임베디드 소프트웨어 경진대회
The World Embedded Software Contest

제16회 임베디드SW경진대회
개발완료보고서 [자율주행 모형자동차]

산업통상자원부

KESSIA 임베디드소프트웨어·시스템산업협회

1. 작성 시 주의사항

※ 제출 분량 : A4 용지 상세내용 포함 20 page 이내 (최대 폰트 크기 12pt)
※ 제출 포맷 : pdf

2. 팀 정보

팀명	SPILL	팀장	김준태
팀원	이호영	팀원	허민호
팀원	김무성	팀원	조재욱

3. 본 개발완료보고서

0. 작품 제목 : ASCC(Advanced Smart Cruise Control)기능을 탑재한 무인 자동차

1. 개요

1.1. 작품 개요

사물인터넷, IT, 로봇공학, 인공지능 등 첨단기술이 발전하며 고부가 기능의 시스템이 구축되고 있습니다. 이러한 기술들은 인간에게 이로움과 편리한 환경을 제공하면서 삶의 질을 높이는 데 유용합니다. 자율주행기술 또한 이러한 개발 중 하나이다. 현재 자율주행기술은 ADAS 기술을 이용해 운전자의 승차 컨디션을 보조해주는 역할로써 활용되고 있지만, 완전한 자율주행을 위해서는 지속적인 기술 개발을 필요로 하고 있다.

우리는 자율주행자동차를 상용화하기 위해 실제로 일어날 수 있는 상황들을 소형화하여 트랙으로 제작하고 그 상황들을 자율주행 모형자동차가 판단한 후, 안전하게 수행할 수 있도록 설계 및 제어 하는 것을 목표로 한다.

1.2. 필요성

자율주행자동차는 차세대 스마트카에 포함되어있습니다. 다양한 전자 기구와 소프트웨어가 융합되어 탑승자에게 도움을 주고 있습니다. 자율주행자동차의 필요성에 대해 말씀드리겠습니다.

① 운전자의 여가시간 증가 - 운전자 없이 자동차를 주행하기 때문에 이동시간을 여가시간으로 활용할 수 있습니다.

② 운전자 컨디션 향상 – 운전자를 대신하여 조향과 제동 등 판단을 하기에 운전자의 스트레스와 피로를 줄일 수 있습니다.

③ 안전성 향상 - 장애물에 반응하는 속도가 운전자의 인지하는 시간보다 빠르고, 교통법을 위반하지 않게 됩니다. 교통사고의 대다수의 운전자로 인한 과실에 따른 사고가 감소시킴으로 인명 피해를 줄일 수 있습니다.

④효율적 주행 – 갑작스런 장애물과 주행 환경을 인식 및 판단하여 목적지에 도달하기 까지 효율적으로 연료를 사용할 수 있다.

1.3. 개발 목표

1) 영상처리를 통한 명확한 차선 인식
2) 경사가 있는 도로 주행 시 안정적인 속도 제어
3) 장애물이나 돌발 상황을 파악하는 정상적인 주행

4) 적외선 센서를 이용한 자동 평행, 수직 주차
5) 안정적인 로터리 진입 및 전후방 장애물 감지를 통한 주행
6) 터널 진입을 인지한 후 안정적인 주행
7) 전방 차량을 감지하고 유동적인 회피 주행
8) 신호등의 지시에 알맞은 피드백 반응 및 주행

2. 작품 설명

2.1. Software 구성

차량이 각기 장애물을 인식하여 그에 따른 대처 방법에 대한 코드로 구성되어 있다.

먼저, 각 장애물 순서에 따른 대처 방법을 함수화 시키고, 함수의 실행은 step_cnt라는 변수의 값을 통해 실행한다. 각 단계에서 대처 방법을 수행한 후 step_cnt를 증가시키며 이전에 수행한 routine의 재실행을 방지한다.

장애물의 인식방법은 Camera, IR Sensor, PSD Sensor를 사용했다. Camera는 차선인식과 장애물 색상 인식에 사용되었고, IR Sensor는 정지선과 종료 지점을 인식하는데 사용되었고, PSD Sensor는 돌발 상황 장애물과의 거리인식, 주차 공간 인식, 원형코스 장애물과의 거리인식, 터미널 진입 인식, 추월 차선을 인식하는데 사용되었다. 하드웨어의 값을 읽어와 상황에 따른 루틴이 실행될 수 있게 경계 값을 지정하였다.

2.2. Software 설계도

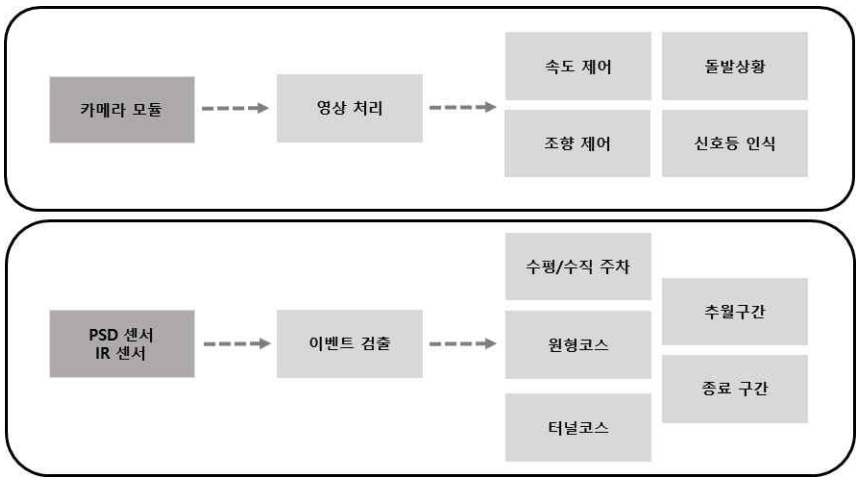
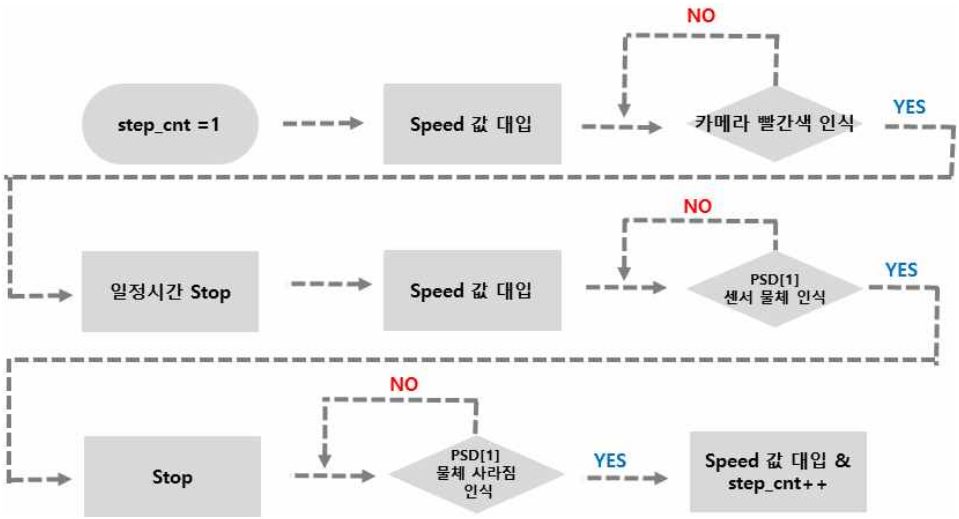


