <li>- 차량의 속도가 빠를 경우에는 조금씩 핸들을 좌우로 계속 왔다갔다 꺾는 방식의 운전은 피한다. 특정 값 이하의 작은 핸들 조작은 Skip 하고, 그 이상의 핸들 조작만 반영하는 방식으로 핸들을 조작한다.</li> <li>- 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.</li> <li>- 이런 상황이 일정 횟수 이상 반복되면 차량의 속도를 빨리 줄인다. 차량의 속도가 줄면 가중치 값을 곱할 필요가 없으므로 핸들 조작량이 정상으로 되돌아와서 차량이 차선을 벗어나지 않도록 신속한 핸들링이 가능해질 것이다.</li> </ul>
(5)	<p>코너에 진입하기 전에 속도를 늦추고 코너에 진입한 후엔 속도를 높여야 함.</p> <ul style="list-style-type: none"> <li>- 레이더 센서를 전방과 대각선 좌우에 설치해 전방의 코너를 검출한다.</li> <li>- 전방 레이더 센서값이 감소할때, 좌우 레이더 센서 값을 비교해 코너의 좌우를 파악한다.</li> <li>- 코너의 유무를 파악한 후, 코너 진입시에는 속도를 낮게 유지한다.</li> <li>- IMU 센서를 사용하여 모터의 하드웨어적 문제에 따른 속도 변화를 예방하고, 차량이 진행되는 속도를</li> <li>- 조절한다.</li> <li>- 코너 검출이 더 이상 없다고 판단될 때, 다시 속도를 높여 주행한다.</li> </ul>
(6)	<p>직선 구간에서는 빨리, 곡선 구간에서는 천천히 달려야 함.</p> <ul style="list-style-type: none"> <li>- 핸들의 조향각을 살펴서 좌우 꺾임 각도가 5도 이하이면 직선이라고 간주하고 속도를 크게 높인다.</li> <li>- 꺾임 각도가 5도 이상이면 곡선 구간이라고 간주하고 속도를 늦춘다.</li> <li>- 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.</li> <li>- 갑작스러운 가속보다는 속도를 점진적으로 올리거나 내리는 방식으로 차량의 속도를 제어한다.</li> <li>- 모터 배터리의 잔량에 따라 모터의 회전속도가 하드웨어적으로 변화하는 경우가 있으므로 IMU센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 체크하여 속도제어에 반영한다.</li> </ul>
(7)	<p>-</p>

	(8)	
	-	
	(9)	
	-	
	(10)	
	-	
<p>③ 기타 구현 기술과 관련된 내용 자유롭게 기술</p> <p>(구현 기술과 관련된 그림 포함)</p>	<div data-bbox="383 712 1385 1070"> <p>— tunnel / pipe</p> <p>↓ surface normal / contact force</p> <p>○ robot and laser</p> <p>□ sliding block</p> </div> <p>초음파 및 레이더 센서 측정 방법 예시  (참조사이트 <a href="https://cafe.naver.com/xytron">https://cafe.naver.com/xytron</a>,  <a href="https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7">https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7</a>)</p> <p>라이다 센서 값 측정 및 장애물, 벽 인식 예시  (참조사이트 <a href="https://www.youtube.com/watch?v=mJTPL7IVU78">https://www.youtube.com/watch?v=mJTPL7IVU78</a>)</p> <div data-bbox="391 1496 853 1908"> </div> <div data-bbox="941 1473 1449 1814"> </div>	

수행 미션 (3번)	<b>장애물을 피해서 주행</b> <ul style="list-style-type: none"> <li>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.</li> </ul>	
① 사용하는 센서별 용도	카메라	영상처리를 통해 차도에서 양쪽 차선의 위치를 파악하기 위해 사용, 카메라를 이용해서 객체를 인식하고 판단
	초음파센서	라이다로 인식할 수 없는 사각지대를 초음파 센서를 이용해 물체를 인식하여 충돌을 방지하는 데 사용
	라이다	도로 위에 놓인 물체, 장애물을 인식하는 데 사용
	IMU	차량의 현재 속도를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적 구현방안	(1)	차선 인식 & 차선의 중간쯤에서 주행.
		<ul style="list-style-type: none"> <li>- 카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.</li> <li>- 칼라 이미지를 회색톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.</li> <li>- 화면의 아래부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.</li> <li>- 흰색점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.</li> <li>- 왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.</li> <li>- 화면의 중심을 기준으로 왼쪽차선과 오른쪽 차선이 동일한 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.</li> <li>- 화면의 중심 기준선에서 왼쪽차선이 오른쪽 차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.</li> </ul>
	(2)	라이다 센서로 장애물을 인식하기
		<ul style="list-style-type: none"> <li>- 라이다 센서가 발광부에서 발사하는 레이저 펄스가 도로 위에 놓인 물체에 반사되어 돌아오는 동안의 비행시간을 측정하여 물체와의 거리를 계산한다.</li> <li>- 라이다 센서를 통해 인식한 물체를 카메라를 통해 어떤 물체인지 판단한다.</li> <li>- 만약 자동차이면, 그대로 주행한다.</li> <li>- 아니라면, 장애물이라고 판단한다.</li> </ul>
	(3)	물체를 인식하고 판단하기
		<ul style="list-style-type: none"> <li>- 장애물이라고 판단한 후에 상, 하, 좌, 우를 확인한다.</li> <li>- if 상, 하, 좌, 우에 물체가 없다면, 원래 속도에서 점점 속도를 줄여나가면서 왼쪽이</li> </ul>



