AD Project REPORT

국민대학교 KOOKMIN UNIVERSITY

과목명 | 창업연계공학설계입문

담당교수 | 이시윤 교수님

학과 | 소프트웨어학부

팀원 | 4조

김성훈(20191562)
김유진(20191567)
원정희(20165155)
이진백(20162836)

제출일 | 2019.12.20

1. 장애물 탐지 및 회피

■ 실제 자율주행 시스템

- 전자기파를 쓰는 레이더와 비슷한 라이다 센서 사용
  - 레이더가 볼 수 없는 사각지대까지 관측이 가능
  - 레이더로 탐지 불가능한 화학가스나 기상 관측도 가능
  - 물체와의 거리를 보다 정밀하게 관측 가능
- 딥러닝 기반의 이미지 인식 기술
  - 인식한 상황에 따라 차가 자체적인 판단을 내려야 하지만 특정 상황에서는 차가 판단을 하지 못하는 상황이 있어 사고가 발생하는 사례가 있다.

라이다 센서

■ 첫번째 설계

센서 오류로 인한 실패

- Xycar 박스를 장애물로 사용
- 1단계 : 장애물을 초음파로 인식
- 2단계 : 장애물이 좌측에 있으면 우회전으로 우측에 있으면 좌회전으로 일정시간

이동

- 3단계 : 다시 차선을 보고 주행
- 변수
  - Xycar가 차선 근처에 있는 다른 장애물을 인식하여 회전해 버린다.
  - 초음파 센서의 값이 불안정하다

■ 두번째 설계

- 장애물을 소화기로 변경
- 1단계 : 프레임을 빨간색, 검은색 두개로 분리
- 2단계 : 모든 픽셀을 검사하여 빨간 영역 픽셀 개수 확인
- 3단계 : 특정 개수 이상의 픽셀이 있으면 장애물이라 판단
  - 코드 문제로 탐지를 못 함

■ 마지막 설계

- 1단계 : 가우시안블러 적용
- 2단계 : HSV 적용
- 3단계 : HSV 범위값을 이용하여 이진화
- 4단계 : 인식된 물체의 픽셀 수를 파악
- 5단계 : 픽셀의 개수가 5000개 이상인지 확인
- 6단계 : 위치 파악 후 좌회전이나 우회전 선택
  - 차선이 인식되는 것을 방지하기 위해 순위를 차선보다 높게 코딩

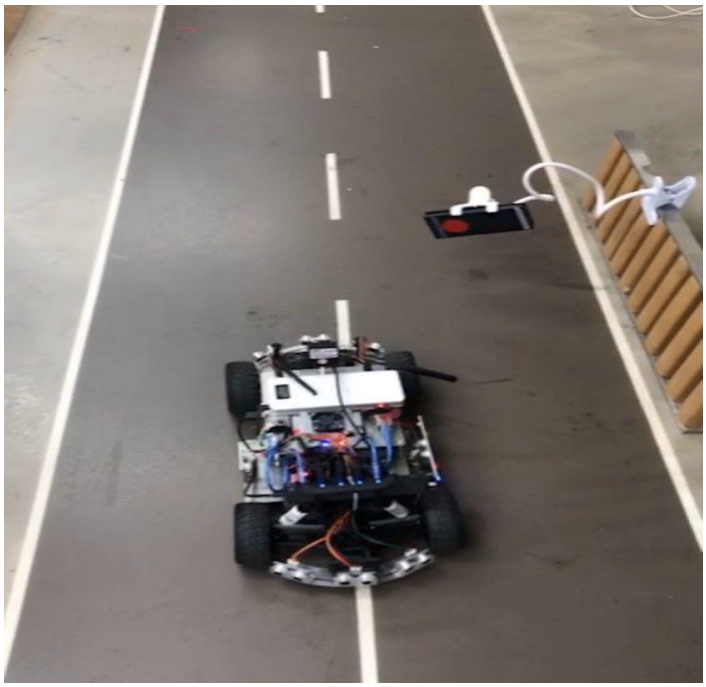
2. 신호등 탐지

■ 실제 자율주행 시스템

- 대부분은 카메라를 이용해서 인식
  - 여라 상황에서 카메라 인식이 어려우면 인식률이 떨어지는 문제 발생
  - 신호등마다 불빛의 개수도 다르고 화살표를 인식해야함
  - 도시마다 신호등의 방향이 가로 또는 세로로 다르다.
- 지도 정보에 신호등에 대한 정보를 넣어두는 방법
  - 카메라 인식으로 구현했을 때 보다 인식이 더 잘 됐다.