그림 4 S/W 설계도

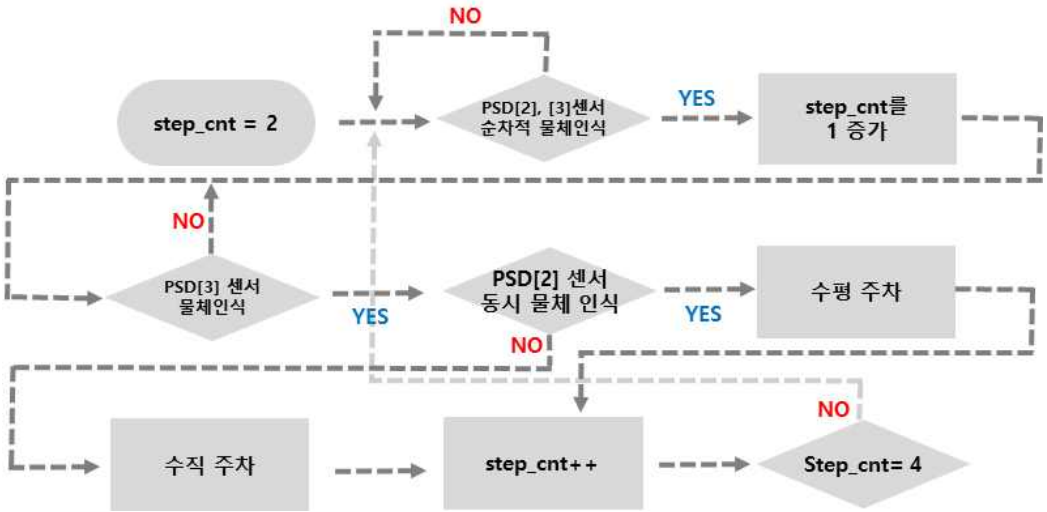
그림 5 알고리즘 흐름도

카메라 모듈을 이용해 구현한 알고리즘과 적외선 센서를 이용해 구현한 알고리즘을 분리해 보았다. S/W의 주된 알고리즘은 영상 처리를 이용해 차선을 검출하고 검출된 차선을 바탕으로 속도제어와 조향 제어를 하는 것이다. 또한 영상 처리를 이용해 특정 색상을 인식한 후 돌발 상황과 신호등의 상황을 인지하여 판단한다. 부가적으로 적외선 센서(PSD, IR)를 이용해 주행 중에 발생하는 이벤트 처리를 하도록 설계하였다.