	<p>나 오른쪽 둘 중에 하나의 방향으로 핸들을 꺾어서 왼쪽 차선이나 오른쪽 차선이 있는 쪽으로 차량의 방향을 튼다.</p> <ul style="list-style-type: none"> <li>- elif 왼쪽에 물체가 없다면, 핸들을 왼쪽으로 꺾어 왼쪽 차선이 있는 쪽으로 차량의 방향을 튼고 속도를 낮추어 물체를 피해서 주행한다. (이때, 왼쪽 차선을 넘어도 상관 없음)</li> <li>- elif 오른쪽에 물체가 없다면, 핸들을 오른쪽으로 꺾어 오른쪽 차선이 있는 쪽으로 차량의 방향을 튼고 속도를 낮추어 물체를 피해서 주행한다. (이때, 오른쪽 차선을 넘어도 상관없음)</li> <li>- elif 뒤에 물체가 없다면, 속도를 늦추고 왼쪽이나 오른쪽에 차가 지나가는 것을 기다린 후 왼쪽, 오른쪽 둘 중에 한쪽으로 물건을 피해서 지나간다.</li> <li>- else(모든 위치에 물체가 있다면): 최악의 상황을 피한다.</li> <li>- if 라이다 센서로 물체를 인식하고 카메라 센서로 어떤 물체인지 인식하여 장애물이 사람이 아니라면, 장애물 쪽으로 계속 이동한다.</li> <li>- else 사람이라면, 속도를 늦춘다.</li> </ul>	
	(4)	<p>핸들을 틀 때, 급격하게 꺾으면 안됨.(안정적으로 주행하게끔 함)</p> <ul style="list-style-type: none"> <li>- 속도가 빠른 고속도로 주행에서는 핸들을 조금만 꺾어도 차량이 휘청이게 되며, 잘 못하면 차체가 중심을 잃어 사고가 날 수도 있다.</li> <li>- 차량의 속도가 빠를 경우에는 조금씩 핸들을 좌우로 계속 왔다갔다 꺾는 방식의 운전은 피한다. 특정 값 이하의 작은 핸들 조작은 Skip 하고, 그 이상의 핸들조작만 반영하는 방식으로 핸들을 조작한다.</li> <li>- 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) 핸들 조작 각도를 일부러 작게 만든다.</li> <li>- 이런 상황이 일정 횟수 이상 반복되면 차량의 속도를 빨리 줄인다. 차량의 속도가 줄면 가중치 값을 곱할 필요가 없으므로 핸들 조작량이 정상으로 되돌아와서 차량이 차선을 벗어나지 않도록 신속한 핸들링이 가능해질 것이다.</li> </ul>
	(5)	<p>장애물을 피한 후 주행트랙으로 복귀하고, 원래 가던 방향으로 계속 주행</p> <ul style="list-style-type: none"> <li>- 장애물을 잘 피했다면, 장애물을 피하기 위해 실행된 알고리즘의 역순으로 다시 주행트랙으로 복귀한다.</li> <li>- 장애물을 피할 때 썼던 속도와 각도를 다시 직진할 때 썼던 속도와 각도로 설정하고 정상 주행한다.</li> </ul>
	(6)	
		-
	(7)	
		-
	(8)	

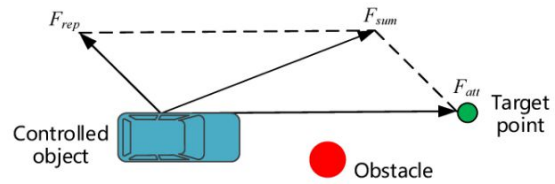
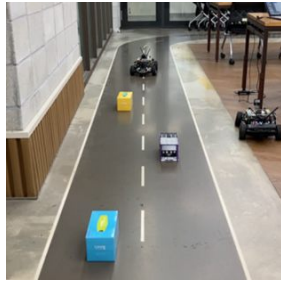
-

(9)

-

(10)

-



라이더 센서, 카메라 활용 장애물 인식 예시

(참조사이트 <https://cafe.naver.com/xytron>,

<https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7>)

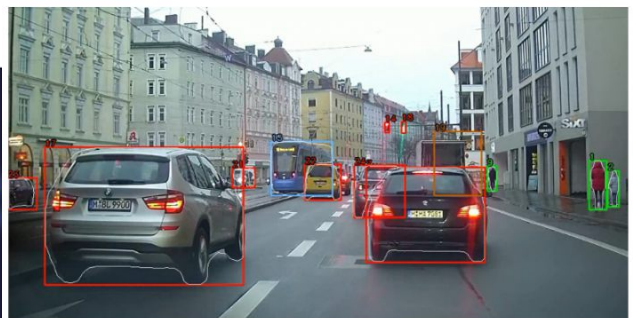
③ 기타 구현  
기술과 관련된  
내용 자유롭게  
기술

(구현 기술과 관  
련된 그림 포함)

