


창업연계공학설계입문
AD Project 최종 발표

2조 : 장정안, 정승우,
정수환, 정민지, 장승훈




목차

01 목표

02 목표 구현

03 시연 영상

01 목표

01 목표

신호등 감지

빨강, 초록 신호를 감지한다

표지판 인식

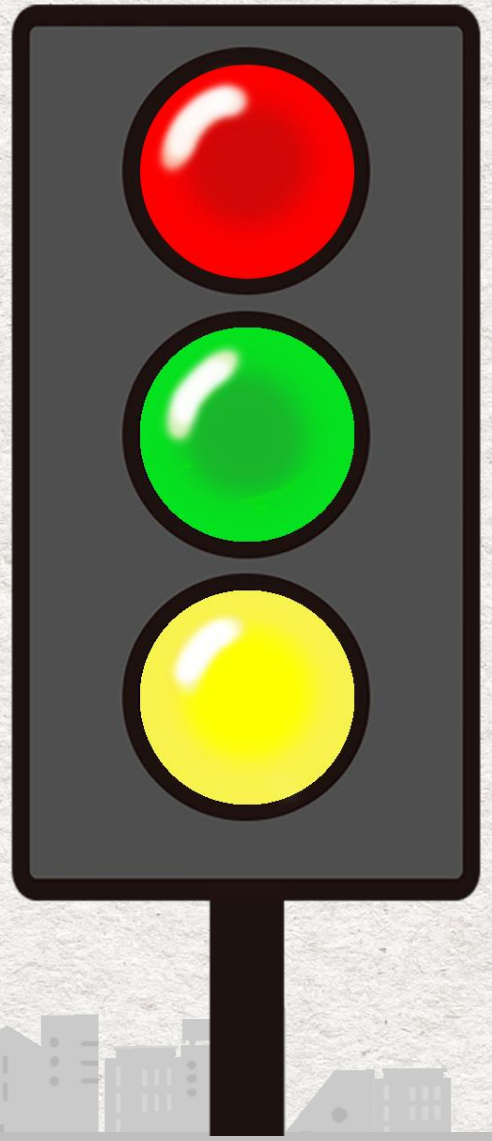
속도 표지판을 인식한다.

T자 주차

여러가지 주차 방법 중 T자 주차를 한다.