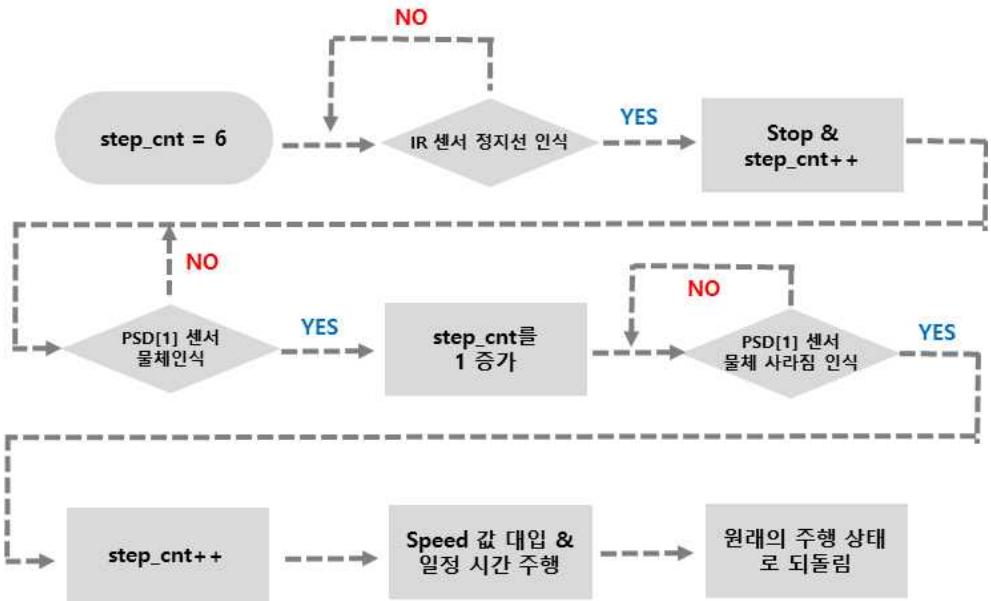
2.2.1 돌발 상황 인식



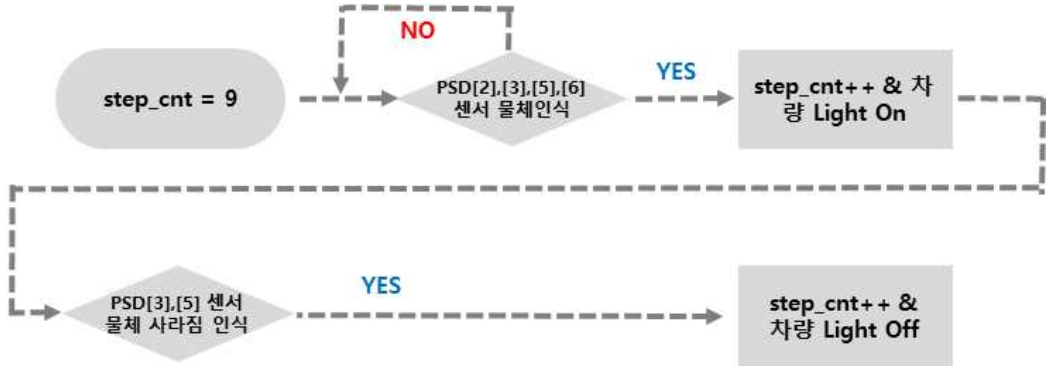
2.2.2 수평/수직 주차



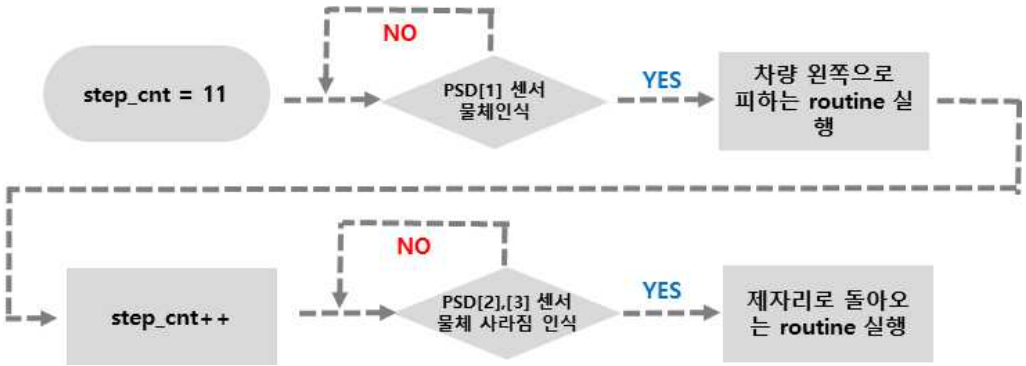
2.2.3 원형 코스



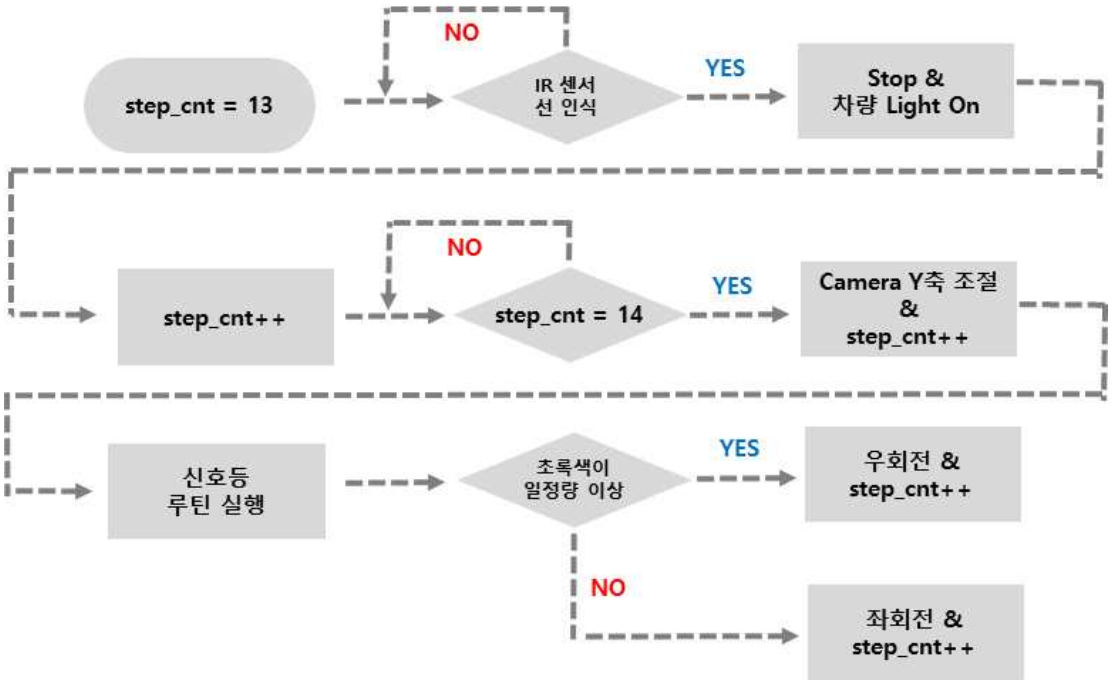
2.2.4 터널 코스



2.2.5 추월 구간



2.2.6 신호등 인식



2.2.7 종료

2.3. Software 기능

2.2.1 돌발 상황 인식

차량이 처음 주행할 때 속도와 조향각 등의 값이 지정되고 돌발 상황 루틴에 들어가게 된다. Camera에 빨간 물체가 잡히면 돌발 상황이란 것을 인식하고 속도를 낮춘다. 그 후, PSD 1번 센서로 장애물까지의 거리를 인식하여 장애물 앞까지 이동 후 정지한다. 장애물이 PSD 1번 센서로부터 인식되지 않을 때 차량은 처음 주행 설정으로 돌아가게 된다.

2.2.2 수평/수직 주차

수평수직주차의 순서는 임의로 정해짐을 염두하고 코드를 작성해야했다. 둘의 차이는 차량이 주차공간을 지나쳐 갈 때 인식되는 벽면이 수평주차공간은 차량 오른쪽 PSD 센서가 전부 인식되고, 수직주차공간은 오른쪽 뒤편 PSD 센서만 인식한다. 이 둘의 차이를 이용하여 수평주차인지 수직주차인지 판단하고 주차루틴을 실행한다.

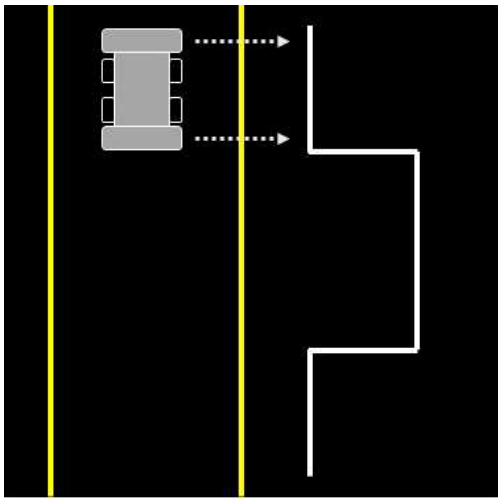


그림 1 수평주차인식

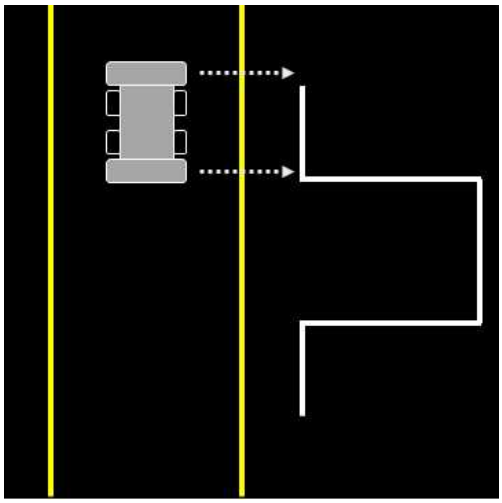


그림 2 수직주차인식

알고리즘을 보면 PSD 2, 3번 센서가 순차적으로 인식되면서 주차공간이 있다는 것을 인식한다. 두 센서가 벽면을 인식하지 않을 때 주차 공간 안을 인식하고 지나쳐 간다. 다시 PSD 3번 센서가 벽을 인식하게 되면 주차루틴이 실행된다. 이때 PSD 2번 센서가 벽면을 인식하고 있으면 수평주차루틴에 들어가고, 벽이 인식되지 않으면 수직주차루틴에 들어가게 된다.

2.2.3 원형 코스

IR 센서를 통해 정지선을 인식하고 일시정지를 한다. 장애물차량이 앞을 지나가면 일정시간 대기한다. 그 후, 일정시간 직진을 하고 원래 주행 설정으로 돌아간다.

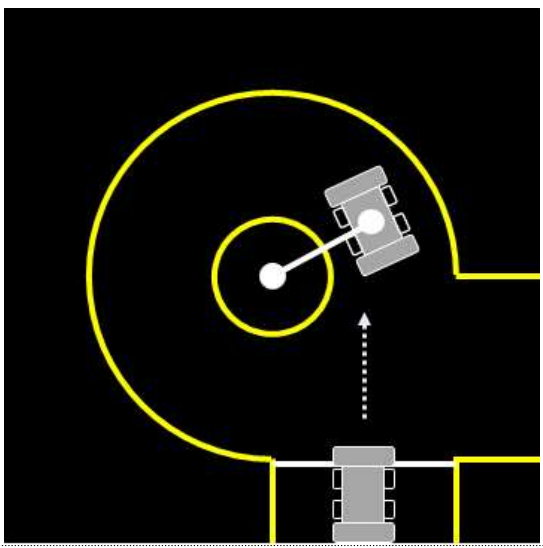


그림 3
원형코스주행

이 때 주의할 점이 있는데 주행차량이 기존의 속도로 원형코스를 주행하게 되면 속도가 빨라 장애물 차량과 부딪칠 가능성이 크다. 이를 방지하기 위해 PSD 1번 센서와 4번 센서의 값을 계속 읽어 장애물과의 거리를 인식한다. 만약 PSD 1번 센서가 장애물을 인식하면 속도를 감속시키고, PSD 4번 센서가 장애물을 인식하면 속도를 증가시킨다.