라이더 센서, 카메라 활용 장애물 인식

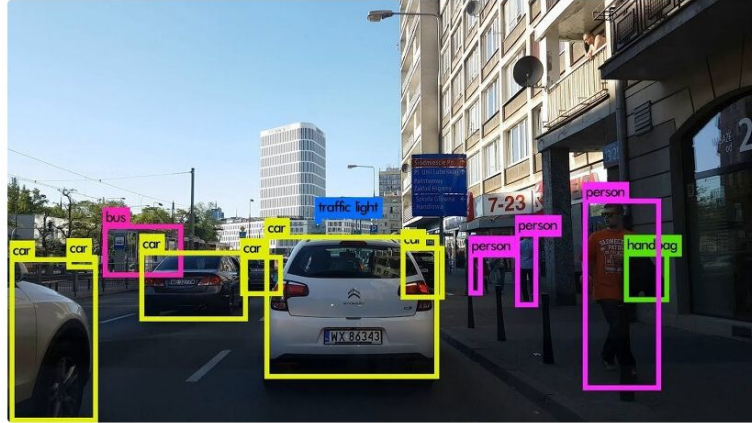
(참조사이트

<https://stradvision.com/ko/%EC%8A%A4%ED%83%80%ED%84%B0-up-%EC%9E%90%EB%8F%99%EC%B0%A8-%EC%9E%90%EC%9C%A8%EC%A3%BC%ED%96%89%EC%9D%84-%EC%9C%84%ED%95%9C-%EB%88%88-%EC%8A%A4%ED%8A%B8%EB%9D%BC%EB%93%9C%EB%B9%84%EC%A0%BC/>)



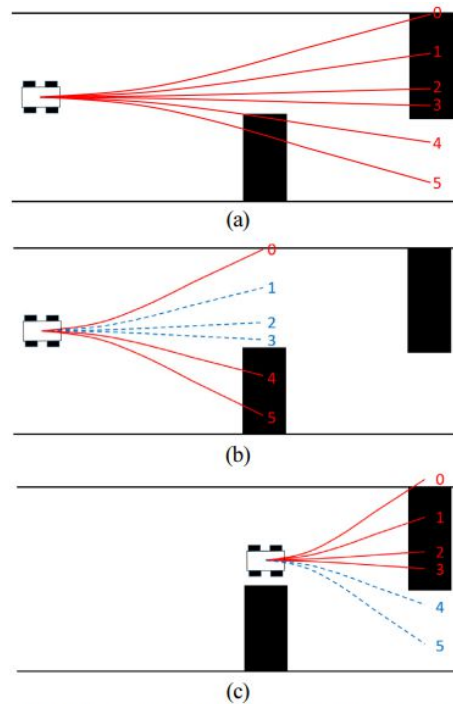
카메라(YOLO)를 활용한 객체 인식

(참조사이트 <https://pjreddie.com/darknet/yolo/>)



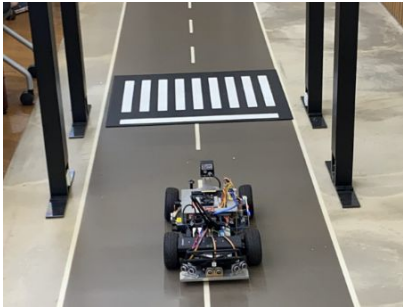
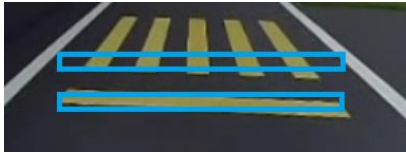
동적 및 정적 물체 회피를 위한 경로 계획

(참조사이트 <http://ikros.org/xml/29290/29290.pdf>)



[Fig. 5] Avoidance problems and solutions to an exceptional occasion. (a) No available paths (solid), (b) available paths (dashed) for avoiding the first obstacle, (c) available paths (dashed) for avoiding the second obstacle

수행 미션 (4번)	<b>정지선 정차 후 일정시간 후에 다시 재출발하여 주행</b> <ul style="list-style-type: none"> <li>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.</li> </ul>	
① 사용하는 센서별 용도	카메라	영상처리를 통해 차도에서 양쪽 차선의 위치를 파악하기 위해 사용, 영상처리를 통해 정지선 인식
	초음파센서	라이다로 인식할 수 없는 사각지대를 초음파 센서를 이용해 물체를 인식하여 충돌을 방지하는 데 사용
	라이다	도로 위에 놓인 물체, 장애물을 인식하는 데 사용
	IMU	차량의 현재 속도를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적 구현방안	(1)	<b>차선 인식 &amp; 차선의 중간쯤에서 주행.</b> <ul style="list-style-type: none"> <li>카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.</li> <li>칼라 이미지를 회색톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.</li> <li>화면의 아래부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.</li> <li>흰색점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.</li> <li>왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.</li> <li>화면의 중심을 기준으로 왼쪽차선과 오른쪽 차선이 동일한 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.</li> <li>화면의 중심 기준선에서 왼쪽차선이 오른쪽 차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.</li> </ul>
	(2)	<b>설정된 roi 영역에 가로 선이 검출되면 정지선으로 판단.</b> <b>[ 백색이 70퍼센트 이상일 경우 정지선으로 판단하기(or roi 안에 있는 백색 선의 두께와 길이의 범위가 맞으면 정지선으로 판단하기) ]</b> <ul style="list-style-type: none"> <li>차선을 인식하는 roi 영역 말고 정지선을 인식하는 roi 영역을 새로 설정해주고 이 부분만 처리한다.</li> <li>차선을 따라 달리다가 roi 영역에 백색이 70퍼센트 이상 인식된다면, 정지선으로 판단하고 속도를 점차 늦춰 속도를 0으로 하여 정지시킨다.</li> <li>차선을 따라 달리다가 roi 영역에 가로 직선이 인식된다면, 정지선으로 판단하고 속도를 점차 늦춰 속도를 0으로 하여 정지시킨다.</li> <li>모터의 배터리 잔량에 따라 모터의 회전속도가 하드웨어적으로 바뀌는 경우가 있으므로 IMU 센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 확인하여 속</li> </ul>

