제18회 임베디드SW경진대회 개발계획서

[자율주행 모형자동차]

□ 개발 개요

○ 요약 설명

팀 명	MECAR
목 표	영상처리를 이용하여 10가지 미션을 수행하는 자율 주행 자동차 시스템 구현하는 것이다. 센서 기술을 활용하여 자율 주행의 신뢰성 높이며 안정적인 시스템을 구축하는 것이다.
개발 내용 (요약)	카메라를 통한 영상 인식으로 차선의 변경, 장애물, 정지 신호등의 외부 상황을 인지하고, 이에 대처할 수 있는 모형차이다. 모형차는 10가지 미션과 관련된 실제 도로 상황에서 발생 할 수 있는 문제들을 정확히 판단하고 대처한다.
기술동영상	https://youtu.be/Lc6EqFeyEWw

개발 목적 및 목표

○ 영상처리를 이용한 자율주행 자동차 시스템 구현

- 영상 정보를 이용한 처리를 위주로 총 10가지 미션에 대한 신속한 판단과 각 상황에 따른 신속한 대응이 이뤄지는 자율주행 자동차 시스템을 구현한다.

○ 센서 기술을 활용한 신뢰성 높이기

- 영상 처리를 이용한 시스템과 더불어 적외선 거리 센서를 이용해 차량 근방의 물체와 신호를 인식한다. 이로 차량의 안정성을 확보하며 주행 제어하는 역할에 도움을 준다.
- 차량 주행의 신뢰성과 정확성을 한층 높여 안정적인 시스템 구축에 도움을 준다

□ 개발 방향 및 전략

○ 개발환경 구축

- opencv설치, 파이썬 설치, python모듈설치, 1280*720 사양의 카메라

○ 개발 방법

- 출발 및 도로주행 : 출발은 손을 인식하여 손바닥에서 주먹으로 바뀔 경우에 출발을 하도록 만든다. 기술동영상의 hand와 fist 인식 값을 이용하여 START를 인식 및 구동 하도록 만든다. 차선을 정확하게 인식하기 위해서 BGR에서 HSV로의 변환을 이용한다(밝기 요소 해결). 차선(노란)색의 HSV값을 범위를 지정하고, 해당 범위 값에 속하는 값이 아닌 나머지 부분은 검은색으로 채워 차선 이외의 노이즈를 없앤다. 이후 Canny검출기를 이용해서 이미지의 가장자리를 검출한다. 허프라인(Hough Line)을 통해서 표준 허프라인과 확률적 허프라인 검출을 구하며 차선을 검출한다. 확률적 허프라인(HoughLinesP)에서 나온 좌표값을 바탕으로 좌우 중심에 생기는 오차 값으로 모터에 출력 값을 조절해 조향한다.
- 고가도로 구간과 돌발 내리막 구간 : 고가도로를 들어가게되는 조건은 좌, 우측의 센서에 고가도로의 벽면이 인식되고, 노란색 차선을 인식을 하게 되면 고가도로 구간임을 인지한다. 고가도로임의 시작을 인지하면 모터의 출력을 높여 오르막길을 올라가도록 한다. 고가도로 위 평지는 일반 도로 주행과 동일하다. 고가도로 주행 중 노란 선이 인식되지 않으면 내리막 구간으로 판단하여 일정구간 동안 속도를 줄여 안전히 전진한다.
- 곡선(S)자 주행 코스 : 곡선 주행의 경우에는 일반주행 방법으로 주행한다. 만약 모니터에 보이는 선이 좌우 중 한 선만 인식이 될 경우, 차선의 좌표값 이용해 차선 기준으로 차선과 모형차의 거리를 (x, y축을 기준으로) 판단하여 조향 할 수 있도록 한다.
- 우선정지 장애물 구간 (위치 랜덤) : 우선 정지 신호 표지판의 빨간색만 검출하여 이진화 후, 4개의 변을 가진 도형을 인식한다. (다른 색의 사각형 모양을 정지신호로 오인식하지 않게 하기 위함)