2.2.4 터널 코스

터널코스에 처음 진입하는 것은 PSD 2, 3, 5, 6번 센서가 장애물을 인식했을 때이다. 이 센서들은 차량 옆면에 배치되어있어서 터널구간 안에서는 차량의 옆면이 다 벽으로 둘러싸이는 것을 이용하였다. 차량이 터널에 진입하게 되면 카메라가 받을 수 있는 광량이 줄어들어 차선을 인식하기 힘들어짐을 볼 수 있었다. 이에 따라 터널 진입 시 차량의 Light를 전부 ON시켜서 차선인식의 오류를 줄였다.

이 후 PSD 5, 6번 센서에 장애물이 인식되지 않으면 터널을 탈출한 것이라 하고 차량의 Light를 전부 OFF시키게 하였다.

2.2.5 추월 구간

멀티라인 주행에서는 PSD 1번 센서를 통해 앞에 장애물차량이 인식되면 잠시 동안 후진한 후 좌회전을 해서 장애물 차량 우측에 나란히 위치하게 만든다. 그 후 차선을 인식하여 주행을 하다가 PSD 2, 3번 센서에 장애물이 인식되지 않을 때 잠시 동안 후진한 후 좌회전하여 기존 주행 코스로 돌아온다.

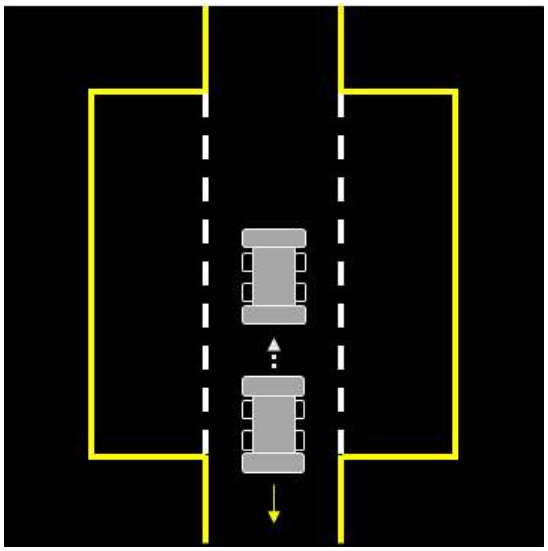


그림 4 추월 구간 주행 (1)

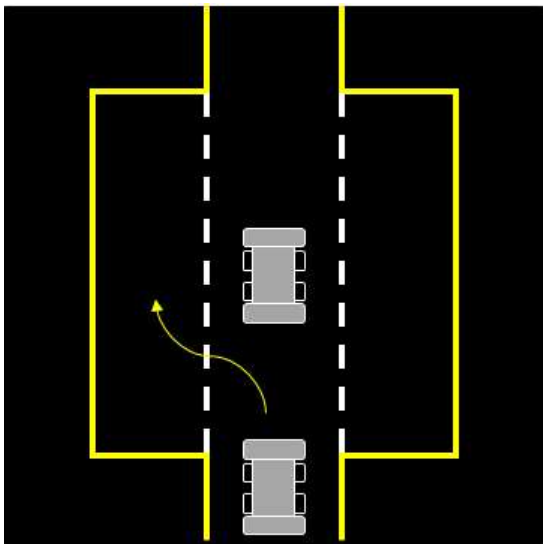


그림 5 추월 구간 주행 (2)

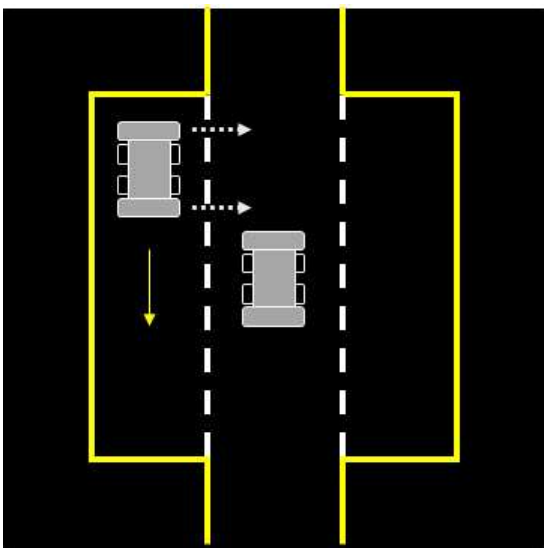


그림 6 추월 구간 주행 (3)

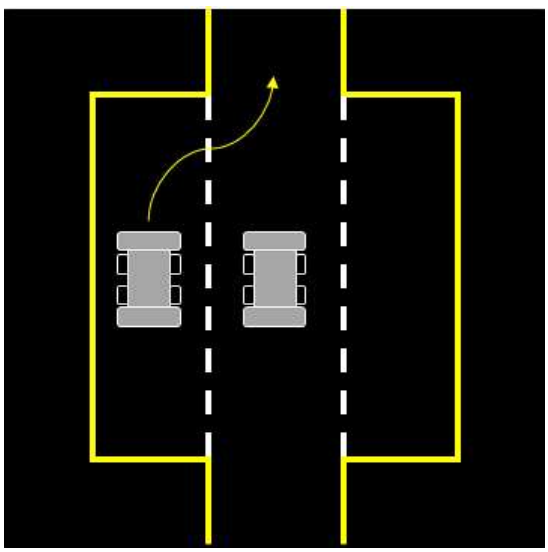


그림 7 추월 구간 주행 (4)

2.2.6 신호등 인식

IR 센서를 통해 정지선을 인식하고 카메라 Y축 값을 조절하여 카메라가 정면을 보게 만든다. 처음 빨간색을 인식하여 정지 신호인 것을 화면에 띄운다. ('RED : STOP!' 이라는 문구가 나온다.) 이후 초록색만 인식시킨다. 이때 좌회전 신호인지 우회전 신호인지는 영상에서의 초록색 픽셀 수에 따라 정한다. 초록색의 픽셀이 일정량보다 많을 시 우회전을 하고, 적을 시 좌회전을 한다. 이때 튀는 값(그림자의 인식 등)이 있을 수도 있으므로 연속 10번 인식시켜서 모두 같은 방향을 나타내고 있으면 원하는 방향으로 주행하게 하였다. 주행 시 다시 차선을 인식하면서 주행해야 해서 카메라의 Y축을 다시 원상태로 돌린다. 이후 회전하는 루틴을 실행하게 된다. 루틴이 끝나게 되면, 카메라의 차선 인식을 통해 주행한다.