	도제어에 반영한다.	
	(3)	3초 동안 정지하고 3초가 지나면 다시 출발.
	<ul style="list-style-type: none"> <li>- IMU 센서로부터 값을 받아 속도가 0이 되었을 시 정지한 것으로 판단하고 3초의 시간을 켜 후에 다시 속도를 높여 기존 주행을 유지한다.</li> </ul>	
	(4)	만약 한계시간(5초)을 초과한 경우, 다시 출발시킴.
	<ul style="list-style-type: none"> <li>- 특정 이유로 시간을 초과한 경우, 핸들 조향각을 0으로 하고 속도를 지정한 후에 출발하도록 한다.</li> </ul>	
	(5)	주행 및 정차 중 장애물과 충돌 방지.
	<ul style="list-style-type: none"> <li>- 주행 및 정차 동작 중 초음파 센서의 거리가 임계값 보다 작게 측정되면 정지한다.</li> <li>- 장애물 체크 시점은, 영상처리 작업을 통해 핸들 조향각과 차량의 속도를 결정한 후 모터제어 토픽을 발행하기 직전에 수행하면 된다.</li> <li>- 장애물이 감지되는 경우에는 모터제어 토픽에서 속도 값을 0으로 세팅하는 방법으로 장애물 충돌을 방지한다.</li> </ul>	
	(6)	
	-	
	(7)	
	-	
	(8)	
	-	
	(9)	
	-	
	(10)	
	-	
<p>③ 기타 구현 기술과 관련된 내용 자유롭게 기술</p> <p>(구현 기술과 관련된 그림 포함)</p>	 	

카메라 활용 정지선 인식 예시

(참조사이트 <https://cafe.naver.com/xytron>,

<https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7>)

정지선 검출 roi 영역 설정 및 edge 위치, 방향 설정 기반으로 RANSAC 기반 평행선 쌍 추정

(참조사이트

<https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=TRKO201600018086&dbt=TRKO&rn=>)