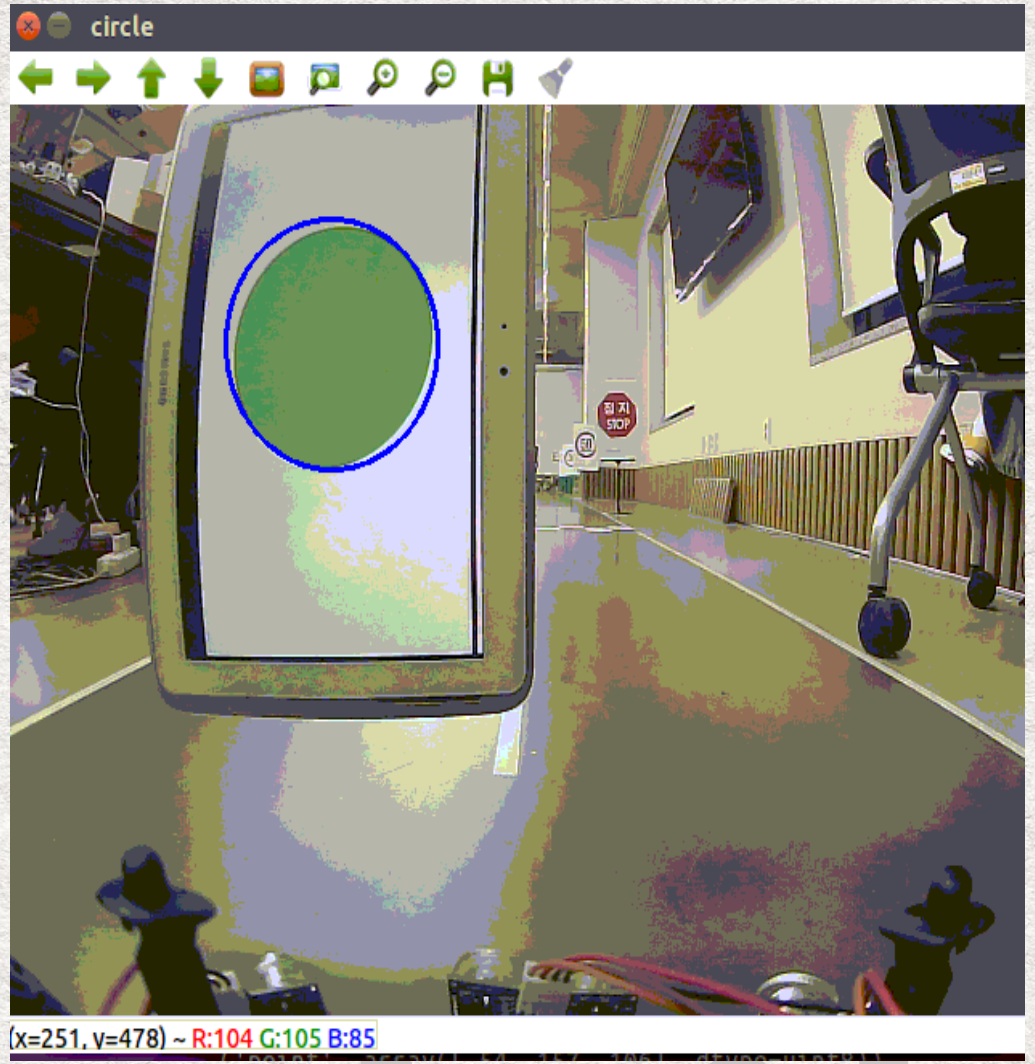

02 목표 구현

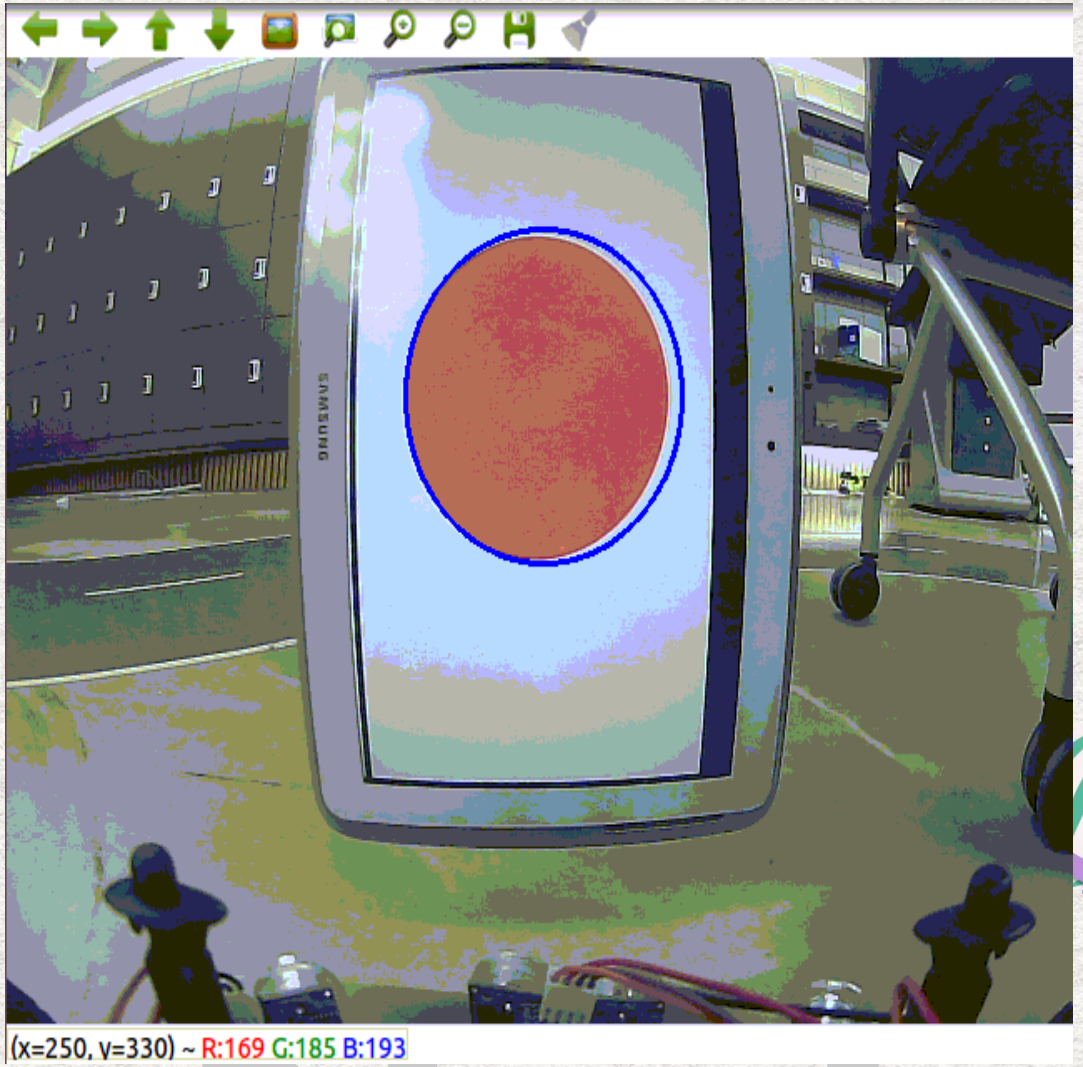
3.1 신호등 감지



자동차는 신호등에 빨강색의 불이 들어올 때 정지하고 초록불이 들어올 때 출발한다. 이를 위해 먼저 원을 검출한 후 그 원의 색을 검출해내어 빨강이면 정지, 초록색이면 출발하게끔 구현한다.

3.1 신호등 감지





3.1 신호등 감지

```
circle_roi = self.cam_img[:380,:]
gray = cv2.cvtColor(circle_roi, cv2.COLOR_BGR2GRAY)
gray = cv2.medianBlur(gray,5)

circles = cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,1,20,
                param1=150,param2=70,minRadius=50,maxRadius=0)
point = []
if circles is None:
    pass
else:
    circles = np.uint(np.around(circles))
    for c in circles[0,:]:
        center = (c[0],c[1])
        radius = c[2]

        self.cam_img = cv2.circle(self.cam_img,center,radius,(255,0,0),2)
        print("point", self.cam_img[center[1]][center[0]])
        point = self.cam_img[center[1]][center[0]]



cv2.imshow("check", frame)
cv2.imshow("des30", self.up_roi)
cv2.imshow("circle", self.cam_img)
cv2.waitKey(5)
