기본적인 교통법규를 준수하는 자율주행 자동차 구현

과목명 창업연계공학설계입문

담당교수 한광수 교수님

학과명 소프트웨어학부

학번, 이름 20141733 조주현

20191598 박정명

20191604 백연선

20191610 성정규

20191617 신지효

서론

1. 주제 선정 동기

<그림 1>

최근 교통사고 발생 원인은 <그림 1>과 같이 가장 기본적인 교통법규인 신호위반이 제일 크다. 선행 연구에서 신호위반 행위 및 신호위반사고 발생에 영향을 미치는 것으로 제시되는 요인은 주로 다음과 같다.

1) 도로교통환경

(1) 차량 속도

고속 주행차량의 운전자는 저속 주행차량의 운전자와 동일한 시점에 교차로에 도착했을 경우, 적색신호를 무시하고 그대로 교차로에 진입할 확률이 높다.

(2) 정지선까지의 이동시간

일반적으로 정지선까지의 도달시간이 길어질수록 정지 가능성이 높아진다. 그러나 이는 신호제어 형태에 따라 선형적이지 않게 결과가 나타난다. 교차로의 황색시간 길이를 적절하게 결정하면 접근차량이 교차로 진입 전에 신호지시에 따라 정지할 확률이 높다.

(3) 신호제어 유형

감응식 신호제어는 고정식 제어 방식에 비해 상대적으로 차량군을 더욱 밀집화시켜 신호위반을 감소시키는데 도움을 줄 수 있다.(Van der Horst, 1998). 녹색신호 연장 제어기법은 녹색신호 종료 전 더 많은 차량을 통과시킴으로 차량의 불필요한 정지 가능성을 최소화하여 운전자의 신호위반을 감소시킬 확률이 높다.

(4) 신호등 시인성

교차로 부근에 주차된 차량, 교차로의 부적절한 구조, 신호등의 물리적 특성(밝기, 크기, 위치 등), 가로수 등을 통해 운전자의 시인성을 제한할 수 있다.

2) 운전자

고령 운전자의 경우에는 시력이 운전 능력에 영향을 미칠 수 있고, 운전자의 인지력 또는 주의력과 관련이 있으며 산만함, 부주의 곁눈질, 졸음 등의 요소가 포함된다. 또한 운전자의 인지 및 반응 시간, 주변 운전자의 영향으로 인해 신호위반을 할 확률이 있다.

자율주행 자동차(XyCar)에서 미리 현재 신호의 정보를 받아 주행 상태를 결정하면 신호위반의 요인 중 운전자의 부주의를 통한 사고를 막을 수 있다고 생각한다.

2. 구현할 내용 및 방안

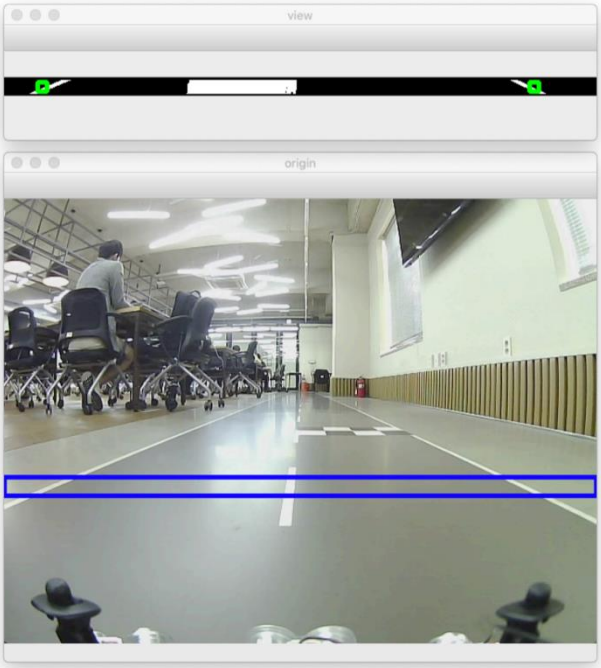
Xycar 의 OpenCV 를 통해 차선을 감지하여 주행을 하다가 황색불을 탐지할 경우 서서히 속도를 늦추며 적색불을 탐지하면 정지한다. 그리고 청색불이 탐지되면 다시 주행을 시작한다. 주행을 하는 도중 장애물을 탐지할 경우 정지를 하고 10 초간 기다린다. 만약 장애물이 10 초 이내에 이동을 하여 탐지가 더 이상 안될 경우에는 다시 주행을 시작하고, 시간이 10 초 이상 지난 후에도 장애물이 계속 탐지될 경우에는 주행을 시작하며 차선을 바꾼다.

