창업연계공학설계입문

AD 프로젝트 6조 보고서

과 목	창업연계공학설계입문
담 당 교 수	한재일
제 출 일	2019.12.18
조 원	20191626 오준호(조장) 20191633 윤서영(팀원) 20191639 이성민(팀원) 20191645 이윤호(팀원)

□ 목차

1. 개요

1.1 프로젝트 개요

본 프로젝트에서는 XyCar 모형차를 가지고, 여러 가지 기능을 구현한 자율 주행 프로젝트를 진행하고자 한다. 국민대학교 소프트웨어학부 '유레카 프로젝트' 교과목에서는 자율 주행 프로젝트를 진행하였다. 단순히 차선을 검출하고, 그에 따른 자율 주행을 하는 반면에, 본 프로젝트에서는 노란 정지선 감지, 표지판 인식, T자 주차 등 다양한 기능들을 탑재한 자율 주행 프로젝트를 진행하고자 하였다.

1.2 프로젝트 배경

'유레카 프로젝트' 교과목에서는 수강자들을 대상으로 한 자율주행대회를 개최하였다. 현재 개발된 시스템의 문제점은 트랙을 다 완주하고 난 후에 자동으로 멈추는 시스템이 존재하지 않는다. 또한, 한바퀴를 다 돌고 나면, 자동으로 멈추려고 했으나, 다른 차들의 주행을 방해하는 요소가 되어버렸다. 따라서 주행 후에 특정한 공간에 T자를 주차하는 시스템을 개발하고자 하였다. 더불어 교통 표지판을 인식하면 그에 맞게 느려지거나, 정지하는 등의 부가적인 기능들을 탑재하도록 하였다.

2. 프로젝트 설명

2.1 기능 소개

	기능 명칭	내용
1	정지선 인식	자율주행스튜디오 안에 존재하는 정지선을 인식하여, 자동으로 멈추는 기능을 구현.
2	표지판 인식	'천천히', '정지' 표지판을 인식한 후에 해당 상황에 맞게 천천히 가거나, 멈추는 구현.
3	T자 주차	완주 후에, T자 주차를 수행하는 기능을 구현.

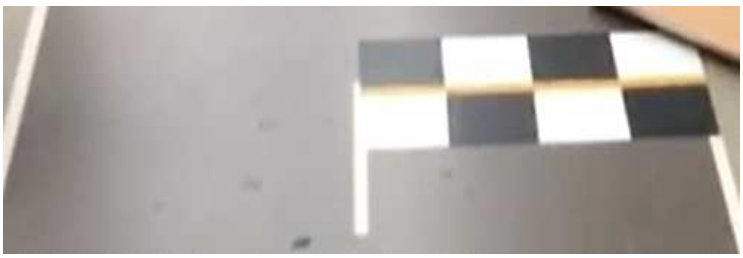
2.2 구현 과정

2.2.1 정지선 인식

국민대학교 자율주행스튜디오에는 두 가지 노란색 선이 존재한다. 더 긴 노란색 선이 정지선이며, 무조건 적으로 노란색을 정지선으로 인식하게 된다면, 중간에 멈추는 불합리가 생길 수도 있다. 따라서 < 그림 1 > 에 있는 노란색 선만 정지선으로 인식해야 한다.

< 그림 1 >

< 그림 2 >



우선 노란색 정지선을 인식하기 위해서 OpenCV RGB 방식으로 검출하도록 하였다. 노란색에 해당하는 RGB 범위를 Lower/Upper 로 정한 후에, 해당 범위에 있으면 흰색(255), 아니면 모두 검정색(0) 으로 변경하는 이진화 방식을 사용했다. 또한, < 그림 1 > 노란색 선만을 정지선으로 인식해야 하므로, cv2.countNonZero 함수를 활용하여, 그 값이 500 이상일 경우에만 정지선으로 인식하게끔 로직을 구현하였다.

< 그림 3 >

```
def detect_yellowline(frame):
    width = 640
    offset_roi = 125
    mask = frame[430 - offset_roi:450 - offset_roi, 0:width]
    lower_white = np.array([120, 200, 200], dtype=np.uint8)
    upper_white = np.array([160, 240, 255], dtype=np.uint8)

    mask = cv2.inRange(mask, lower_white, upper_white)

    if cv2.countNonZero(mask) >= 500:
        return True
    return False
```

2.2.2 표지판 인식

자율 주행 프로젝트의 부가적인 기능으로, 교통 표지판을 인식하도록 한 후에 표지판의 내용에 맞는 기능을 수행하도록 하는 로직을 추가로 개발하였다. Github에 올라와있는 "ghostbbbmt/Traffic-Sign-Detection" 라이브러리를 참조하여 구현하였으며, 해당 이미지를 모델링하여 OpenCV를 통해 해당 이미지가 검출되는 로직을 구현하도록 하였다. 본 프로젝트에서는 "천천히", "정지" 두 가지의 표지판을 인식하도록 구현하였다.