return point
```

- param1 :
HOUGH_GRADIENT의 경우 canny edge detecto의 높은 threshol값이다. 낮은 thres값은 0.5배해서 사용하게 된다.

- param2 :
HOUGH_GRADIENT의 경우 accumulator threshoding값이다. 이 값이 너무 낮으면 거짓 원이 검출된다.

- minRadius : 검출하고자 하는 원의 최소 반지름의 값이다. default값으로 0으로 지정할 수 있다.

- maxRaidus : 검출하고자 하는 원의 최대 반지름의 값이다. default값으로 0으로 지정할 수 있다.

3.1 신호등 감지

```
if len(point) == 0:
    pass
else:


    if point[1] >= 155 and point[2] >= 155:
        pass
    elif 180 <= point[2]:
        print("red")
        self.speed2 = 0
        self.speed = 0
    elif 150 <= point[1]:
        print("green")
        self.speed2 = 15



return self.speed2 + self.speed
```

- if문으로 point리스트의 3번째 요소, BGR에서 R값이 180보다 큰지 확인해서 만일 들어온 값이 180 보 다 크 다 면 , red 로 인 식 하 고 속 도 를 90 으 로 지정한다.

- point리스트의 2번째 요소, G값이 150보다 큰지 확인하고 만 일 크 다 면 green 으 로 인식하고 속도를 75로 지정한다.

- 표 지 판 을 인 식 해 서 멈 추 면 안되므로 맨 처음에 G와 R값이 155이상일 때는 pass한다.

3.2 표지판 인식

시속 30km
표지판 이미지

출발 speed 값 보다

5만큼 빠르게

.....