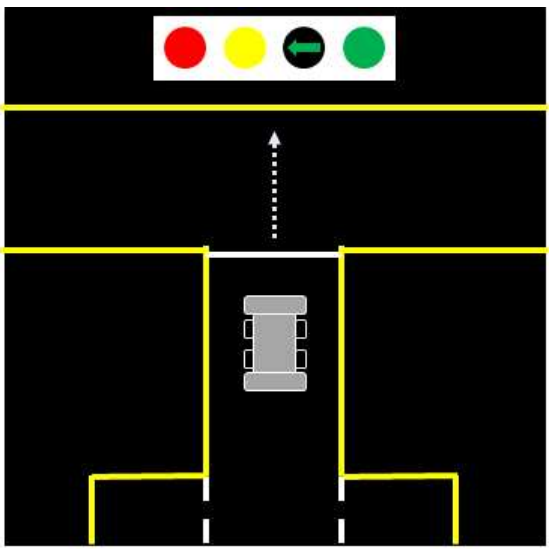


그림 8 신호등 주행 (1)

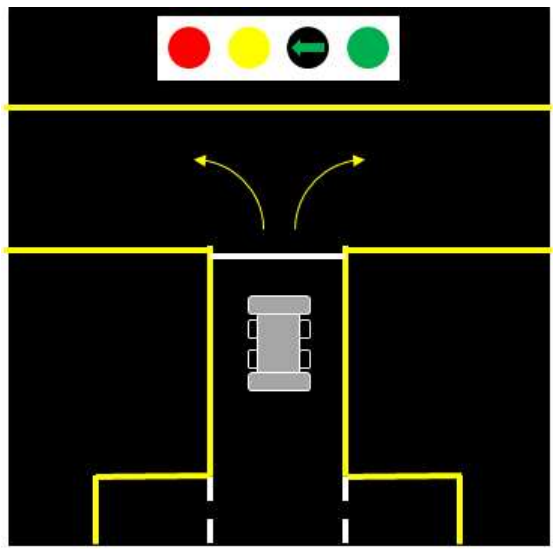


그림 9 신호등 주행 (2)

2.2.7 종료

앞에서 차선 인식하면서 주행하는데 이를 계속 유지하다가 IR 센서에 정지선이 인식되면 일정시간 앞으로 주행하다가 정지하게 만들었다. 이 부분에서 흰선과 노란선의 경계를 엣지로 인식하여 정지 공간을 여러 개의 차선으로 인식하는 문제를 해결하기 위해 canny edge의 임계값을 높혀주었다.

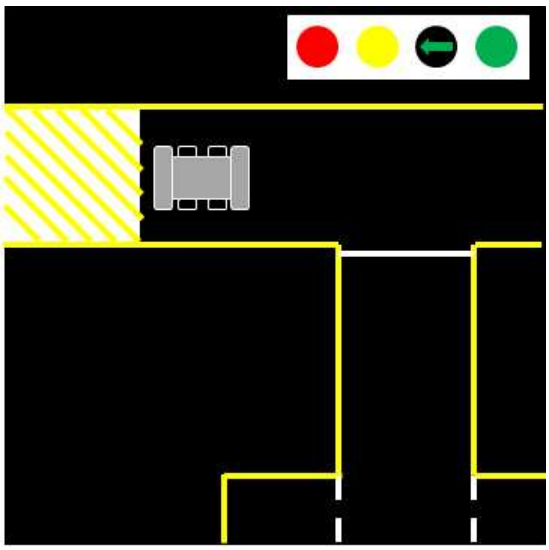


그림 10 엔드라인주행

2.4. 프로그램 사용법 (Interface)

먼저, CarSDK를 이용할 Linux 컴퓨터가 필요해 개발 PC에 Ubuntu를 설치했다. 또한 minicom과 ssh를 설치하여 차량과 데이터를 송수신할 수 있게 만들었다.

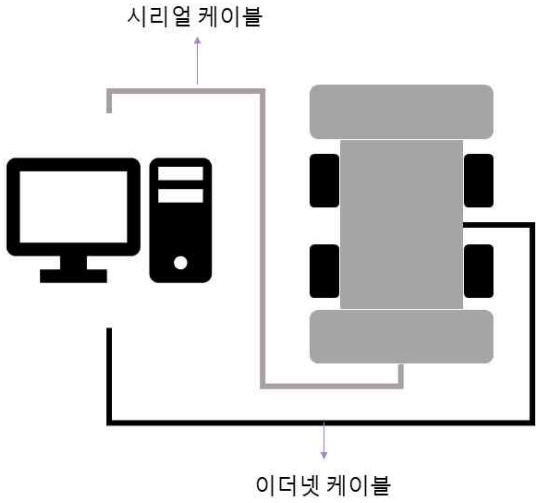


그림 11 차량의 케이블 연결

위와 같이 시리얼 포트와 이더넷을 사용하는데, 시리얼 포트는 Target Board의 터미널을 확인하고, 이더넷은 데이터 전송이 이루어진다. .profile 파일을 이용하여 차량 부팅 시 자동으로 코드가 실행 되게 만들었다.

2.5. 개발환경 (언어, Tool, 사용시스템 등)

2.5.1 언어 및 Tool

Linux 컴퓨터에 비쥬얼 스튜디오를 설치하고 코딩을 하였다. 주행 시 필요한 코드는 C언어로 작성하였고, 카메라로 차선 인식과 신호등 식별, 돌발상황인식을 하기 위해 OpenCV를 사용했는데 이는 C++을 사용한다. 이 외에 차량 통신에 필요한 프로그램은 ssh, minicom이 있다.

2.5.2 적외선 센서(PSD)

총 6개의 적외선 거리 센서를 이용하여 정밀하게 거리 제어를 할 수 있도록 거리에 따른 ADC값 측정.

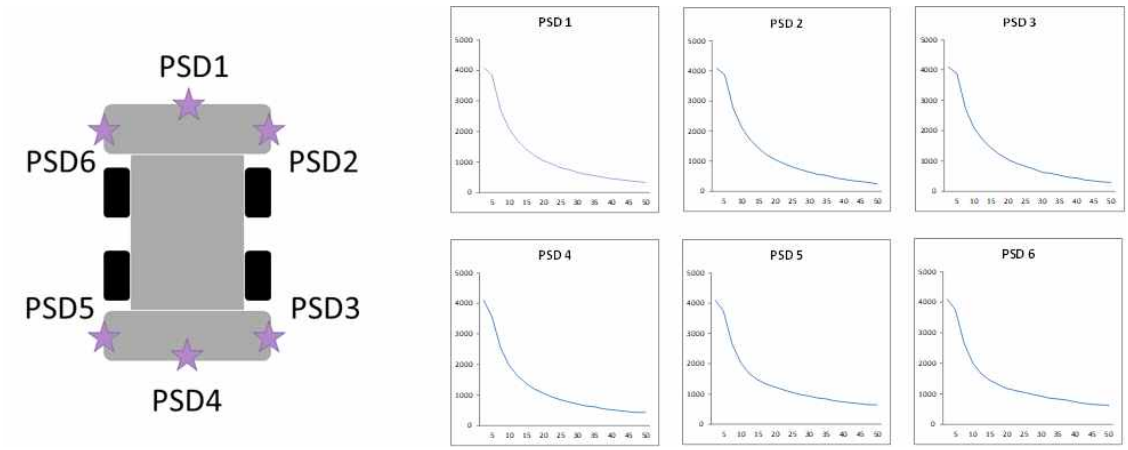


그림 12 적외선 센서 거리에 따른 측정값

6개의 그래프가 모두 유사한 형태를 보이는 것을 확인할 수 있었다. 다만 PSD5, PSD6는 다른 4개의 센서보다 거리에 대한 ADC값이 약간 크게 나타나는 것을 파악하였다. 그 이유로는 수광부가 발광부보다 위에 있어서 바닥에서 반사되는 빛의 양에 영향을 받는 것으로 판단된다.

