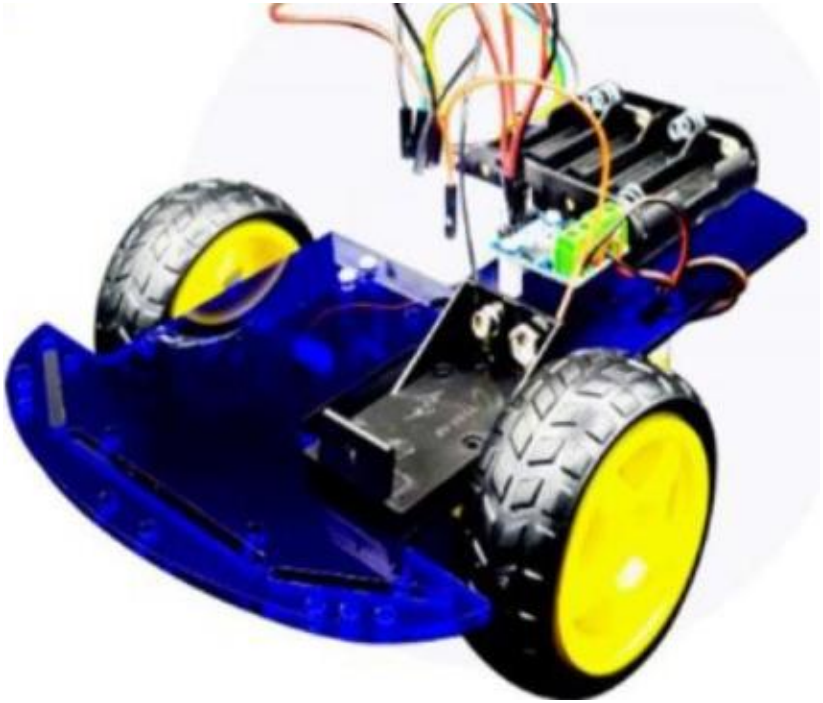
자율주행자동차 SW

자율주행자동차 HW

- **자율주행자동차 HW**
  - 라즈베리파이 + 아두이노 보드
  - 카메라로 전방 도로 촬영, 자율주행



자율주행자동차 자료

- https://www.robotshop.com/community/robots/show/motoko-aftermath-line-follower
- https://github.com/georgesung/road_lane_line_detection
- https://github.com/Williangalvani/RaspLineStalker
- https://github.com/CRM-UAM/VisionRace
  - openCV를 이용하여 도로 윤곽선 및 기울기 계산

자율주행자동차 SW

- **방법 1**
  - OpenCV를 이용한 도로 곡선 계산
  - Ex) Autonomous Racing Robot
    - https://becominghuman.ai/autonomous-racing-robot-with-an-Arduino-a-raspberry-pi-and-a-pi-camera-3e72819e1e63
    - 소스코드 주소
      - https://github.com/sergionr2/RacingRobot
- **방법 2**
  - Deep Learning을 이용한 자율 주행

방법 1: 자율주행자동차 기본 알고리즘 (강의록게시판:opencv_countour_example.zip 볼 것)

```
# -*- coding: utf-8 -*-
# **********main.py**********
import socket
import sys
import os
import numpy as np
import pdb

import cv2
import time

from Image import *
from Utils import *

font = cv2.FONT_HERSHEY_SIMPLEX
direction = 0

#N_SLICES만큼 이미지를 조각내서
# Images[] 배열에 담는다
Images=[]
N_SLICES = 3

for q in range(N_SLICES):
    Images.append(Image())

img = cv2.imread('dave.jpg')

if img is not None:
    #이미지를 조각내서 윤곽선을 표시하게
    # 무게중심 점을 얻는다
    Points = SlicePart(img, Images, N_SLICES)
    print('Points : ', Points)

    #N_SLICES 개의 무게중심 점을 x좌표, y좌표끼리 나눈다
    x = Points[::2]
    y = Points[1::2]

    #조각난 이미지를 한 개로 합친다
    fm = RepackImages(Images)

    #완성된 이미지를 표시한다
    cv2.imshow("Vision Race", fm)
    if cv2.waitKey(0) & 0xFF == ord('q'):
        cv2.destroyAllWindows()
else:
    print('not even processed')
```