시속 40km
표지판 이미지

출발 speed 값 보다

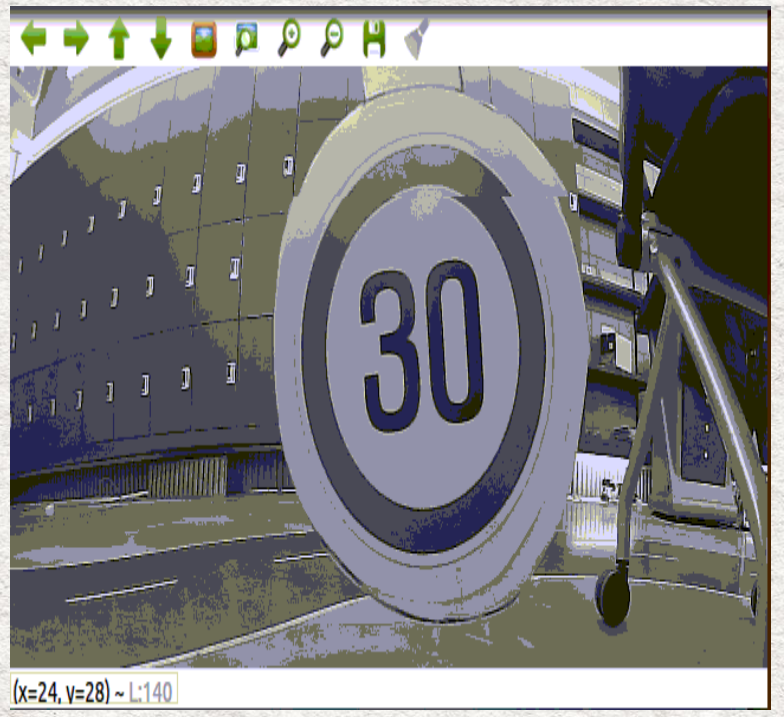
10만큼 빠르게

시속 50km
표지판 이미지

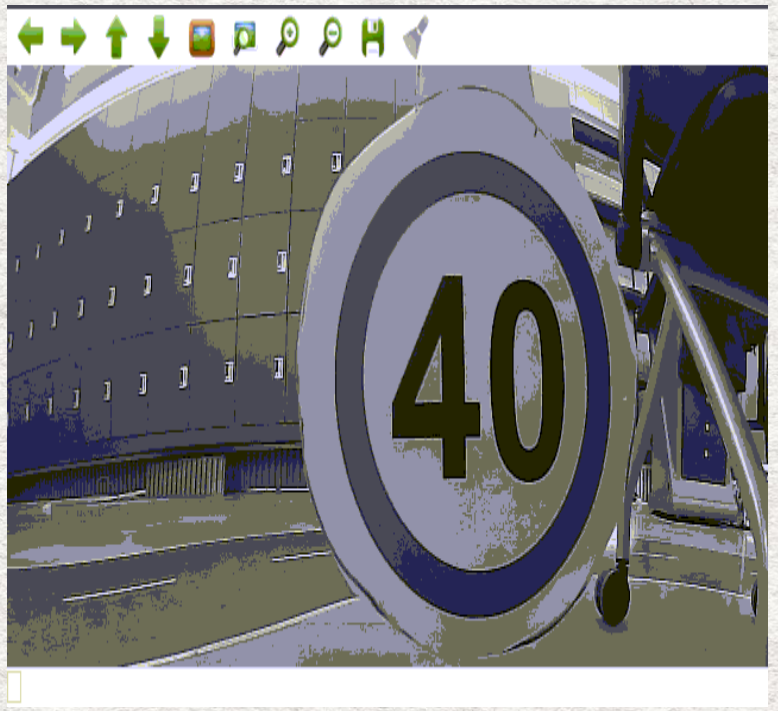
출발 speed 값 보다

15만큼 빠르게

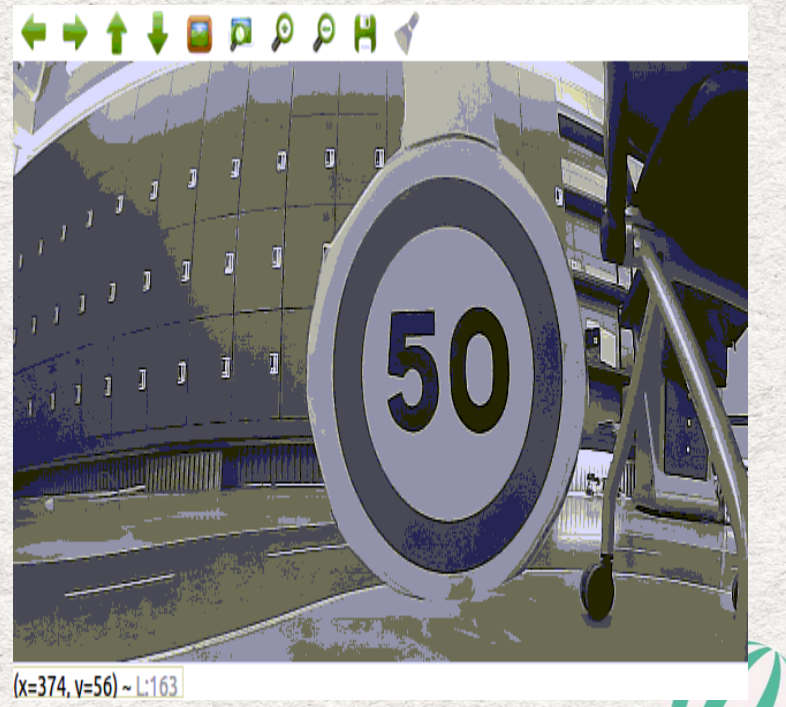
3.2 표지판 인식




```
(31, 2, 3)
('point', array([150, 146, 138], dtype=uint8))
('point', array([57, 53, 51], dtype=uint8))
speed is 30
```




```
speed is 40
(10, 27, 10)
```




```
speed is 50
(9, 3, 18)
```

3.2 표지판 인식

```
        self.orb = cv2.ORB_create()
        self.bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

        self.img30 = cv2.imread('/home/nvidia/xycar/src/auto_drive/src/30km.jpg',
cv2.IMREAD_GRAYSCALE)
        self.kp30, self.des30 = self.orb.detectAndCompute(self.img30, None)

        self.img40 = cv2.imread('/home/nvidia/xycar/src/auto_drive/src/40km.jpg',
cv2.IMREAD_GRAYSCALE)
        self.kp40, self.des40 = self.orb.detectAndCompute(self.img40, None)

        self.img50 = cv2.imread('/home/nvidia/xycar/src/auto_drive/src/50km.jpg',
cv2.IMREAD_GRAYSCALE)
        self.kp50, self.des50 = self.orb.detectAndCompute(self.img50, None)

        self.bridge = CvBridge()
```

3.2 표지판 인식

```
def img_match(self):
    kp, des = self.orb.detectAndCompute(self.up_roi, None)
    if des is None:
        print("des is None")
        return 0, 0, 0
    elif self.des30 is None:
        print("des 30 is None")
        return 0, 0, 0
    matches30 = self.bf.match(des, self.des30)
    matches30 = sorted(matches30, key=lambda x:x.distance)
    matches40 = self.bf.match(des, self.des40)
    matches40 = sorted(matches40, key=lambda x:x.distance)
    matches50 = self.bf.match(des, self.des50)
    matches50 = sorted(matches50, key=lambda x:x.distance)

    dist30 = [m.distance for m in matches30 if m.distance < 38]
    dist40 = [m.distance for m in matches40 if m.distance < 38]
    dist50 = [m.distance for m in matches50 if m.distance < 38]


    return len(dist30), len(dist40), len(dist50)
```

3.2 표지판 인식

```
def __init__(self):
    rospy.init_node('xycar_driver')
    self.line_detector = LineDetector('/usb_cam/image_raw')
    self.obstacle_detector = ObstacleDetector('/ultrasonic')
    self.driver = MotorDriver('/xycar_motor_msg')

    self.speed = 0
    self.speed2 = 0
```

