

xycar 면허 도로주행 시험

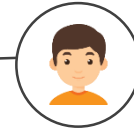
성창엽



신혜은



박준서



백현식



오탉봉

목차

주제 선정 이유

도로 주행 시험 상세사항

코드에 대한 간단한 설명

한계점

주제: xycar 면허 도로주행 시험

주제 선정 이유:

어디서나 쉽게 구할 수 있으며 타는 방법 또한 간단한 전동킥보드와 같은 이동수단도 운전할 수 있는 능력이 있음을 알려주는 면허증이 있어야 이용이 가능한 법률이 있는 것 처럼 이 시대는 발전하는 기술에 따라 그에 맞는 안전 수칙과 법률 등이 중요시 되고 있다.

이처럼 자율주행 자동차인 xycar 또한 운전자가 운전할 수 있다는 자격을 증명할 수 있는 면허증이 필요하다고 생각되어 ad 프로젝트 주제로 'xycar 면허 시험'을 선정하게 되었다.

도로주행시험 상세사항

xycar 면허는 도로 주행 시험으로 취득할 수 있으며, 기준은 다음과 같다.

- 1) 차선을 이탈하였는가?
- 2) 주차할 때 벽과 일정 거리 이상 떨어져 있나?
- 3) 정해진 시간 안에 들어왔는가?

도로주행시험 상세사항

'Lane Departure'

: 도로에 있는 차선을 벗어난 횟수



도로주행시험 상세사항

'Parking_mid'

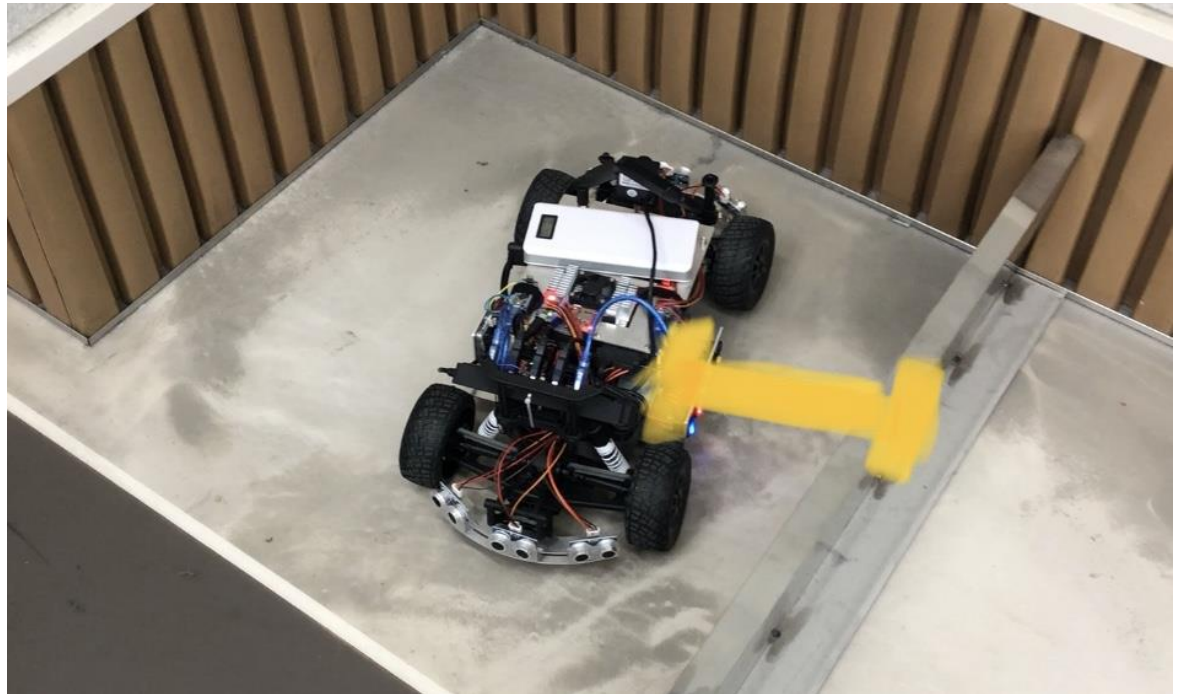
: 주차를 완료하였을 때 차의 중간에 있는 초음파 센서와 마주보고 있는 벽면과의 거리



도로주행시험 상세사항

'Parking_right'

: 주차를 완료하였을 때 차의 오른쪽에 있는 초음파 센서와 마주보고 있는 벽면과의 거리



도로주행시험 상세사항

'Parking_left'


: 주차를 완료하였을 때 차의 왼쪽에 있는 초음파 센서와 마주보고 있는 벽면과의 거리



도로주행시험 상세사항

```
Lane Departure :      2
Pakring_mid :      12
Pakring_left :      36
Pakring_right :      16 ... Fail
Runnung Time :      83.93sec
    ## FAIL ##
```

도로 주행 완료시, 성적을 바로 확인할 수 있으며 그에 맞는 합불 여부도 받아볼 수 있다.