카메라에서 영상 수집하도록 수정

방향과 속도계산, 모터제어 루틴 추가

방법 1: Opencv 3.x 버전 수정 사항

- **Opencv_countour_example은 opencv 2.4.x 버전임**
- **Opencv 3.x 버전의 경우 image.py를 아래와 같이 수정**
  - image.py의 17번째줄 맨 뒤에 [-2:] 추가
    - self.contours, _ = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)[-2:]
  - 44,45번째 줄
    - 'cv2.CV_AA'를 모두 'cv2.LINE_AA'로 수정 (2곳)
  - 수정 후 winPython 2.7로 실행

방법 1: 자율주행자동차 기본 알고리즘 (utils.py)

```
# -*- coding: utf-8 -*-
import numpy as np
import cv2
import time
from Image import *

# 그림을 slices 의 수만큼 조각낸다
def SlicePart(im, images, slices):
    height, width = im.shape[:2]
    sl = int(height/slices);
    points = []

    for i in range(slices):
        part = sl*i
        crop_img = im[part:part+sl, 0:width]
        #조각난 이미지 crop_img를 images[]에 저장
                images[i].image = crop_img
                #Image.py에서 윤곽선을 그리고 무게중심을 표시
        cPoint = images[i].Process()
        points.append(cPoint)
    return points
```

```
#조각난 이미지를 다시 합친다
def RepackImages(images):
    img = images[0].image
    for i in range(len(images)):
        if i == 0:
            img = np.concatenate((img, images[1].image), axis=0)
        if i > 1:
            img = np.concatenate((img, images[i].image), axis=0)

    return img

def Center(moments):
    if moments["m00"] == 0:
        return 0

    x = int(moments["m10"]/moments["m00"])
    y = int(moments["m01"]/moments["m00"])
    return x, y
```

방법 1: 자율주행자동차 기본 알고리즘 (utils.py)

```
def RemoveBackground(image, b):
    up = 100
    # create NumPy arrays from the boundaries
    lower = np.array([0, 0, 0], dtype = "uint8")
    upper = np.array([up, up, up], dtype = "uint8")
    #----------------COLOR SELECTION---------- (Remove any area that is whiter than 'upper')
    if b == True:
        mask = cv2.inRange(image, lower, upper)
        image = cv2.bitwise_and(image, image, mask = mask)
        image = cv2.bitwise_not(image, image, mask = mask)
        image = (255-image)
        return image
    else:
        return image
    #////////////////COLOR SELECTION/////////////
```

방법 2: 딥 러닝 자율주행

- **딥 러닝 자율주행**
  - 초기화: 도로 이미지들을 이용하여 학습
  - 무한 루프
    - 도로 사진을 찍어 도로 휘어진 정도를 딥 뉴럴 네트워크로 계산
    - 모형 자동차 왼쪽 바퀴과 오른쪽 바퀴의 회전 값을 조절
- **딥러닝 자율주행 SW 플랫폼 다운로드**
  - 강의록 게시판에서 AI_RC_CAR.zip 다운로드

방법 2: 딥러닝 자율주행

- **SW 플랫폼 SW**
  - AI_RC_Car/
    - get_image_data.py
    - rc_car_interface.py
    - self_driving.py
    - tf_learn.py
    - traindata.p: 학습용 도로 사진과 라벨
  - AI_RC_CAR/TF_test
    - get_image_data.py
    - image_playback.py
    - learntest.py
    - traindata.p: 학습용 도로 사진과 라벨