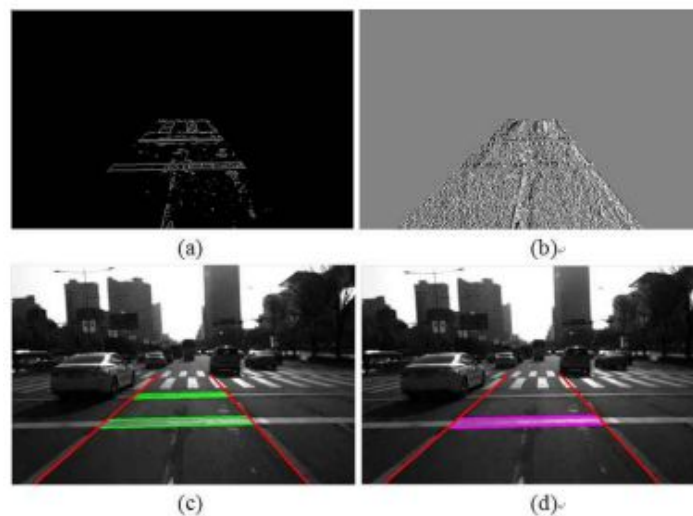


그림 84. 정지선 검출 방법. (a) 에지 위치, (b) 에지 방향, (c) 정지선 후보 생성 결과, (d) 정지선 검증 결과



그림 102. 정지선 인식 알고리즘

수행 미션 (5번)	<b>T자 수직주차 후 다시 출발</b> <ul style="list-style-type: none"> <li>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.</li> </ul>	
① 사용하는 센서별 용도	카메라	영상처리를 통한 차선 인식, 주차공간 인식을 위해 사용
	초음파센서	주행 중 전후방에 있는 장애물, 벽과 충돌하는 것을 방지하기 위해 사용
	라이다	SLAM 기술로 차량의 위치 정보 확인과 주차공간 확보를 위해 사용
	IMU	차량의 현재 속도와 자세를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적인 구현방안	(1)	차선 인식 및 주차공간 인식.
		<ul style="list-style-type: none"> <li>- 카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.</li> <li>- 칼라 이미지를 회색 톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.</li> <li>- 화면의 아랫부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.</li> <li>- 흰색 점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.</li> <li>- 왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.</li> <li>- 화면의 중심을 기준으로 왼쪽차선과 오른쪽차선이 같은 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.</li> <li>- 화면의 중심 기준선에서 왼쪽차선이 오른쪽차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.</li> <li>- 라이다 센서의 값을 받아 SLAM 기술로 현재 차량의 위치 정보를 받는다.</li> <li>- 라이다 센서와 SLAM 기술로 주차공간을 인식한다.</li> </ul>
	(2)	주차공간을 발견한 이후, 주차 및 이동 시에는 속도를 줄여 주행.
		<ul style="list-style-type: none"> <li>- 카메라로 AR 태그를 인식하고, 라이다 센서의 위치 정보로 주차공간을 인식한 후에는 낮은 속도로 주행한다.</li> <li>- 모터의 배터리 잔량에 따라 모터의 회전속도가 하드웨어적으로 바뀌는 경우가 있으므로 IMU 센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 확인하여 속도제어에 반영한다.</li> </ul>
	(3)	주차를 위한 Way Point로 이동.
		<ul style="list-style-type: none"> <li>- 라이다 센서와 SLAM 기술로 인식한 주차공간에 기준을 두고, 후진 주차 알고리즘을 실행시킬 수 있는 위치를 Way Point로 정한다.</li> <li>- 후진 주차 알고리즘 실행하기 위해 Way Point로 이동하고, IMU 센서로 차량의 자</li> </ul>



	<p>세를 확인한다.</p> <ul style="list-style-type: none"> <li>- IMU 센서를 통한 차량의 자세값이 일정 오차 범위 이내로 들어올 수 있도록 차량의 방향 및 위치를 조정한다.</li> </ul>
(4)	<p>차량 위치에 맞게 핸들 조향각을 회전하여 후진 주차하고, 차량 자세를 정렬.</p> <ul style="list-style-type: none"> <li>- 주차공간이 오른쪽에 있다면, 핸들 조향각을 오른쪽으로 +X도(ex.+90도)로 설정한 후 후진하고, 차량의 IMU 센서로부터 값을 받아, 차량 후방 벽과의 각도(yaw)가 0도 일 때 정지하고 핸들 조향각을 전방 0도로 설정하고 다시 후진한다.</li> <li>- 주차공간이 왼쪽에 있다면, 핸들 조향각을 왼쪽으로 -X도(ex.-90도)로 설정한 후 후진하고, 차량의 IMU 센서로부터 값을 받아, 차량 후방 벽과의 각도(yaw)가 0도일 때 정지하고 핸들 조향각을 전방 0도로 설정하고 다시 후진한다.</li> </ul>
(5)	<p>핸들 조향각을 전방으로 맞추고 차량이 주차공간 가운데 위치하도록 정차.</p> <ul style="list-style-type: none"> <li>- 좌우의 거리를 라이다 센서를 이용해 비교하고, 차량이 주차공간의 가운데에 위치할 수 있도록 위치를 이동한다.</li> <li>- 차량 좌우 측의 옆 차 또는 벽 등의 장애물과의 거리가 일정 거릿값 이상인 경우, 주차 알고리즘을 재실행하기 위해 다시 Way Point로 이동한다.</li> <li>- 차량의 핸들 조향각을 0으로 맞춘다.</li> </ul>
(6)	<p>3초 정차한 후 다시 주행트랙으로 복귀.</p> <ul style="list-style-type: none"> <li>- T자 수직 주차 완료 시, 영상처리를 이용하여 정지선을 인식하여 정확한 위치에 3초 간 정지한 후 원래 주행하던 주행트랙으로 복귀한다.</li> <li>- T자 수직 주차를 실행했던 주차 알고리즘의 역순으로 주차공간을 빠져나간다.</li> <li>- 주행트랙 복귀 중 예상치 못한 변수나 장애물이 생길 수 있으므로 초음파 센서와 라이다 센서로 주변 지형지물과 장애물들을 인식하고 충돌을 방지한다.</li> </ul>
(7)	<p>주행 및 주차 중에 장애물과 충돌 방지.</p> <ul style="list-style-type: none"> <li>- 주행 및 주차 동작 중 초음파 센서의 거리가 임계값 보다 작게 측정되면 정지한다.</li> <li>- 장애물이 감지되는 경우에는 모터제어 토크에서 속도 값을 0으로 세팅하는 방법으로 장애물 충돌을 방지한다.</li> <li>- 후방 주차 중 장애물이 차량의 오른쪽 부분에서 감지된다면 후진을 멈추고, 핸들 조향각을 +X도보다 크게 한 상태(ex.+90도 이상)에서 일정 거리 직진한 다음, 핸들 조향각을 오른쪽으로 +X도(ex.+90도) 만큼 꺾어 다시 후진한다.</li> <li>- 후방 주차 중 장애물이 차량의 왼쪽 부분에서 감지된다면 후진을 멈추고, 핸들 조향각을 -X도보다 작게 한 상태(ex.-90도 이하)에서 일정 거리 직진한 다음, 핸들 조향각을 왼쪽으로 -X도(ex.-90도) 만큼 꺾어 다시 후진한다.</li> </ul>
(8)	<p>주차공간 인식 실패 시 원래 주행을 유지.</p> <ul style="list-style-type: none"> <li>- 라이다 센서의 값을 받아 SLAM 기술로 현재 차량의 위치 정보를 받는다.</li> <li>- 라이다 센서와 SLAM 기술로 주차공간을 인식한다.</li> <li>- 카메라로 주차공간의 AR 태그를 발견하고, 주차공간임을 인식한다.</li> <li>- 주차공간 인식에 실패하면 기존 주행상태를 유지한다.</li> </ul>