Lane Departure : 2
Pakring_mid : 12
Pakring_left : 36
Pakring_right : 16 ... Fail
Runnung Time : 83.93sec
FAIL

코드에 대한 설명

```
class LineDetector:

    def __init__(self, topic):
        self.bridge = CvBridge()
        self.frame = np.empty(shape=[0])
        self.image_width = 640
        self.scan_width, self.scan_height = 270, 40
        self.lmid, self.rmid = self.scan_width, self.image_width - self.scan_width
        self.area_width, self.area_height = 10, 5
        self.roi_vertical_pos = 310
        self.row_begin = (self.scan_height - self.area_height) // 2
        self.row_end = self.row_begin + 0.04 * self_width * self.area_height
        rospy.Subscriber(topic, Image, self.conv_image)

    def conv_image(self, data):
        self.frame = self.bridge.imgmsg_to_cv2(data, "bgr8")

    def detect_lines(self):
        self.left, self.right = -1, -1
        self.roi = self.frame[self.roi_vertical_pos:self.roi_vertical_pos +
                               self.scan_height, :]
        self.hsv = cv2.cvtColor(self.roi, cv2.COLOR_BGR2HSV)
        self.avg_value = np.average(self.hsv[:, :, 2])
        self.value_threshold = self.avg_value * 0.8
        self.gray = cv2.cvtColor(self.roi, cv2.COLOR_BGR2GRAY)
        self.blur = cv2.GaussianBlur(self.gray, (5, 5), 0)
        self.edges = cv2.Canny(self.blur, 50, 150)
        self.edges = cv2.cvtColor(self.edges, cv2.COLOR_GRAY2BGR)
        lbound = np.array([0, 0, self.value_threshold], dtype=np.uint8)
        ubound = np.array([100, 255, 255], dtype=np.uint8)
```

코드에 대한 설명

```

lbound = np.array([0, 0, self.value_threshold], dtype=np.uint8)
ubound = np.array([100, 255, 255], dtype=np.uint8)
self.edges = cv2.cvtColor(self.edges, cv2.COLOR_BGR2HSV)
self.bin = cv2.inRange(self.edges, lbound, ubound)
self.view = cv2.cvtColor(self.bin, cv2.COLOR_GRAY2BGR)

for i in range(self.area_width, self.lmid):
    area = self.bin[self.row_begin:self.row_end, 1 - self.area_width:1]
    if cv2.countNonZero(area) > self.pixel_cnt_threshold:
        self.left = 1
        break

for r in range(self.image_width - self.area_width, self.rmid, -1):
    area = self.bin[self.row_begin:self.row_end, r:r + self.area_width]
    if cv2.countNonZero(area) > self.pixel_cnt_threshold:
        self.right = r
        break

return self.left, self.right

def show_images(self, left, right):
    if left != -1:
        lsquare = cv2.rectangle(self.view,
                                (left - self.area_width, self.row_begin),
                                (left, self.row_end),
                                (0, 255, 0), 3)
    if right != -1:
        rsquare = cv2.rectangle(self.view,
                                (right, self.row_begin),
                                (right + self.area_width, self.row_end),
                                (0, 255, 0), 3)

```

코드에 대한 설명

```
class License_test:

    def __init__(self):
        rospy.init_node('xycar_driver')
        self.line_detector = LineDetector('/usb_cam/image_raw')
        self.obstacle_detector = ObstacleDetector('/ultrasonic')
        self.count = 0
        self.depart_start = -5

    def line_test(self):
        line_l, line_r = self.line_detector.detect_lines()
        self.line_detector.show_images(line_l, line_r)
        if 5 < time.time() - self.depart_start:
            if line_l == -1 & line_r == -1:
                self.count += 1
                self.depart_start = time.time()

    def parking_test(self):
        return self.obstacle_detector.get_distance()

    def totalScore(self, line, parking_l, parking_m, parking_r, time):
        if time > 60:
            return 0, "Fail"

        parking = "Success"
        if (parking_l > 3) and (parking_m > 3) and (parking_r > 3):
            totalscore = 100
```

코드에 대한 설명

```
else:
    totalscore = 80
    parking = "Fail"

totalscore -= 5 * line

return totalscore, parking

def exit(self):
    print("finished")

if name == 'main':
    startTime = time.time()
    test_car = License_test()
    time.sleep(3)
    rate = rospy.Rate(15)
    while cv2.waitKey(1) & 0xFF != 27: #escape key
        test_car.line_test()
        rate.sleep()
    while cv2.waitKey(1) & 0xFF != 32: #space bar key
        left, mid, right = test_car.parking_test()
        rate.sleep()

    cv2.destroyAllWindows()
    finishTime = time.time()
    totalTime = finishTime - startTime
    score, result = test_car.totalScore(test_car.count, lefh, mid, right, totalTime)
```

코드에 대한 설명

```
#show license result
print("-" * 40)
print("Lane Departure : \t%d" %test_car.count)
print("Parking_mic : \t\t%d" %mid)
print("Parking_left : \t\t%d" %left)
print("Parking_right : \t%d ... %s" %(right, result))
print("Running Time : \t\t%.2fsec" %totalTime)

if score > 60:
    print("\n## PASS ##")
else:
    print("\t## FAIL ##")
print("-" * 40)

rospy.on_shutdown(test_car.exit)
```

한계점

1. 부정확한 차선 인식 - 차선을 반만 걸칠 경우 판별 불가
2. 터미널 인터페이스 개선
3. 모터 노드의 사용 불가