딥러닝 자율주행: self_driving.py

```
from rc_car_interface import RC_Car_Interface
from tf_learn import DNN_Driver
import numpy as np
import time
import cv2
class SelfDriving:
    def __init__(self):
        self.rc_car_cntl = RC_Car_Interface()
        self.dnn_driver = DNN_Driver()
        self.rc_car_cntl.set_left_speed(0)
        self.rc_car_cntl.set_right_speed(0)
        self.velocity = 0
        self.direction = 0
        self.dnn_driver.tf_learn()
    def rc_car_control(self, direction):
        #calculate left and right wheel speed with direction
        if direction < -1.0:
            direction = -1.0
        if direction > 1.0:
            direction = 1.0
        if direction < 0.0:
            left_speed = 1.0+direction
            right_speed = 1.0
        else:
            right_speed = 1.0-direction
            left_speed = 1.0

        self.rc_car_cntl.set_right_speed(right_speed)
        self.rc_car_cntl.set_left_speed(left_speed)

    def drive(self):
        while True:

# For test only, get image from DNN test images
#             img from get_test_img() returns [256] array.
#             Do not call np.reshape()
#             img = self.dnn_driver.get_test_img()
#             direction = self.dnn_driver.predict_direction(img)

            img = self.rc_car_cntl.get_image_from_camera()
# predict_direction wants [256] array, not [16,16].
# Thus call np.reshape to convert [16,16] to [256] array
            img = np.reshape(img,img.shape[0]**2)

# predict with single image
            direction = self.dnn_driver.predict_direction(img)
            print(direction[0][0])
            self.rc_car_control(direction[0][0])

            # For debugging, show image
#             cv2.imshow("target",  cv2.resize(img, (280, 280)) )
#             cv2.waitKey(0)

            time.sleep(0.001)

        self.rc_car_cntl.stop()
        cv2.destroyAllWindows()

SelfDriving().drive()
```

딥러닝자율주행: rc_car_interface.py

```
import numpy as np
import cv2

from picamera.array import PiRGBArray
from picamera import PiCamera

class RC_Car_Interface():

    def __init__(self):
        self.left_wheel = 0
        self.right_wheel = 0
        self.camera = PiCamera()
        self.camera.resolution = (320,320)
        # set camera resolution to (320, 320)
        self.camera.color_effects = (128,128)
        # set camera to black and white

    def finish_iteration(self):
        print('finish iteration')

    def set_right_speed(self, speed):
        print('set right speed to ', speed)

    def set_left_speed(self, speed):
        print('set left speed to ', speed)

    def get_image_from_camera(self):
        img = np.empty((320, 320, 3), dtype=np.uint8)
        self.camera.capture(img, 'bgr')

        img = img[:,:,0]
        # 3 dimensions have the same value because camera is set to
black and white
        # remove two dimension data
#         print(img)

        threshold = int(np.mean(img))*0.5
#         print(threshold)

        # Invert black and white with threshold
        ret, img2 = cv2.threshold(img.astype(np.uint8), threshold, 255,
cv2.THRESH_BINARY_INV)

        img2 = cv2.resize(img2,(16,16), interpolation=cv2.INTER_AREA )
#         cv2.imshow("Image", img2)
#         cv2.waitKey(0)
        return img2

    def stop(self):      # robot stop
        print('stop')

# Test Only
# RC_Car_Interface().get_image_from_camera()
```

딥러닝 자율주행: tf_learn.py

```
# Copyright Reserved (2020).
# Donghee Lee, Univ. of Seoul
#__author__ = 'will'
from keras.models import Sequential
from keras.layers import Dense
#from sklearn.model_selection import train_test_split
import numpy as np
#import pandas as pd
import tensorflow as tf
#import pickle
from get_image_data import *
class DNN_Driver():
    def __init__(self):
        self.trX = None
        self.trY = None
        self.teX = None
        self.teY = None
        self.model = None
```

영상 설명 수정
TrX, TeX: 이미지
trY, TeY: 라벨

```
    def tf_learn(self):
        self.trX, self.trY = get_training_data()
        self.teX, self.teY = get_test_data()
        seed = 0
        np.random.seed(seed)
        tf.random.set_seed(seed)
        self.model=Sequential()
        self.model.add(Dense(512, input_dim=np.shape(self.trX)[1], activation='relu'))
        self.model.add(Dense(64, activation='relu'))
        self.model.add(Dense(1))
        self.model.compile(loss='mean_squared_error', optimizer='adam')
        self.model.fit(self.trX, self.trY, epochs=2, batch_size=1)
        return
    def predict_direction(self, img):
        print(img.shape)
#         img = np.array([np.reshape(img,img.shape**2)])
        ret =  self.model.predict(np.array([img]))
        return ret
    def get_test_img(self):
        img = self.teX[10]
        return img
```