- 수평주차, 수직주차 (위치랜덤) : 우측 좌측 중 한 방향의 거리 감지 센서만이 일정 시간 이내 장애물을 두 번 인식할 경우 주차 공간으로 인식한다. 수직 주차와 수평 주차의 구분은 주차공간 사이의 센서인식 시간이 길면 수평주차로 시간이 짧으면 수직주차로 인식한다. 주차는 주차 공간에 따라 다르게 입력된 데이터 값에 따라 구동된다. 주차 값은 연습을 통한 최적화된 주차 알고리즘을 찾아 적용시킨다. 충돌 방지 기능은 후방 센서를 이용해 차량과 벽면의 거리 감지로 차량이 벽면에 부딪히기 직전에 작용시킨다. 이후에 주차할 때와 반대 방법으로 주차 공간에서 빠져나온다.
- 회전 교차로 : 회전교차로 구간에 도입하기 전 바닥 적외선 센서로 회전 교차로 진입선(흰색 실선)을 인식하면 정지한다. 그 후, 카메라로 적색, 녹색, 황색 중 모두 인식이 되지 않는다면 회전 교차로에 진입 한 것임을 판단한다. 차량 전방의 적외선 거리 센서를 이용하여 전방의 장애물 여부를 판단한다. 전방의 장애물이 한차례 감지 될 때까지 대기한다. 전방의 장애물 감지 후, 장애물이 더 이상 인지되지 않을 때, 일정시간 대기 후 직진한다. 그 후, 일반 도로 주행을 이용하여 주행한다. 이 때, 차량 전방의 적외선 거리 센서를 이용하여 전방의 자량과 충돌이 나지 않도록 속도를 조절하며 일정 거리를 유지한다. 만일 전방 거리 센서에 교차로 내의 차량이 감지가 안 되고 차선이 인식이 되지 않으면 교차로의 끝에 있음을 인식하고 일정시간 직진으로 교차로를 빠져나간 후 기본 도로 주행과 같이 차선을 따라 주행한다.
- 터널 코스 : 터널 구간은 차선이 인식이 되지 않으며 좌, 우의 거리 감지 센서 모두가 장애물을 감지하면 터널 구간임을 인식한다. 터널 진입 시 차량 양 옆의 적외선 거리센서만을 이용하여 차량과 양 벽면 간의 거리를 측정하며 터널 공간의 중앙으로 주행 할 수 있게 한다. 양 벽면의 감지가 사라졌을 시 터널 구간의 종료를 인식한다.
- 차로 추월 구간 : 노란 차선을 인식한 후, 노란 차선 내의 흰색 점선을 인식하면 차로 추월 구간에 진입했음을 인식한다. 전방(중앙)에 장애물이 없을 시 직진한다. 전방(중앙)에 장애물이 있을 시 노란 차선 내의 흰 차선을 기준으로 카메라의 영역을 세 부분으로

나눈다. 전방이 아닌 양옆의 구간에서 장애물이 만들어낸 그림자를 검출한다. 그레이스케일(GRAY)을 이용하여 그림자를 검출하고, 그림자가 없는 쪽으로 조향하여 주행한다. 조향한 방향의 반대쪽 적외선 거리 센서에 장애물이 감지되지 않을 경우 다시 중앙 차선으로 돌아온다.

- 신호등 분기점 코스 : 신호등 정지라인을 바닥 적외선 센서로 인식하면 신호등 분기점에 도달했음을 인식하고 정지한다. 신호등에 출력되는 색(적색, 녹색, 황색)중 하나를 인식을 하면 신호등 구간임을 우선 인식하고 판단하여 회전교차로와 구별한다. 신호등 색은 차선검출과 같이 각각의 색(적색, 녹색, 황색) HSV의 범위 지정을 한 후 카메라 화면에서 범위안의 색이 인식되면 색깔에 맞는 프로그램을 적용한다. 초록 화살표는 각 점의 꼭짓점 개수로 인식한다. 이진영상으로 변환 후 컨투어(윤곽)을 찾는다. 검출된 컨투어를 근사화 시켜 직선의 개수 최소화한다. 화살표는 직선의 개수가 7개이므로 출력값으로 7이 나올 경우, 화살표로 인식하여 우회전을 한다.