3. 프로그램 설명

3.1. 파일 구성

① exam_cv.cpp - 영상 처리 알고리즘

- OpenCV_hough_transform()
- lkas()
- opencv_obstacle()
- colordetect()

② main.c - 주행 알고리즘

- steering()
- parking()
- horizental_park()
- vertical_park()
- CircleCourse()
- multilane()
- stopline()
- detect_cnt()
- endline()

3.2. 함수별 기능

① OpenCV_hough_transform() : 차선 검출 알고리즘

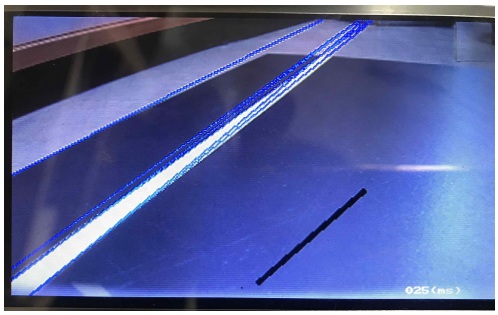


그림 13 검출된 라인

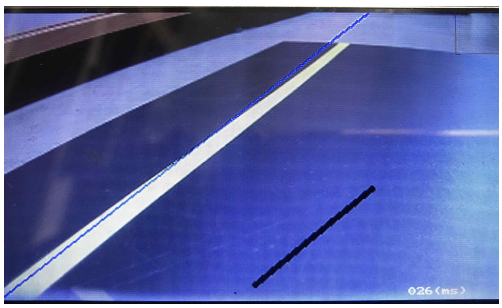


그림 14 검출된 라인들의 평균

Canny 함수를 이용해 edge를 검출한 영상을 기반으로 hough 변환을 하였다. hough 변환은 연속된 점들의 개수가 임계값보다 많으면 직선으로 인식하는 함수이다. 위와 같이 많은 선들이 검출되는 이유는 임계값을 낮게 설정했기 때문이다. 임계값을 낮게 설정해 다량의 선을 인식한 후, 검출 된 선들의 rho값과 theta값의 평균을 이용해 하나의 선으로 변환시킨다.

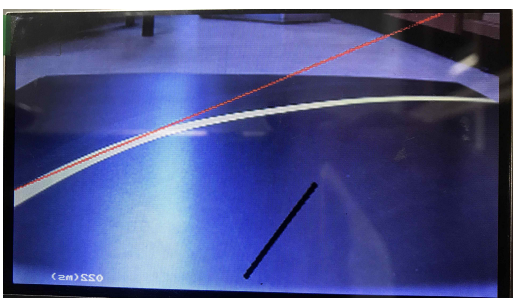


그림 15 조향각 표시

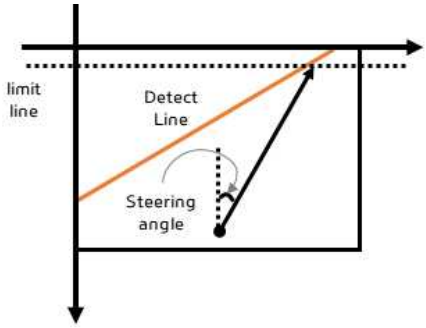


그림 16 조향각 산출 과정

검출된 라인을 바탕으로 조향각을 설정한다. 좌, 우 라인 모두 검출되었다면 두 직선의 교점과 영상의 수직 법선을 이용해 각도를 계산한다. 만약 한 라인만 검출되었다면 검출된 라인과 임의의 가로축 라인을 이용해 교점을 구하고 영상의 수직 법선과의 각도를 계산한다. 검출된 각도를 이용해 서보의 조향각을 설정해준다.

② lkas() : 차선 유지

차량이 차선의 중앙에 있지 않고 한쪽으로 쏠렸을 경우에 다시 차선 중앙으로 돌아오게 해주는 함수이다. 검출된 라인과 영상 아래의 가로축과의 교점의 x좌표를 고려하여 판단하였다.

③ opencv_obstacle() : 돌발 장애물 검출

일반 주행을 하다가 빨간 색상의 돌발 장애물을 만났을 경우 해당 색상을 인식하여 제동을 하는 함수이다.

④ colordetect() : 신호등 색상 인식

RGB영상을 HSV영상으로 변환하고 Red, Yello, Green에 대한 색상 range를 정해주어 binary 영상으로 변환한 후, 일정 픽셀 개수 이상 검출이 된다면 신호로 인식하고 주행 제어를 해주는 함수이다.

⑤ steering() - 차선 검출을 통해 얻은 각도를 이용해 조향각 설정

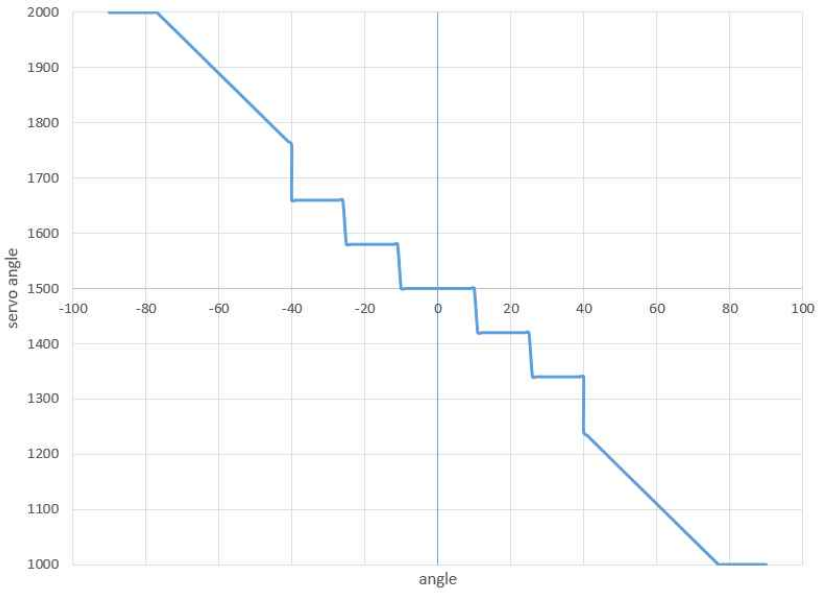


그림 19 조향각 전달 그래프

차선 검출을 통해 나온 각도를 바탕으로 조향각을 설정한다. 조향각이 작은 단계에서는 비선형적 방식으로 조향하고, 조향각이 큰 단계에서는 선형적으로 조향한다. 이러한 조향 방법을 선택한 이유는 차량의 조향각 변화가 작을 때(직진 주행)는 비선형적인 조향 방법이 더욱 안정성 있게 주행하고 차량의 조향각 변화가 클 때(코너 주행)는 선형적인 조향 방법이 더욱 안정성 있게 주행하기 때문에 이러한 방식을 선택하였다.

⑥ parking() - 수직 주차, 수평 주차 판단

PSD 2번과 3번을 이용해 주차 공간을 판단한다. 주차 공간의 형태를 이용해 수직 주차인지 수평 주차인지 판별해 해당 함수를 실행시킨다.