딥러닝자율주행: get_image_data.py

```
import pickle
import numpy as np

data = pickle.load( open( "trainingdata.p", "rb" ), encoding="latin1" )
n_images = len(data)
test, training = data[0:int(n_images/3)], data[int(n_images/3):]

def get_training_data():

    trX = np.array([np.reshape(a[2],a[2].shape[0]**2) for a in training])
    print(np.shape(trX)[1])
    trY = np.zeros((len(training)),dtype=np.float)
    for i, data in enumerate(training):
        trY[i] = float(data[0])
    return trX, trY
'''
    if one_hot:
        trY = np.zeros((len(training),len(outputs)),dtype=np.uint8) #[onehot(a[0]) for a in training]
        for i, data in enumerate(training):
            trY[i][data[0] + 1] = 1
    else:
        trY = np.zeros((len(training),1),dtype=np.int8) #[onehot(a[0]) for a in training]
        for i, data in enumerate(training):
            trY[i][0] = data[0]
'''
```

```
def get_test_data():
    teX = np.array([np.reshape(a[2],a[2].shape[0]**2) for a in test])
    teY = np.zeros((len(test)),dtype=np.float)
    for i, data in enumerate(test):
        teY[i] = float(data[0])
    return teX,teY

'''
    if one_hot:
        teY = np.zeros((len(test),len(outputs)),dtype=np.uint8) #[onehot(a[0]) for a in training]
        for i, data in enumerate(test):
            teY[i][data[0] + 1] = 1
    else:
        teY = np.zeros((len(test),1),dtype=np.int8) #[onehot(a[0]) for a in training]
        for i, data in enumerate(test):
            teY[i][0] = data[0]
'''
```

```
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
...
```

딥러닝자율주행: image_playback.py

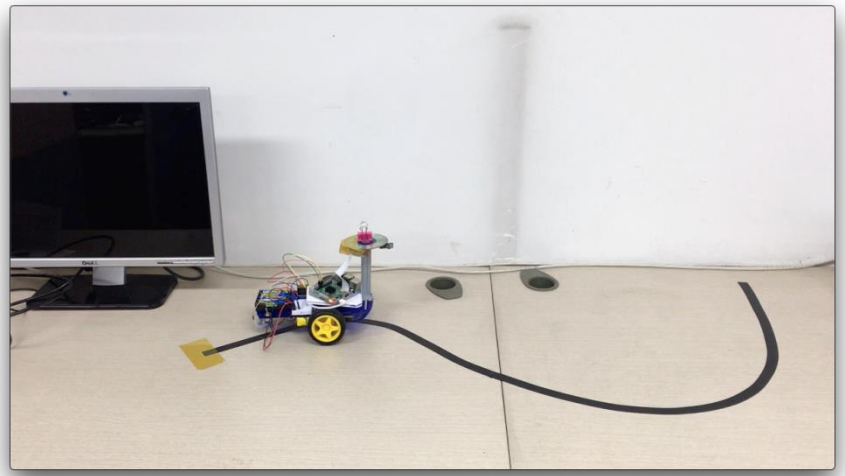
```
import pickle
import cv2
import time
import numpy as np

data = pickle.load( open( "trainingdata.p", "rb" ), encoding="latin" )
n_images = len(data)
print(n_images)
test, training = data[0:int(n_images/3)], data[int(n_images/3):]

cv2.namedWindow('disp')
for direcao,velocidade,img in data:
    img = np.array(img,dtype=np.uint8)
    print (direcao, velocidade)
    cv2.imshow('disp',np.array(cv2.resize(img,(280,280))))

#    time.sleep(0.05)
    cv2.waitKey(0)
cv2.destroyAllWindows()
```

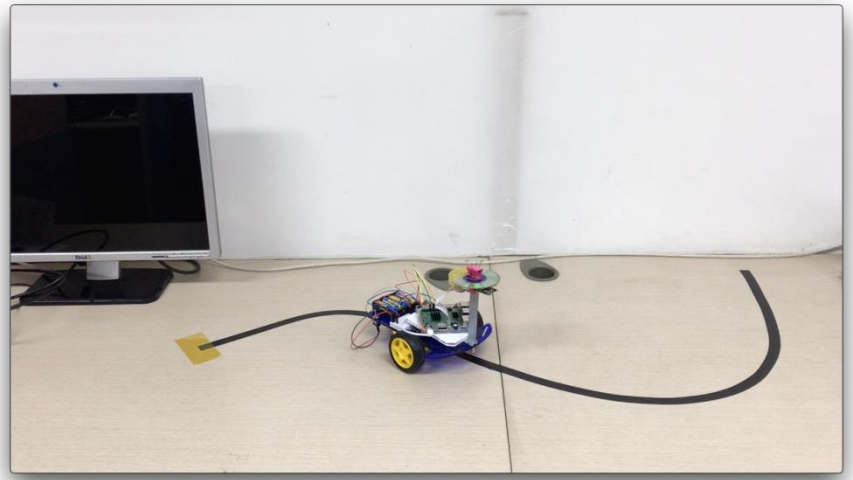
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
[d][v][i,,i,,i,...]
...