○ 참고 자료 및 기술적 요구사항

1) https://docs.opencv.org/3.4.0/d9/db0/tutorial_hough_lines.html : 체스판 사진을 기반으로 허프라인을 사용하여 선을 검출한다. 기본적인 파이썬 Opencv 허프라인인 확률적 허프란인 검출과 표준허프라인 검출하는 방법.

2) https://bradbury.tistory.com/64 : HSV로 색을 검출방법

3) https://myyac.tistory.com/133(HSV색검출표 : HSV색 검출을 돕기 위한 HSV 색상표

4) https://poorman.tistory.com/193(도형검출) : 도형을 검출하기 위한 방법

5) https://076923.github.io/posts/Python-opencv-10/ : 2진 코드로 출력을 하기 위해 그레이스케일 이용 방법

6) https://webnautes.tistory.com/1378 : OpenCV를 사용하여 손 검출 및 인식하는 방법 (Hand Detection and Recognition using OpenCV)

○ 예상되는 장애요인 및 해결방안

- 경기장의 밝기에 따른 색 인식의 변화

: 경기장이 너무 밝거나 너무 어두울 경우, 카메라에서 인식하는 색이 설정했던 색의 범위를 벗어나게 되어 전체적인 색 인식에 오류가 있을 수 있다. -> 다양한 밝기의 환경에서 실험하며 최대한 정밀하게 색 검출하면서 밝기에 따른 인식의 범위가 넓어지도록 최적화한다.

- S자 또는 곡선 코스에서 오인식 가능한 외부 차선

: 곡선 주행 시 현재 및 다음의 차선 모두 인식되어 차선 검출 결과에 따른 하드웨어 구동에서 오작동이 있을 수 있다. -> 이 경우 카메라 화면을 기준으로 현재 차선만 인식되도록 ROI를 사용하여 영역을 최적화 시킨다. 카메라 각도를 더욱 아래로 기울여 화면이 현재의 도로 중심으로 보이도록 조정한다. 가장 깔끔하게 검출되는 선을 기준으로 조향을 정한다.

- 신호등 색 검출 시 환경의 영향

: 신호등 색 검출 시 카메라 화면의 시야가 넓어 주변 환경의 영향을 받아 인식의 오류가 있을 수 있다. -> 신호등 색 검출 시, 기존의 색 검출만을 사용했던 것과 달리 Hough 원 검출을 통하여 원 안의 색만 검출하도록 한다. 또는 신호등 정지라인을 인식한 후에, 신호등의 영역만 ROI영역 검출을 통해 잘라 영역 내의 색만 인식하도록 한다.

- 고가도로 내리막길 인식 불가

: 내리막길 시 카메라 화면 상 넓은 구간의 차선이 보이기 때문에 내리막길의 특성을 인식하지 못 할 수 있다. -> ROI 영역 검출을 이용해 내리막길의 차선 부분만 보이게 하여 시야를 제한한다.

○ 예상 결과 작품이 활용될 분야 및 방법 제시

- 실제 자동차에 적용시킬 수 있는 자율주행 시스템 시뮬레이션

: 모형 자동차의 차선인식, 신호등인식, 장애물 인식 등으로 실제 도로 주행의 시뮬레이션을 보일 수 있다. 이 시뮬레이션을 통해 실제 차세대 스마트카인 자율주행 자동차 시스템을 실험하는데 활용할 수 있다.