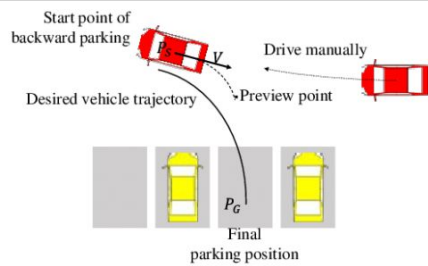


(9)

-

(10)

-



T자 수직 주차 예시

(참조사이트 <https://cafe.naver.com/xytron>

<https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7>)

GPS로 획득된 사람이 직각 주차 수행 시 주차 궤적

(참조사이트 <http://koreascience.kr/article/CFKO201123552840475.pdf>)



[그림 7] GPS로 획득된 사람이 직각주차를 수행했을 때의  
주차 궤적

주차공간 인식 후 Way Point 지정 및 주차 알고리즘 실행

(참조사이트

[http://www.riss.kr/search/detail/DetailView.do?p\\_mat\\_type=be54d9b8bc7cdb09&control\\_no=8cec4aaa57d44c97ffe0bdc3ef48d419&outLink=K](http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=8cec4aaa57d44c97ffe0bdc3ef48d419&outLink=K))

③ 기타 구현  
기술과 관련된  
내용 자유롭게  
기술

(구현 기술과 관  
련된 그림 포함)

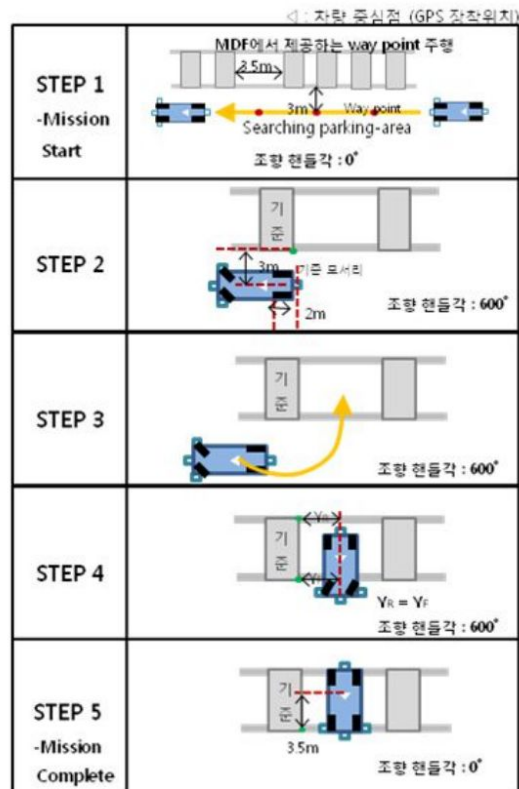
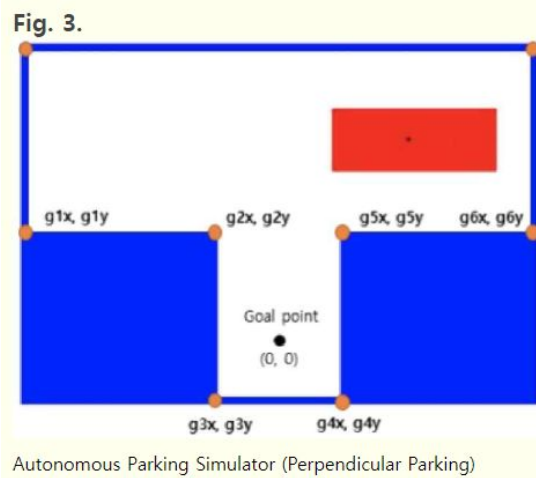


그림 4-13 직각주차 공식

Python언어로 Matplotlib 라이브러리 Pyplot 모듈을 사용한 강화학습 기반 자율주행 주차 알고리즘 적용  
(참조사이트 <http://journal.dcs.or.kr/PR/view/?aidx=23204&bidx=1886#!po=41.6667>)