Xycar 의 OpenCV, 모터, 초음파 센서, IMU 센서를 활용하여 주행을 하고 신호등 같은 경우 우드락으로 기본 뼈대를 만든 후 아두이노를 활용하여 신호등의 색을 표시할 것이다. 주행 중 만날 장애물은 우드락을 사용하여 직접 만들고 손으로 이동시킬 것이다.

본론

1. 차선 인식 및 신호등 감지

먼저, 신호등 감지 및 차선인식 에 대해서 이해하기 위해서는 '창업연계공학설계입문' 시간에 배웠던 OpenCV 에 대한 충분한 이해가 필요하다. 필자 또한 신호등 감지를 진행하기 위해 기존에 수업시간에 배웠던 OpenCV 에 대한 내용을 복습하였다.

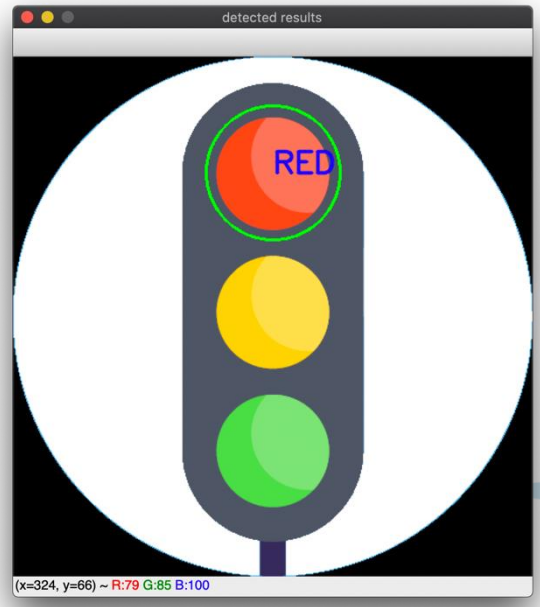


<그림 1> 차선 인식을 위해 사용하는 OpenCV

기존에 우리는 OpenCV 를 이용하여 <그림 1>과 같이 차선을 인식하는 방법에 대하여 배웠다. ROI(Region of Interest)를 통하여 필요한 부분을 취득하고 HSV 를 통한 GrayScale 이진화 및 가우시안 블러를 통한 노이즈 제거, Canny 를 통한 Edge 획득을 통해 얻은 이미지에 범위를 지정하여 그 범위가 하한 값 (pixel threshold) 을 넘어갈 경우 차선으로 인식하는 알고리즘이 기존 수업시간에 배웠던 차선 인식 방법이다. 우리 조는 이 방법을 대체하여 기존 방법에서 가우시안 블러를 통해 노이즈를 제거하는 부분까지는 동일하지만 소벨 마스크 (Sobel Mask) 를 통한 엣지 검출 및 허프 변환 (Hough Transform) 방식을 새롭게 채택하여 차선 인식 알고리즘에 적용하였다. 이 방식에 대해 간략하게 소개하자면 소벨 마스크는 영상에서 윤곽선을 검출하는데 자주 쓰이는 마스크이며, 모든 방향의 윤곽선, 즉 엣지를 추출 하는 방법 중 하나이다. 엣지를 추출 할 때, 수직마스크와 수평마스크를 적용하여 엣지를 추출하는 과정을 거친다. 이렇게 추출한 엣지의 기울기를 모아서 새로운 직선을 그려주는 방법이 허프 변환 방법이다. 이렇게 두 가지를 이용했을 때 기존의 Canny 를 통한 엣지 검출에 비해 정확도가 높아져서 사용하게 되었다.