□ 영상처리 및 센서 기술 활용 방안 및 공부 내용

○ 영상처리 기술 활용 방안 및 공부 내용

- 허프라인 검출 및 특정 색 검출

: 검출 방법

1) 검출 할 화면 불러 오기(실시간 영상, 사진)

cap = cv2.VideoCapture(1)

cap = cv2.VideoCapture("실시간영상 주소")

2) 영상 전반적인 화면을 BGR값에서 HSV값으로 변경

hsv = cv2.cvtColor (src, cv2.COLOR_BGR2HSV)

3) HSV 값 범위 지정으로 원하는 색만 검출되도록 지정

lower_yellow = np.array ([20, 100, 180]) //노란색의 HSV 범위 지정

upper_yellow = np.array ([30, 255, 255])

mask_yellow = cv2.inRange (hsv, lower_yellow, upper_yellow) //노란색만 검출하고 나머지색은 검은색으로 채우는 함수

4) 전체 영상을 Canny검출기 사용하여 이미지 가장자리 검출

dst = cv.Canny (src, 50, 200, None , 3)

5) 표준 허프라인 변환

lines = cv.HoughLines(dst, 1, np.pi / 180, 150, None, 0, 0)

lines : 감지 된 선의 매개 변수 (r, θ)를 저장할 벡터

rho : 매개 변수 r의 픽셀. 여기서는 1 픽셀을 사용

theta : 매개 변수의 해상도 라디안으로. 우리가 사용하는 1 명 정도 (CV_PI / 180)

threshold : 라인을 감지하기 위한 최소 교차 수

srn and stn : 기본 매개 변수는 0

6) 확률적 허프라인 변환

linesP = cv.HoughLinesP(dst, 1, np.pi / 180, 15, None, 8, 2)

//한 번 더 필터링 함으로 신뢰성 높은 값을 구해냄

앞부분은 표준 허프라인과 동일

minLinLength : 선을 형성 할 수있는 최소 포인트 수, 이 개수보다 적은 선은 무시

maxLineGap : 동일한 선에서 고려할 두 점 사이의 최대 간격

7) 이미지 검출 및 결과 확인 : 표준 허프라인, 확률적 허프라인, 실시간 화면 출력, HSV를 이용한 차선 검출 화면 출력

- 도형 검출(꼭짓점 검출, 컨투어(윤곽선) 검출)

1) 검출 할 화면 불러 오기(실시간 영상, 사진)

2) 컬러로 불러온 영상을 2진 그레이 영상으로 변환

cv.cvtColor(img_color, cv.COLOR_BGR2GRAY) //영상을 그레이영상으로 변환

3) 임계점 127을 기준으로 2진영상으로 변환

cv.threshold(img_gray, 127, 255, cv.THRESH_BINARY|cv.THRESH_OTSU)

4) 이진영상에서 컨투어(윤곽)를 찾는 함수 findContours 사용

cv.findContours(img_binary, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

//윤곽을 찾는 함수로 도형에 꼭짓점을 찾아서 화살표를 검출한다.

5) 검출된 컨투어를 approxPolyDP 함수로 근사화 시켜 직선의 개수 최소화

size = len(cnt)

epsilon = 0.005 * cv.arcLength(cnt, True)

approx = cv.approxPolyDP(cnt, epsilon, True)

size = len(approx)