수행 미션 (6번)	<b>평행주차</b> <ul style="list-style-type: none"> <li>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.</li> </ul>	
① 사용하는 센서별 용도	카메라	영상처리를 통한 차선 인식, 주차공간 및 AR 태그 인식을 위해 사용
	초음파센서	주행 중 전후방에 있는 장애물, 벽과 충돌하는 것을 방지하기 위해 사용
	라이다	SLAM 기술로 차량의 위치 정보 확인과 주차공간 확보를 위해 사용
	IMU	차량의 현재 속도와 자세를 측정하기 위해 사용
② 구현할 기능 요약, 그리고 구체적인 구현방안	(1)	차선 인식 및 주차공간, AR 태그를 발견.
		<ul style="list-style-type: none"> <li>- 카메라 ROS 토픽을 수신하여 OpenCV 이미지로 변환하여 영상처리를 진행한다.</li> <li>- 칼라 이미지를 회색 톤으로 변환하고 다시 이진화 작업을 거쳐 차선은 흰색으로 나머지는 모두 검은색으로 표시하는 이미지를 만든다.</li> <li>- 화면의 아랫부분을 관심영역(ROI)으로 설정하여 이 부분만 처리한다.</li> <li>- 흰색 점이 많이 몰려 있는 구역을 찾고 그곳을 차선이 있는 곳으로 지정한다.</li> <li>- 왼쪽차선과 오른쪽차선을 각각 찾고, 해당 위치를 저장한다. X좌표값만 저장한다.</li> <li>- 화면의 중심을 기준으로 왼쪽차선과 오른쪽차선이 같은 거리만큼 떨어져 있으면 현재 차량이 차선의 중앙을 달리고 있는 것으로 파악한다.</li> <li>- 화면의 중심 기준선에서 왼쪽차선이 오른쪽차선보다 멀리 있으면 핸들을 왼쪽으로 꺾어 왼쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 화면의 중심 기준선에서 오른쪽차선이 왼쪽차선보다 멀리 있으면 핸들을 오른쪽으로 꺾어 오른쪽차선이 있는 쪽으로 차량의 방향을 튼다.</li> <li>- 트랙 중앙에 점선으로 그려진 차선은 제외하고 좌우에 실선으로 그려진 차선만 인식해야 한다. 점선을 왼쪽차선 또는 오른쪽차선으로 인식하는 경우를 막는다.</li> <li>- 라이다 센서의 값을 받아 SLAM 기술로 현재 차량의 위치 정보를 받는다.</li> <li>- 라이다 센서와 SLAM 기술로 주차공간을 인식한다.</li> <li>- 카메라로 주차공간의 AR 태그를 발견하고, 주차공간임을 인식한다.</li> </ul>
	(2)	주차공간 및 AR 태그를 발견한 이후, 주차 및 이동 시에는 속도를 줄여 주행.
		<ul style="list-style-type: none"> <li>- 카메라로 AR 태그를 인식하고, 라이다 센서의 위치 정보로 주차공간을 인식한 후에는 낮은 속도로 주행한다.</li> <li>- 모터의 배터리 잔량에 따라 모터의 회전속도가 하드웨어적으로 바뀌는 경우가 있으므로 IMU 센서를 사용하여 실제 차량의 이동속도가 어떤지 주기적으로 확인하여 속도제어에 반영한다.</li> </ul>
	(3)	주차를 위한 Way Point로 이동.
		<ul style="list-style-type: none"> <li>- 라이다 센서와 SLAM 기술로 인식한 주차공간에 기준을 두고, 후진 주차 알고리즘을 실행시킬 수 있는 위치를 Way Point로 정한다.</li> </ul>

	<ul style="list-style-type: none"> <li>- 후진 주차 알고리즘 실행하기 위해 Way Point로 이동하고, IMU 센서로 차량의 자세를 확인한다.</li> <li>- IMU 센서를 통한 차량의 자세값이 일정 오차 범위 이내로 들어올 수 있도록 차량의 방향 및 위치를 조정한다.</li> </ul>
(4)	차량 위치에 맞게 핸들 조향각을 회전하여 후진 주차하고, AR 태그를 정면 응시할 수 있게 차량 자세를 조정.
	<ul style="list-style-type: none"> <li>- 주차공간이 오른쪽에 있다면, 핸들 조향각을 오른쪽으로 +X도로 설정한 후 후진하고, 차량의 오른쪽 모서리가 주차공간의 앞 벽을 지나치면 핸들 조향각을 -X도로 변경한 후 후진한다.</li> <li>- 주차공간이 왼쪽에 있다면, 핸들 조향각을 왼쪽으로 -X도로 설정한 후 후진하고, 차량의 왼쪽 모서리가 주차공간의 앞 벽을 지나치면 핸들 조향각을 +X도로 변경한 후 후진한다.</li> <li>- 후진 중 카메라에 AR 태그가 화면 중앙에 위치하면, 후진을 멈추고 차량 자세를 조정한다.</li> <li>- AR 태그가 차량의 전방보다 왼쪽에 위치하면, 전진할 때는 핸들 조향각을 왼쪽으로 하고, 후진할 때는 핸들 조향각을 오른쪽으로 하여 전진, 후진을 반복하며 자세를 조정한다.</li> <li>- AR 태그가 차량의 전방보다 오른쪽에 위치하면, 전진할 때는 핸들 조향각을 오른쪽으로 하고, 후진할 때는 핸들 조향각을 왼쪽으로 하여 전진, 후진을 반복하며 자세를 조정한다.</li> <li>- 차량의 IMU 센서값이 일정 오차범위 내에 들어오면, 자세 조성을 그만한다.</li> </ul>
(5)	핸들 조향각을 0으로 맞추고 차량이 주차공간 가운데 위치하도록 정차.
	<ul style="list-style-type: none"> <li>- 앞뒤의 거리를 라이다 센서를 이용해 비교하고, 차량이 주차공간의 가운데에 위치할 수 있도록 위치를 이동한다.</li> <li>- 차량의 핸들 조향각을 0으로 맞춘다.</li> </ul>
(6)	주행 및 주차 중에 장애물과 충돌을 방지.
	<ul style="list-style-type: none"> <li>- 주행 및 주차 동작 중 초음파 센서의 거리가 임계값 보다 작게 측정되면 정지한다.</li> <li>- 장애물이 감지되는 경우에는 모터제어 토크에서 속도 값을 0으로 세팅하는 방법으로 장애물 충돌을 방지한다.</li> <li>- 후방 주차 중 장애물이 차량의 오른쪽 부분에서 감지된다면 후진을 멈추고, 핸들 조향각을 0으로 한 상태에서 일정 거리 직진한 다음, 핸들 조향각을 오른쪽으로 +X도 만큼 꺾어 다시 후진한다.</li> <li>- 후방 주차 중 장애물이 차량의 왼쪽 부분에서 감지된다면 후진을 멈추고, 핸들 조향각을 0으로 한 상태에서 일정 거리 직진한 다음, 핸들 조향각을 왼쪽으로 -X도 만큼 꺾어 다시 후진한다.</li> </ul>
(7)	주차공간 인식 실패시 원래 주행을 유지.
	<ul style="list-style-type: none"> <li>- 라이다 센서의 값을 받아 SLAM 기술로 현재 차량의 위치 정보를 받는다.</li> <li>- 라이다 센서와 SLAM 기술로 주차공간을 인식한다.</li> </ul>