이렇게 차선 검출에서 이용한 소벨 마스크와 허프 변환 모두 cv2 모듈에서 제공하고 있는 방법을 이용하여 사용했기 때문에 큰 어려움 없이 사용할 수 있었다. 코드에서

linedetector.py 에 존재하는 LineDetector 의 detect_lines 메소드에서 이 알고리즘이 담긴 Python 코드를 확인할 수 있다.

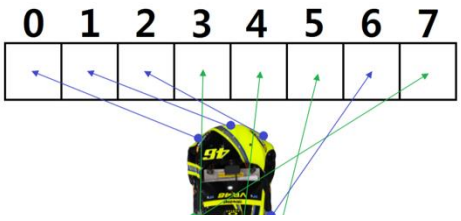


<그림 2> TrafficLight-Detector 을 이용한 모습. 아직 수정 전이라 빨간색 외의 나머지 색을 인식하지 못한다.

신호등 감지는 GitHub HevLfreis 레파지토리에 공개 되어있는 TrafficLight-Detector 을 이용하였다. <그림 2>는 코드 그대로를 작동했을 때 나타나는 화면이다. 이런식으로 빨간색, 노란색, 초록색을 인식하여 화면에 띄어주는 코드로 작성되어 있었다. 이 부분을 수정하여 신호등에서 빨간색과 노란색, 초록색을 이용할 수 있도록 변경해 주었고 boolean 형을 이용해 return 해 주어서 빨간색 불이 감지되었을 경우에는 red is True, 노란색 불이 감지되었을 경우에는 yellow is True, 초록색불이 감지되었을 경우에는 green is True 로 인식할 수 있게 하였다. 그리하여 Red 가 True 일 경우에는 어떤 경우라도 speed 가 90 이 될 수 있게 하였고, Yellow 가 True 일 때는 속도를 125, Green 이 True 일때는 속도를 140 으로 움직일 수 있도록 설정하였다. 아무것도 감지되지 않았을 경우에는 초음파 센서를 이용하여 감지한 장애물의 유무에 따라서 움직일 수 있도록 하였다. 그 부분에 대해서는 다음 항목인 '초음파 센서 이용'에서 다루도록 하자.

이 두가지 모두 OpenCV 를 통한 Detect 부분은 LineDetector 에서 확인할 수 있고 이 return 값에 맞게 drive 해주는 것은 MainDriver 에서 처리해주었다.

2. 초음파 센서 사용



<그림 3> 초음파 센서의 위치와 데이터 값

초음파 센서는 전체적으로 8 개의 초음파 센서를 가지고 있다. 이번 주제에서 사용할 초음파 센서는 총 3 개로 Front 부분의 left, mid, right 이다. 이 세가지 부분을 이용하여 (<그림 3>에 따르면 data 에서는 0, 1, 2

index) 들어온 데이터 값이 51 이하일 경우, 즉 장애물과의 거리가 51cm 이하일 경우에 속도를 90 으로 설정하여 멈추게 한다. 여기서 51 로 설정한 이유는 코드로 51 로 작성했을 때와 실제에서의 51cm 로 감지되는 것이 다르기 때문이다. 실제 달리는 기본 속도인 130 에서 51cm 를 감지하라고 코드를 작성하면 실험결과 약 15cm 정도 되었을 때 멈추도록 작동한다. 이를 이용하여 51 이라는 숫자로 코드에 작성한 것이고 실제로 멈추는 정도는 약 15cm 이다. 또한, 이렇게 return 받은 초음파의 값이 잘못들어왔을 경우를 대비하여 filter 를 이용해 안정적으로 이용할 수 있도록 하였다. 초음파 센서의 값을 받아오고 처리하여 return 해주는 곳은 ObstacleDetetor 이고 이를 MainDriver 의 accerlate 에서 이용하게 된다. 위에 신호등에서도 언급되었는데 신호등에서 아무것도 감지되지 않았을 경우, 초음파 센서 값의 의존하여 accerlate 하게 된다.