6) 검출된 직선의 개수에 따라 도형 구분

//예) 직선 3개 = 삼각형, 직선 4개 = 사각형, 직선 7개 = 화살표, 관찰된 직선 값의 범위를 지정해 펼친 손과 주먹 구별 가능

- 그 외 이미지 변경

cv2.resize(src, (480, 270)) //변수 src에 저장된 이미지(영상)의 픽셀을 (가로, 세로)값으로 재지정

cv2.imshow('src', src) //변수 src에 저장된 이미지(영상)를 출력

○ 센서 기술 활용 방안 및 공부 내용

- 라인트레이서 제작

: 적외선 센서로 빛 반사에 따른 값을 받아 검은 선을 따라가는 라인트레이서 제작

- 노크 박자 암호 감지를 하는 개방 장치

: 피에조 센서를 통해 소리의 크기에 따른 값을 받아 노크임을 판단하여 박자를 인식하는 문 개방 장치

- 자이로 가속도 센서(mpu6050)를 이용한 밸런싱 로봇 제작

: 자이로 가속도 센서를 이용해 x, y, z 좌표의 변화를 감지해 설정각도로 로봇을 유지하는 밸런싱 로봇 제작

- 자이로 센서를 통해 상황인식 헬멧 제작

: 자이로 센서의 값 변화를 통해 넘어지는 등의 위급한 상황 파악 후 경고해주는 헬멧 제작

- 아두이노와 초음파 센서를 통한 거리 값 받기

: 초음파 센서를 이용한 아날로그 값으로 센서와 다른 물체 간의 거리 인식을 실험

- 제빙기 제작

: 적외선 센서를 통해 제빙기의 얼음통의 얼음양 측정, 모듈이 아닌 센서부를 직접 제작함으로써 해당 환경에 대해 신뢰성 있는 센서부 개발

□ 개발 일정

No	내용	2020年															
		6月				7月				8月				9月			
1 (개발 환경 구축)	개발 환경 구축																
	경기장 제작																
2 (영상 처리)	차선 검출 및 주행																
	내리막길 진입인식																
	우선 정지 장애물 인식																
	곡선(S자) 주행																
	회전 교차로 진입 인식																
	추월구간 진입 인식																
	신호등 및 화살표 인식																
3 (센서 및 제어)	수평 주차 수직 주차																
	회전교차로시 차량 인식																
	터널 주행																
4 (실험 및 완성)	1차 실험																
	보완																
	최종 테스트 및 디버깅																

□ 팀 구성 및 역량

No	구분	성명	팀 내 담당 업무	업무 관련 역량 (개발 언어, 프로젝트 경험 등)
1	팀장	김기태	● 개발환경 구축 ● 구현 기술 문서화 ● 영상처리 및 시스템 총괄	○ 개발 언어 - python,C언어,C++언어 ○ 프로젝트 경험 - python과 tensorflow를 활용한 딥러닝

			● Hough변환 구현 ● S자(곡선) 구현 ● 오르막길, 내리막길 구현	- 모션캡쳐를 이용한 머니퓰레이터 - 라인트레이서 센서부 회로 제작 및 프로그래밍 - 라즈베리파이를 이용한 실시간 영상 RC카 와이파이를 이용한 통신 이용(앱인벤터 제작) - MPU6050을 이용한 밸런싱로봇 제작 및 프로그래밍
2	팀원	박영우	● 신호등 색깔 구현 ● 화살표 모형 구현 ● 장애물 구간 구현 ● 개발 환경 구축	○ 개발 언어 - C,C++,Python,JAVA ○ 프로젝트 경험 - 아두이노를 이용한 모바일 자동차(MFC와 어플리케이션 연동) - OpenCV를 이용한 도형검출 프로그램 제작 - OpenCV를 이용한 사람얼굴인식 프로그램 제작(CCTV)
3	팀원	이하나	● 터널 구간 구현 ● Hough변환 구현 ● 고가도로 구현 ● 오르막길 구현 ● 회전 교차로 구현 ● 구현 기술 문서화	○ 개발 언어 - C, C++ ○ 프로젝트 경험 - 라인트레이서 센서부 회로 제작 및 프로그래밍 - 통신을 이용한 라즈베리파이를 이용한 실시간 영상 RC카(앱인벤터 제작) - 박자인식기반 노크식 도어락 - MPU6050을 이용한 밸런싱로봇 제작 및 프로그래밍
4	팀원	허남규	● 수평 주차 수직 구차 구현 ● 센서 필터링 및 맵핑 ● 영상 처리에서 나오는 값을 통해 모터 구현 ● 차로 추월 구간 구현	○ 개발 언어 - C, C++, python ○ 프로젝트 경험 - 웨어러블 방식의 동작인식 기반 로봇 팔 - 음료 농도 제빙기 - 라인트레이서 센서부 회로 제작 및 프로그래밍 - 박자인식기반 노크식 도어락 - 영상처리 기반 좌표이동 볼 밸런싱