- 카메라로 주차공간의 AR 태그를 발견하고, 주차공간임을 인식한다.
- 주차공간 인식에 실패하면 기존 주행상태를 유지한다.

(8)

-

(9)

-

(10)

-



AR태그를 활용한 평행주차 예시

(참조사이트 <https://cafe.naver.com/xytron>,

<https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7>)

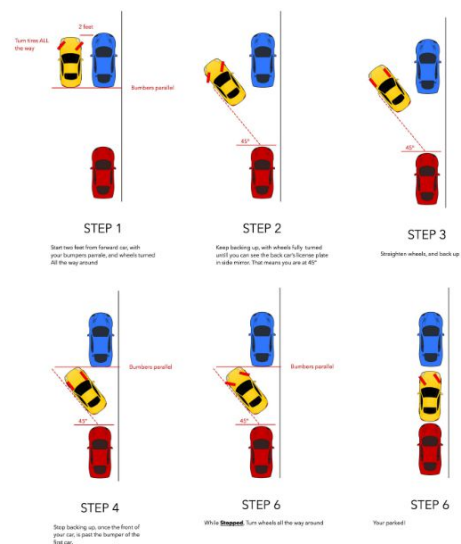
평행주차 알고리즘 예시

(참조사이트 <https://cafe.naver.com/xytron>,

<https://drive.google.com/drive/folders/1yP363VQAvsCm-cSuvj-QqV4xFLUcUX7>)

③ 기타 구현  
기술과 관련된  
내용 자유롭게  
기술

(구현 기술과 관  
련된 그림 포함)



GPS로 획득된 사람이 평행 주차 수행 시 주차 궤적

(참조사이트 <http://koreascience.kr/article/CFKO201123552840475.pdf>)



[그림 5] GPS로 획득된 사람이 평행주차를 수행했을 때  
의 주차 궤적

주차공간 인식 후 Way Point 지정 및 주차 알고리즘 실행

(참조사이트

[http://www.riss.kr/search/detail/DetailView.do?p\\_mat\\_type=be54d9b8bc7cdb09&control\\_no=8cec4aaa57d44c97ffe0bdc3ef48d419&outLink=K](http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=8cec4aaa57d44c97ffe0bdc3ef48d419&outLink=K))

◁: 차량 중심점 (GPS 장착위치)

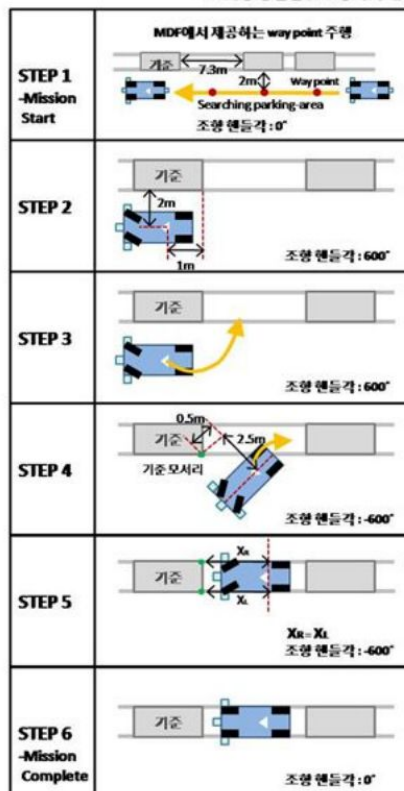


그림 4-11 평행주차 공식