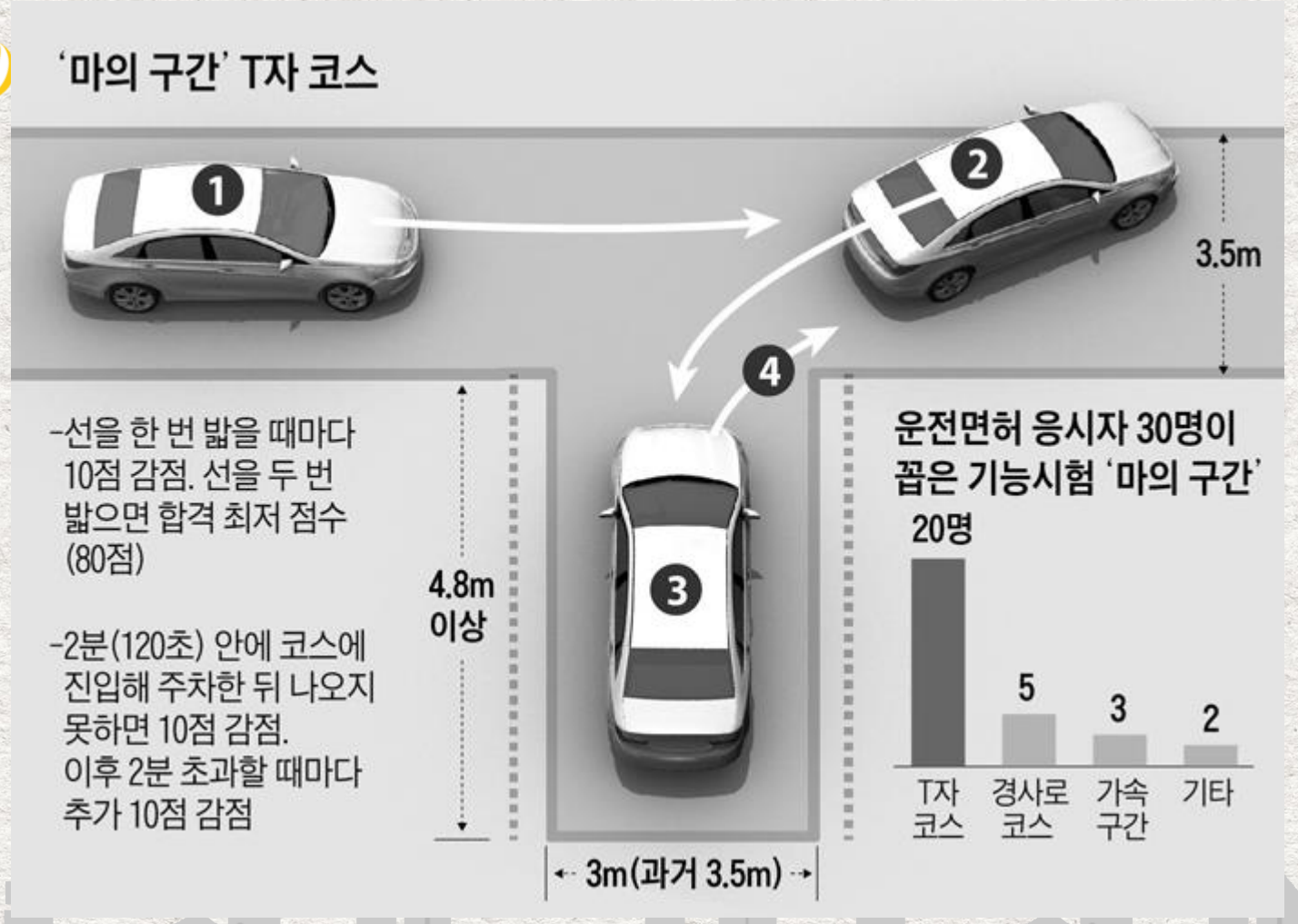
```
if self.len30 > 30:
    self.speed = 5
    print("speed is 30")

elif self.len40 > 20:
    self.speed = 10
    print("speed is 40")


elif self.len50 > 15:
    self.speed = 15
    print("speed is 50")
```

```
return self.speed2 + self.speed
```

3.3 T자 주차



초음파 센서를 사용하여 조향각을 조절하며 T자 주차를 한다.