1) 장애물이 10 초 이내에 움직일 경우

위 방식에 따라 장애물을 감지하여서 차선 변경에 적용하거나 잠시 멈추는 것에 적용하게 되는데 장애물이 10 초 이내에 움직일 경우는 잠시 멈추는 경우에 해당된다. 위에서 언급한 것처럼 초음파 센서의 Front left, mid, right 의 값이 51 이하일 경우 time 모듈을 이용하여 시간을 측정하게 된다. 이렇게 시간을 측정하게 되어서 약 10 초 이내에 이 초음파 값이 다시 51 보다 큰 숫자가 된다면 다시 정상적으로 움직이게 된다.

2) 10 초가 지난 후 장애물 탐지가 계속될 경우

위 방식에 따라 장애물을 감지하여서 차선 변경에 적용하거나 잠시 멈추는 것에 적용하게 되는데 장애물 탐지가 10 초동안 계속 지속되는 경우네는 차선 변경에 적용하게 된다. 만약 기존 차선 (오른쪽) 에서 장애물 탐지가 약 10 초동안 계속된다면 왼쪽으로 조향각을 조정하게 하고 (50 도) 직진하게 하여 다시 차선인식 진행 및 움직이도록 설계하였다.

결론

1. 구현 결과

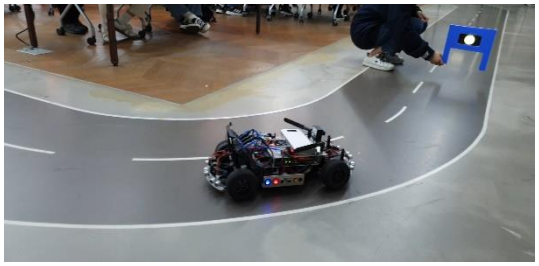
1) 신호등 코드 및 주행 사진

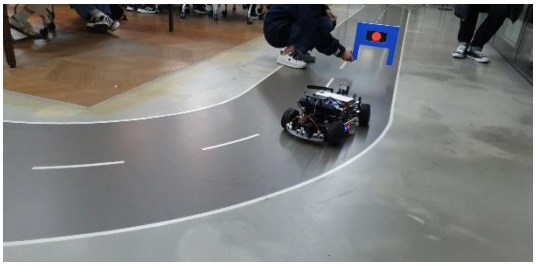
```python
def detect_lights(self):

    font = cv2.FONT_HERSHEY_SIMPLEX
    img = self.cam_img
    cimg = img
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # color range
    lower_red1 = np.array([0, 100, 100])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([160, 100, 100])
    upper_red2 = np.array([180, 255, 255])
    lower_green = np.array([40, 50, 50])
    upper_green = np.array([90, 255, 255])
    lower_yellow = np.array([15, 150, 150])
    upper_yellow = np.array([35, 255, 255])
    mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
    maskg = cv2.inRange(hsv, lower_green, upper_green)
    masky = cv2.inRange(hsv, lower_yellow, upper_yellow)
    maskr = cv2.add(mask1, mask2)

    size = img.shape
    # print size

    # hough circle detect
    r_circles = cv2.HoughCircles(maskr, cv2.HOUGH_GRADIENT, 1, 80,
                                 param1=50, param2=10, minRadius=0, maxRadius=100)

    g_circles = cv2.HoughCircles(maskg, cv2.HOUGH_GRADIENT, 1, 60,
                                 param1=50, param2=10, minRadius=0, maxRadius=100)

    y_circles = cv2.HoughCircles(masky, cv2.HOUGH_GRADIENT, 1, 30,
                                 param1=50, param2=5, minRadius=0, maxRadius=100)

    red, green, yellow = False

r = 5
bound = 4.0 / 10
if r_circles is not None:
    r_circles = np.uint16(np.around(r_circles))

    for i in r_circles[0, :]:
        if i[0] > size[1] or i[1] > size[0] or i[1] > size[0] * bound:
            continue

        h, s = 0.0, 0.0
        for m in range(-r, r):
            for n in range(-r, r):

                if (i[1] + m) >= size[0] or (i[0] + n) >= size[1]:
                    continue
                h += maskr[i[1] + m, i[0] + n]
                s += 1
        if h / s > 50:
            red = True
```

```python
if g_circles is not None:
    g_circles = np.uint16(np.around(g_circles))

    for i in g_circles[0, :]:
        if i[0] > size[1] or i[1] > size[0] or i[1] > size[0] * bound:
            continue

        h, s = 0.0, 0.0
        for m in range(-r, r):
            for n in range(-r, r):

                if (i[1] + m) >= size[0] or (i[0] + n) >= size[1]:
                    continue
                h += maskg[i[1] + m, i[0] + n]
                s += 1
        if h / s > 100:
            green = True

if y_circles is not None:
    y_circles = np.uint16(np.around(y_circles))

    for i in y_circles[0, :]:
        if i[0] > size[1] or i[1] > size[0] or i[1] > size[0] * bound:
            continue

        h, s = 0.0, 0.0
        for m in range(-r, r):
            for n in range(-r, r):

                if (i[1] + m) >= size[0] or (i[0] + n) >= size[1]:
                    continue
                h += masky[i[1] + m, i[0] + n]
                s += 1
        if h / s > 50:
            yellow = True

return red, green, yellow

def accelerate(self, angle, mid, red, green, yellow):
    if angle < -50 or angle > 50:
        speed = 40
    else:
        speed = 40

    self.obs_filter.add_sample(mid)
    if len(self.obs_filter.data) >= 3:
        if self.obs_filter.get_wmm() <= 50:
            speed = 0

    if red:
        speed = 0
    elif yellow:
        speed = 25
    elif green:
        speed = 40

    return speed
```