■ 설계

- 실제와는 다른 하나의 신호등만을 이용하여 빨간불이면 정차하는 코드 작성
- 상단부의 신호등을 잡기 위해서 카메라 각도를 위로 ROI를 아래로 내려준다.
- 1단계 : 흑백화
- 2단게 : HoughCircles 함수를 이용하여 원 탐지
- 3단계 : param2(오류 확률, 검출률) 조절
- 4단계 : 원의 반지름 선택(0~40px로 설정)
- 5단계 : 중심으로부터 픽셀을 퍼뜨려 픽셀이 200개 이상되는 곳을 감지
- 6단계 : 감지가 된다면 빨간불 신호등이라 판단하여 정차



3. 코드

```
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        center = (i[0], i[1])
        radius = i[2]

        cv2.circle(frame, center, radius, (0, 255, 0), 2)
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        pos = [center]
        visited_position = {center}
        red_pixel_count = 0
        while len(pos) > 0:
            x, y = pos.pop(0)
            h, s, v = hsv[y, x]
            for dx, dy in zip([-1, 0, 1, 0], [0, 1, 0, -1]):
                cur_x = x + dx
                cur_y = y + dy
                cur_pos = (cur_x, cur_y)
                if 0 < cur_x < 640 and 0 < cur_y < 480 and cur_pos not in visited_pc
                    visited_position.add(cur_pos)
                    if 150 <= h <= 180 and 100 <= s <= 255 and 100 <= v <= 255:
                        pos.append(cur_pos)
                        red_pixel_count += 1
                        if red_pixel_count >= 200:
                            self.left_right = False
                            self.traffic_light_detected_count = 100
                            break
```

신호등 탐지 코드

```
def detect_obstacle(self, frame):

    lower_red = np.array([150, 100, 50])

    upper_red = np.array([180, 255, 200])

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

    mask_red = cv2.inRange(hsv, lower_red, upper_red)

    if cv2.countNonZero(mask_red[0:240, :320]) > 5000:

        self.left_right = 'left'

    if cv2.countNonZero(mask_red[0:240, 320:]) > 5000:

        self.left_right = 'right'
```

장애물 탐지 및 회피 코드

4. 실행예시(영상)

github(https://github.com/KSH-code/self-driving-car/blob/master/ad-project/%EC%8B%9C%ED%98%84.mp4)

5. 소감

- 김성훈 : 창업연계공학설계입문은 학생들을 공학자의 업무를 직접적으로 체험할 수 있게 해주었고, 지나온 일들을 되짚어 보면 수정, 테스트, 수정 등 일련의 과정을 반복함으로써 보다 정교한 이동 또는 인식을 하였고 실패율을 줄일 수 있었습니다. 이러한 과정에서 상세 구현 내용을 명확하게 이해할 수는 없었지만 어림 짐작을 할 수는 있었다. 그 점이 아쉬웠

- 지만 앞으로 이러한 기반 지식들을 배울 것이니 동작 원리까지 이해할 수 있도록 노력해야겠다.

- 김유진 : 학우는 순수소프트웨어의 개발만 해보다가 임베디드 개발을 해본 것은 처음 이였고, 디버깅할 때 항상 소스의 문제점을 보다가 물리적인 차의 환경도 같이 고려해야했다. 고려할 범위가 넓어져서 디버깅하기 더 힘들었다. 그래도 차가 잘 굴러가서 뿌듯했습니다. opencv는 한 번쯤 다뤄보고 싶었는데 수업시간에 배워서 좋았습니다. 영상처리에 쓰이는 기법들을 많이 배운 것 같다라고 말했습니다.

- 원정희 : xycar라는 자율주행자동차가 비싸서 실제로 구해서 실습해보기 어려웠는데 이번기회에 open cv를 이용해서 실제 자율주행차들처럼 구현해 볼 기회가 있어서 좋았고 실제 자율주행에도 아직 문제점들이 있는데 실제 구현해보면서 많은 제한사항들이 많다는 것을 느끼면서 많은 노력이 필요하다고 생각하였습니다.

- 이진백 : 처음으로 접한 파이썬 코딩이라 처음에 수강을 포기할까 생각도 많이 했는데 다른 조원들의 코딩을 보면서 그걸 바탕으로 코딩하는 방법을 연습할 수 있는 기회가 되었고 막상 실제로 코드를 실행했을 때 변수로 인해서 코드를 많은 부분을 수정하고 다시 작성해서 원했던 코딩이 성공했을 때의 재미를 느낄 수 있었습니다