⑦ horizental_park() - 수평 주차

수평 주차 알고리즘을 수행한다. PSD 2, 3, 4번을 이용해 주차 공간에 부딪히지 않고 주차를 하도록 설계하였다.

⑧ vertical_park() - 수평 주차

수직 주차 알고리즘을 수행한다. PSD 2, 3, 4, 5번을 이용해 주차 공간에 부딪히지 않고 주차를 하도록 설계하였다.

⑨ CircleCourse() - 교차로 충돌방지 알고리즘

전방에 차량이 지나가는 것을 인식하고 일정 시간이 지난 후 출발한다. 만약 다른 차량이 PSD 1번에 가까워 질 경우 속도를 완전 느리게 하여 충돌을 방지한다. 반대로 PSD 4번에 가까워 질 경우 속도를 올려 교차로를 통과한다.

⑩ multilane() - 전방 차량 감지 후 차선 변경 알고리즘

PSD 1번을 이용해 전방의 물체가 감지될 경우 차선 변경 알고리즘을 수행한다. 일정 시간 뒤로 후진한 후 옆 차선으로 변경하고 PSD 2, 3번에 물체가 감지되지 않을 경우 다시 원래 차선으로 돌아온다.

⑪ stopline() - 신호등 앞 정지선 검출

IR 센서를 이용해 신호등 앞 정지선을 검출한다면 기존의 차선 검출하던 알고리즘 대신 색상 검출 알고리즘을 수행한다. 카메라의 각도를 변경하여 전방의 신호등을 보기 위해 준비한다.

⑫ detect_cnt() - 신호등의 색상 검출을 바탕으로 조향 판단

신호등의 색상을 판별해 주행 제어를 한다. 적색, 황색의 경우에는 주행하지 않고 대기하고 녹색이 검출되었을 경우 픽셀의 개수에 따라 좌, 우 조향에 대한 판단을 한다.

⑬ endline() - 종료 지점 판단 및 정지

신호등 검출 후 주행하다 IR 센서가 종료 지점을 검출할 경우 일정 시간 직진 주행 후 정지한다. 흰선과 노란색의 경계를 라인으로 인식하지 않도록 Canny함수의 threshold를 크게해 정확하게 종료지점에 들어가도록 설계하였다.

3.3. 주요 함수의 흐름도

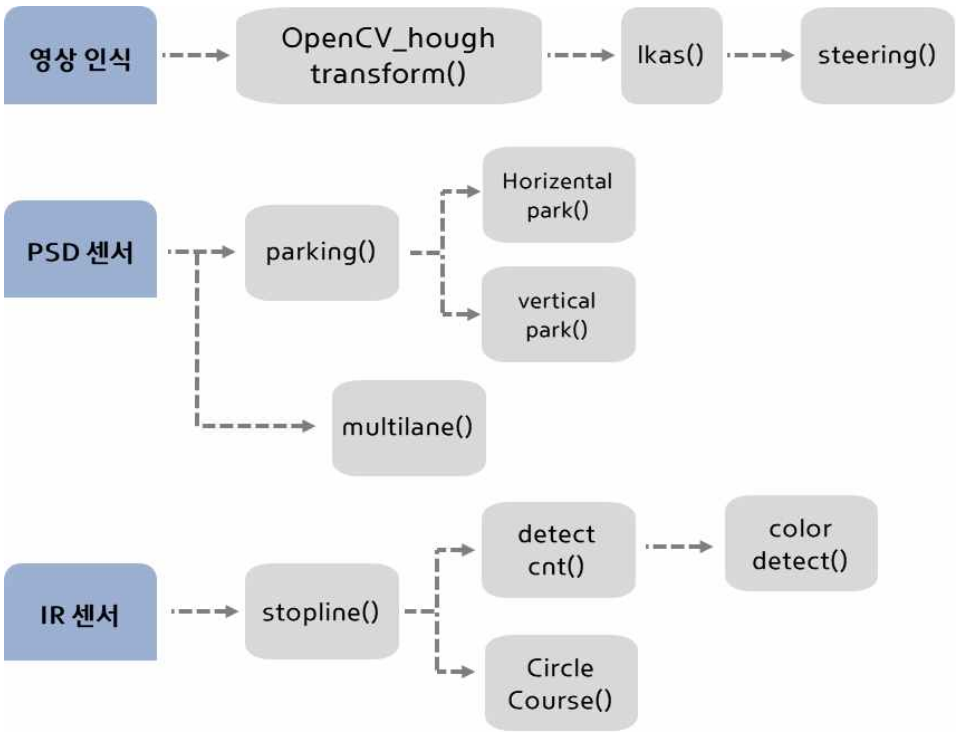


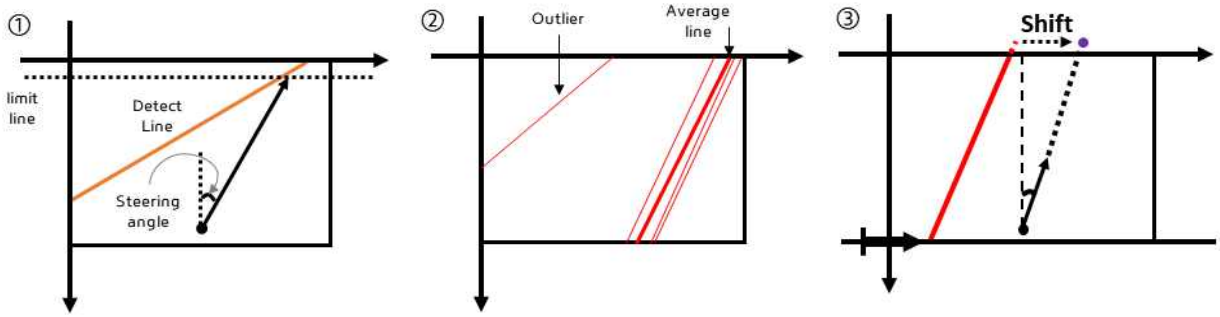
그림 20 함수 흐름도

3.4. 기술적 차별성

3.4.1 조향각 설정

3.4.2 아웃라이어 제거

3.4.3 차량 중앙 유지



- Limit line과 detect line의 교점의 좌표를 이용하여 조향 각을 설정한다.
- Steering angle에 따라 서보 모터를 동작시켜 차량이 차선 중앙에서 주행하도록 한다.

- 차선 이외의 noise가 라인으로 검출된 경우 모든 라인들의 평균을 취한 후 일정 범위 내에 있는 라인들만 올바른 라인으로 인식하고 나머지 outlier들은 제거한다.

- 차량이 차선에 붙어 있을 경우 차량이 차선 중앙으로 돌아가게 한다.
- 검출된 차선과 영상의 아래 가로축과의 교점 좌표의 위치를 고려해 조향각을 설정한다.