(1) 빨간색, 초록색, 노란색의 범위를 정한 뒤 색깔별로 원을 검출하도록 하기

(2) 색깔별로 원을 감지한 후 빨간색일 경우 정지, 노란색일 경우 감속, 초록색일 경우 정상 주행하기





2) 장애물 코드 및 주행 사진

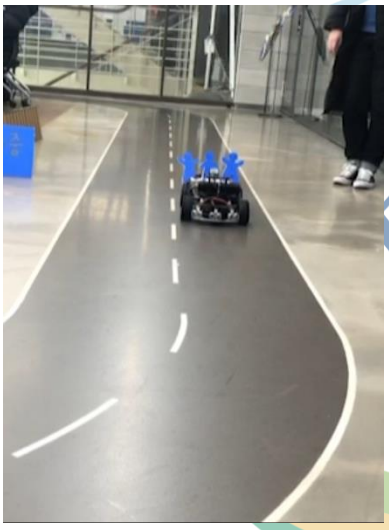
```python
class ObstacleDetector:

    def __init__(self, topic):
        self.left = -1
        self.mid = -1
        self.right = -1
        rospy.Subscriber(topic, Int32MultiArray, self.read_distance)

    def read_distance(self, data):
        self.left = data.data[0]
        self.mid = data.data[1]
        self.right = data.data[2]

    def get_distance(self):
        return self.left, self.mid, self.right
```

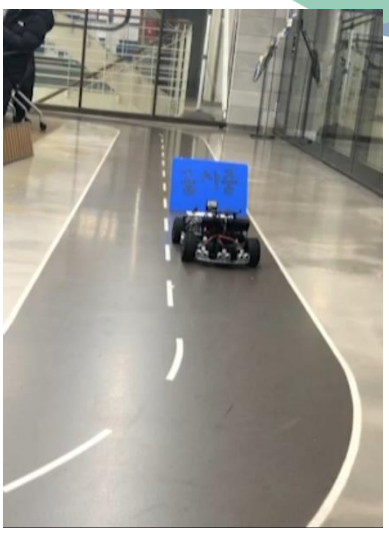
```python
def trace(self):
    line_l, line_r = self.line_detector.detect_lines()
    left, mid, right = self.obstacle_detector.get_distance()
    red, green, yellow = self.line_detector.detect_lights()
    self.line_detector.show_images(line_l, line_r)
    angle = self.steer(line_l, line_r)
    speed = self.accelerate(angle, mid, red, green, yellow)

    if speed == 0:
        if not self.stop_time == None && time.time() - self.stop_time >= 10:
            angle = 45
            speed = 40
        else:
            self.stop_time = time.time()
    else:
        self.stop_time = None

    if angle != None:
        self.driver.drive(angle + 90, speed + 90)
```