< 그림 4 > 교통 표지판

< 그림 5 > 표지판 인식 구현

```
# traffic signs
coordinate, image, sign_type, text = localization(cv_image, 2,
                                                  similitary_contour_with_circle, model, count, current_sign)
if sign_type > 0 and (not current_sign or sign_type != current_sign):
    current_sign = sign_type
    current_text = text

# 1) Slow
if current_sign != None and current_text == "SLOW":
    speed = 116

# 2) Stop
if current_sign != None and current_text == "STOP":
    speed = 90
```

2.2.3 T자 주차

처음에는 평행 주차 기능을 구현하려고 하였다. 하지만 예상보다 조향각 설정이 쉽지 않았다는 사실을 깨닫게 된 후에는, T자 주차를 하는 것으로 계획을 변경하였다. 우선 현실 세계에서 T자 주차를 어떻게 행하고 있는 지에 대해 분석하였다. 순서는 아래와 같다.

1) 주차장의 반 정도까지 직진을 수행한다.

2) 조향각을 60도로 설정한 후에, 자회전 직진을 수행한다.

3) 조향각을 140도로 설정한 후에 우회전으로 후진을 수행한다.

4) 주차장과 XyCar가 평행하다면, 조향각 90도에 후진을 수행하고 종료한다.

위와 같은 순서로, < 그림 6 > 과 같이 XyCar에 해당 기능을 구현하도록 하였다.

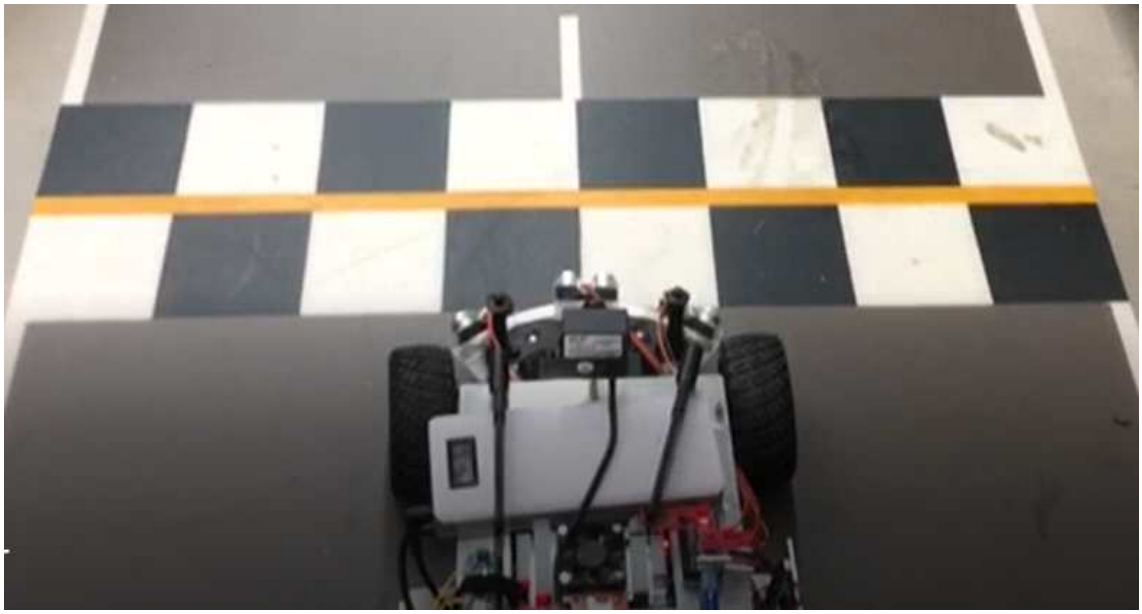
< 그림 6 >

```
print("parking")
for i in range(2):
    auto_drive(90, 90)
    time.sleep(0.1)

for i in range(12):
    auto_drive(90, 120)
    time.sleep(0.1)

for i in range(22):
    auto_drive(60, 125)
    time.sleep(0.1)

for i in range(2):
    auto_drive(90, 90)
    time.sleep(0.1)

for i in range(28):
    auto_drive(145, 63)
    time.sleep(0.1)

for i in range(10):
    auto_drive(90, 75)
    time.sleep(0.05)
```

2.3 테스트 확인

정지선 인식, 표지판 인식, T자 주차 총 3가지 기능을 구현하였으며, 자율주행스튜디오 라인을 기준으로 테스트를 진행하였다.

< 그림 7 >



< 그림 8 >

< 그림 9 >

< 그림 10 >

3. 프로젝트 수행 후기

본 프로젝트를 진행하면서 힘든 경험도 분명 있었지만, 평소 별로 중요하게 생각하지 않았던 수학 및 물리 개념을 더 깊게 파고들 수 있는 기회가 된 것 같다.

Github에 올라와있는 표지판 인식 3rd Party 라이브러리를 활용하면서, OpenCV 라이브러리를 좀 더 깊게 다루어볼 수 있는 시간이 되었고, ROS 기반 시스템을 잘 이해할 수 있는 시간이 되었다. 특히, XyCar에 존재하는 모든 센서를 다뤄볼 수 있었다는 사실이 인상 깊은 시간이었던 것 같다.

또한, 이 프로젝트를 진행하면서 알게 된 사실은 다양한 케이스에 테스트를 진행해보아야 한다는 것이다. 임베디드 시스템은 환경에 의해 많은 변화가 있다는 사실을 깨닫게 되었다.