4. 개발 중 장애요인과 해결방안

장애요인	발생요인	해결방안
심한 굴곡변화가 있는 곡선 주행 시, 코스를 이탈하는 문제	비선형적인 조향제어방식으로 매끄러운 조향 변화 불가	검출된 차선 각도에 따른 선형적인 조향제어방식 적용
주차 미션 수행 시, 주차 벽과 충돌하는 문제	물체 반사각에 따라 multi-path 문제가 발생하여 거리 측정치 이상 값 발생	주차 알고리즘의 각 단계 수행 시, 연속된 거리 측정값들을 모아 임계값과 비교하는 방식 적용
회전 교차로 구간 탈출 시, 코스를 이탈하는 문제	수직 방향 교차로를 차선으로 인식하여 의도치 않은 조향 변화 발생	회전 교차로 구간 탈출 시, Hough Transform 직선 검출 방식의 임계값을 높여 차선인식 범위를 줄임
터널 구간 주행 시, 터널 벽과 충돌하는 문제	터널 구간의 조도 변화로 차선 인식이 불안정하여 의도치 않은 조향 변화 발생	터널 구간을 주행하는 동안, Hough Transform 직선 검출 방식의 임계값을 낮춰 차선인식 범위를 늘림
종료 지점 도착 시, 안전지대를 이탈하는 문제	안전지대 공간의 빗금을 차선으로 인식하여 의도치 않은 조향 변화 발생	신호등 미션 통과 후, Canny Edge Detection 방식의 임계값을 높여 Dominant한 직선만 추출하도록 변경

5. 개발결과물의 차별성

기존 조향제어 알고리즘에서 차선 검출 각의 구간별로 서보 조향 각을 설정하는 비선형적인 방법을 사용하였다. 이 제어 방법을 사용할 경우, 급 곡선 주행 시 차선을 이탈하거나 안정적으로 주행하지 못하는 문제를 발견하였고 이를 해결하기 위해 조향제어 방식을 2개의 Part로 나눠 설계하였다.

우선 Part를 나누기 위해 차선 검출 각 5° 간격으로 주행 안정성 테스트를 진행하였다. 테스트 결과, ±40°부터 주행 안정성이 떨어지는 것을 파악하였고 이를 기준으로 Part를 나눴다.

직진 주행 Part에서는, 비선형 조향 각 제어방식을 적용하여 차량의 직진성을 확보하였고, 곡선 주행 Part에서는, 선형 조향 각 제어방식을 적용하여 굴곡 변화에도 매끄럽게 주행할 수 있도록 하였다. 차선 검출 각 5° 간격으로 최적 서보 조향 각을 파악하였고 이를 표본 데이터로 설정하였다. 이후 Linear Regression 방식을 사용하여 표본 데이터들을 대표하는 직선 그래프를 찾아 조향제어에 적용하였다.

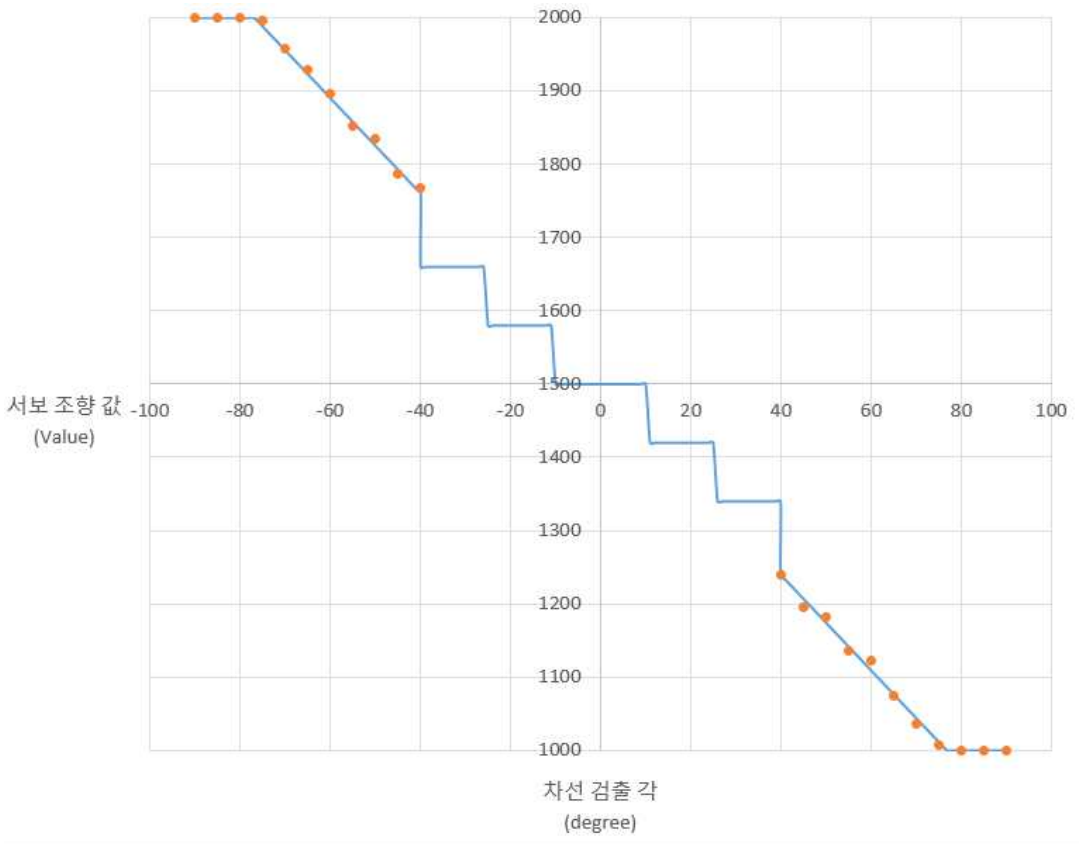


그림 24 차선 검출 각 – 서보 조향 값 그래프

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

수행 내용 \ 기간		7월				8월				9월				10월				11월			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
개발환경 구축																					
주행 코스제작																					
기본기능	API함수 분석																				
	모터·엔코더 제어																				
	서보모터 제어																				
	PSD센서 제어																				
	적외선센서 제어																				
기본주행 알고리즘	차선 인식																				
	조향 알고리즘																				
	직선 주행																				
	곡선 주행																				
	주행 안정화																				
개별 모듈	돌발상황 인식																				
	수평/수직 주차																				
	원형코스 주행																				
	터널코스 주행																				
	추월구간 주행																				
	신호등 인식																				
	엔드라인 주행																				
모듈 통합																					
종합 테스트 및 안정화																					
환경변화 대처 매뉴얼 작성																					

수행 내용		참여 인원	참여자
개발환경 구축		3	김준태, 이호영, 허민호
주행 코스제작		2	김무성, 조재욱
기본기능	API함수분석	공동	공동
	모터·엔코더 제어		
	서보모터 제어		
	PSD센서 제어		
	적외선센서 제어		
기본주행 알고리즘	차선 인식	2	김준태, 허민호
	조향 알고리즘		
	직선 주행	2	김준태, 이호영
	곡선 주행		
	주행 안정화		
개별 모듈	돌발상황 인식	2	김준태, 허민호
	수평/수직 주차	2	김무성, 조재욱
	원형코스 주행	2	김준태, 이호영
	터널코스 주행	2	김무성, 조재욱
	추월구간 주행	2	김준태, 이호영
	신호등 인식	2	김준태, 허민호
	엔드라인 주행	2	김준태, 허민호
모듈 통합		3	김준태, 이호영, 허민호
종합 테스트 및 안정화		공동	공동
환경변화 대처 매뉴얼 작성		공동	공동