(1) 초음파 센서를 이용하여 장애물을 감지하기

(2) 장애물 탐지 시 정지하기

(3) 장애물이 10 초 이상 탐지될 경우 우회하여 주행하기





2. 느낀 점

창업연계공학설계입문 AD 프로젝트를 통하여 최근 대두되고 있는 자율주행자동차의 소프트웨어를 직접 개발해볼 수 있는 유익한 시간을 가질 수 있었다. 시간과 장소의 제약때문에 많은 아쉬움이 있었지만 하드웨어도 정상적이었고, 무엇보다 성실한 팀원들과 꾸준히 작업해온 끝에 만족스러운 결과를 얻을 수 있었다. (20141733 조주현)

이번에 창업연계공학설계입문 AD 프로젝트를 진행하면서 아쉬운 점도 많았지만 그로써 배우는 것들이 많았다. 아쉬운 점은 먼저 장소가 너무 한정적인 곳에서 여러 명이 테스트하기 위해 트랙에 많은 차들이 올라가 있었기 때문에 차가 정상적으로 못 움직였다. 하지만, 곰곰히 생각해보면 그런 테스트 과정에서도 정상적으로 작동하는 차가 진정하게 성공한 것이고 많은 어려움이 있을 수록 고치고 배워야 할 점이 많다는 것도 깨달을 수 있었다. 두 번째는 배운 것에 비해서 새로운 것들을 시도하는 것에 어려움을 많이 겪었는데 이것도 새로운 것들을 찾아보면서 직접 경험해보고 난관에도 처해보면서 정말 지식적인 면에서도 생학하는 면에서도 많은 부분을 배웠다고 생각한다. 쉽지 않은 활동들이었지만 좋게 마무리되서 너무 좋았고 팀원들과의 소통도 원활하여 최고였다고 생각한다. 다음에 들어오는 신입생들을 위해 가장 추천해주고 싶은 프로젝트이다. (20191598 박정명)

창업연계공학설계입문 수업시간 초반에 AD 프로젝트를 진행한다고 들었을 때는 아무것도 못하고 수업 교안의 pdf 를 따라가면서 작성한 코드만 제출할 것 같다고 조원들끼리 이야기를 했었다. 하지만 시험공부도 하고 수업시간 과제도 하고 xycar 를 사용하는 유레카 수업의 대회까지 준비해보니까 코드를 작성하고 수정할 수 있게 되었다.
시간이 더 있었다면 더 창의적이고 실제 자율주행자동차에 적용할 프로젝트를 진행할 수 있을 것 같아서 아쉬웠다. AD 프로젝트를 진행하면서 좋은 팀원들과 끝까지 포기하지 않고 서로에게 힘이 되어주면서 초기 구현하고 싶었던 신호등부터 추가 구현하고 싶었던 장애물 탐지까지 성공하여 더 뜻 깊고 재미있었다. (20191604 백연선)

창업연계공학설계입문 AD 프로젝트는 많은 것을 배울 수 있게끔 해주었다. 배운 것을 응용하는 능력과 수업시간에 배우지 않은 새로운 내용들을 따로 공부하여 배운 것에 적용하는 능력을 기를 수 있었다. 제일 좋았던 것은 좋은 팀원들을 만나 서로 협동하고, 난관에 봉착하더라도 다같이 이겨내는 팀적인 능력을 기를 수 있었다는 것이다. 물론 차를 작동할 수 있는 공간이 협소하다는 점과, 우리 조와 달리 다른 조는 조원이 조별 활동에 전혀 참여하지 않아 많은 고민을 하며 스트레스를 받는 조들도 존재했다는 점은 아쉬웠다. 그래도 다양하고 새로운 경험을 할 수 있게 해주어 학생들이 한층 성장할 수 있는 계기가 되게 해준 과목이였다. (20191610 성정규)

예상과 다른 결과와 변수들에 어려움이 있었지만 이번 창업연계공학설계입문 AD 프로젝트 활동을 통해서 팀프로젝트의 중요성에 대해서도 깨닫고 스스로 돌아보고 성찰할 수 있는 뜻 깊은 시간을 갖게 되었다고 생각합니다. (20191617 신지효)