3.3 T자 주차

```
def parkingMatch(self):
    self.img1 = cv2.imwrite('/home/nvidia/xycar/src/auto_drive/src/f.jpg', self.cam_img)
    self.img2 = cv2.imread('/home/nvidia/xycar/src/auto_drive/src/f.jpg', cv2.IMREAD_GRAYSCALE)
    self.kp2, self.des2 = self.orb.detectAndCompute(self.img2, None)

    self.imgTrainColor = cv2.imread('/home/nvidia/xycar/src/auto_drive/src/parking.jpg', cv2.IMREAD_GRAYSCALE)

    self.kp1, self.des1 = self.orb.detectAndCompute(self.imgTrainColor, None)

    self.matches = self.bf.match(self.des1, self.des2)

    self.matches = sorted(self.matches, key=lambda x: x.distance)

    self.dist = [m.distance for m in self.matches if m.distance < 50]

    print(self.dist)

    if len(self.dist) >= 12:
        self.result = True
        return self.result
```

3.3 T자 주차

```
def __init__(self, topic):
    self.left = -1
    self.mid = -1
    self.right = -1
    self.filter_0 = []
    self.filter_1 = []
    self.filter_2 = []
    self.filter_3 = []
    self.filter_4 = []
    self.filter_5 = []
    self.filter_6 = []
    self.filter_7 = []
    self.weights = list(range(1, 6))
    rospy.Subscriber(topic, Int32MultiArray, self.read_distance)
```

```
def for_parking(self, data):
    self.parking_0 = self.filter(self.filter_0, data.data[0], self.weights)
    self.parking_3 = self.filter(self.filter_3, data.data[3], self.weights)
    self.parking_4 = self.filter(self.filter_4, data.data[4], self.weights)
    self.parking_6 = self.filter(self.filter_6, data.data[6], self.weights)
```

3.3 T자 주차

```
def filter(self, filterLst, data, weights):
    s = 0
    if data < 300:
        if len(filterLst) < 5:
            filterLst.append(data)
        else:
            filterLst = filterLst[1:] + [data]
        for i, x in enumerate(filterLst):
            s += x * weights[i]
            s = float(s) / sum(weights[:len(filterLst)])
    else:
        if len(filterLst) < 5:
            filterLst.append(300)
        else:
            filterLst = filterLst[1:] + [300]
        for i, x in enumerate(filterLst):
            s += x * weights[i]
            s = float(s) / sum(weights[:len(filterLst)])
    return s
```

```
def for_parking_distance(self):
    return self.parking_0, self.parking_3, self.parking_4, self.parking_6
```

3.3 T자 주차

```
def parking(self):
    parking_0, parking_3, parking_4, parking_6 = self.obstacle_detector.for_parking_distance()
    while parking_6 <= 50:
    for go1 in range(5):
        drive(85, 110)
        if parking_6 >= 50:
            time.sleep(1)
            break
        time.sleep(0.1)

    for stop1 in range(2):
        drive(90, 90)
        time.sleep(0.1)
        drive(65, 110)
        time.sleep(0.1)

    for left1 in range(50):
        drive(65, 110)
        if data_0 <= 5:
            time.sleep(1)
            break
        time.sleep(0.1)

    for stop2 in range(2):
        drive(90, 90)
        time.sleep(0.1)
        drive(180, 70)
        time.sleep(0.1)

    for back1 in range(30):
        drive(180, 70)
        if data_3 <= 5:
            time.sleep(1)
            break
        time.sleep(0.1)
```

```
for stop3 in range(2):
    drive(90, 90)
    time.sleep(0.1)
    drive(70, 110)
    time.sleep(0.1)

for go2 in range(20):
    drive(70, 110)
    if data_0 <= 5:
        time.sleep(1)
        break
    time.sleep(0.1)

for stop4 in range(2):
    drive(90, 90)
    time.sleep(0.1)
    drive(140, 70)
    time.sleep(0.1)

for back2 in range(5):
    drive(140, 70)
    time.sleep(0.1)

for stop5 in range(2):
    drive(90, 90)
    time.sleep(0.1)
    drive(90, 110)
    time.sleep(0.1)

for go3 in range(5):
    drive(90, 110)
    time.sleep(0.1)
```

```
    for go3 in range(5):
        drive(90, 110)
        time.sleep(0.1)

    for stop6 in range(2):
        drive(90, 90)
        time.sleep(0.1)
        drive(90, 70)
        time.sleep(0.1)

    for back3 in range(50):
        drive(90, 70)
        if data_4 <= 5:
            time.sleep(1)
            break
        time.sleep(0.1)
    for finish in range(2):
        drive(90, 90)
        time.sleep(0.1)
        drive(90, 90)
        time.sleep(0.1)
    rate.sleep()
    break
rospy.on_shutdown(exit_node)
```

03 시연 영상


Thanks for watching!