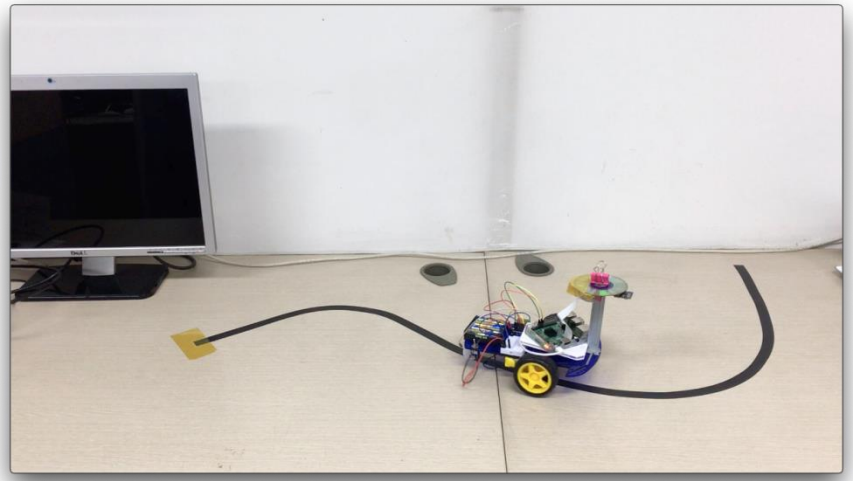
딥러닝자율주행: learntest.py

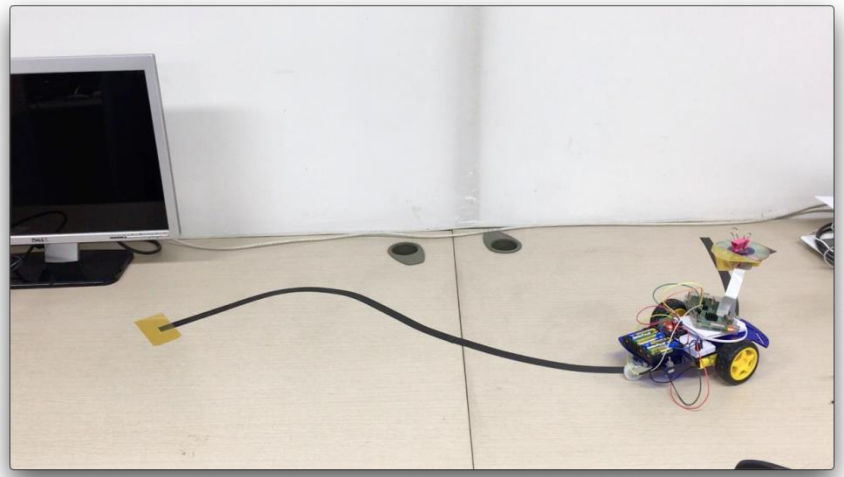
```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd
import tensorflow as tf
import pickle

outputs = 1

from get_image_data import *

trX,trY = get_training_data()
teX,teY = get_test_data()

seed = 0
np.random.seed(seed)
tf.random.set_seed(seed)

model=Sequential()
model.add(Dense(512, input_dim=np.shape(trX)[1], activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')

model.fit(trX, trY, epochs=50, batch_size=1)


Y_prediction = model.predict(teX).flatten()

for i in range(1000):
    label = teY[i]
    pred = Y_prediction[i]
    print("label:{:.2f}, pred:{:.2f}".format(label, pred))


def get_direction(img):
    print(img.shape)
#    img = np.array([np.reshape(img,img.shape**2)])
    ret =  model.predict(np.array([img]))
    return ret

# Predict direction with single image
dir=get_direction(teX[10])
print(dir[0][0])
```

자율주행자동차 시제작품

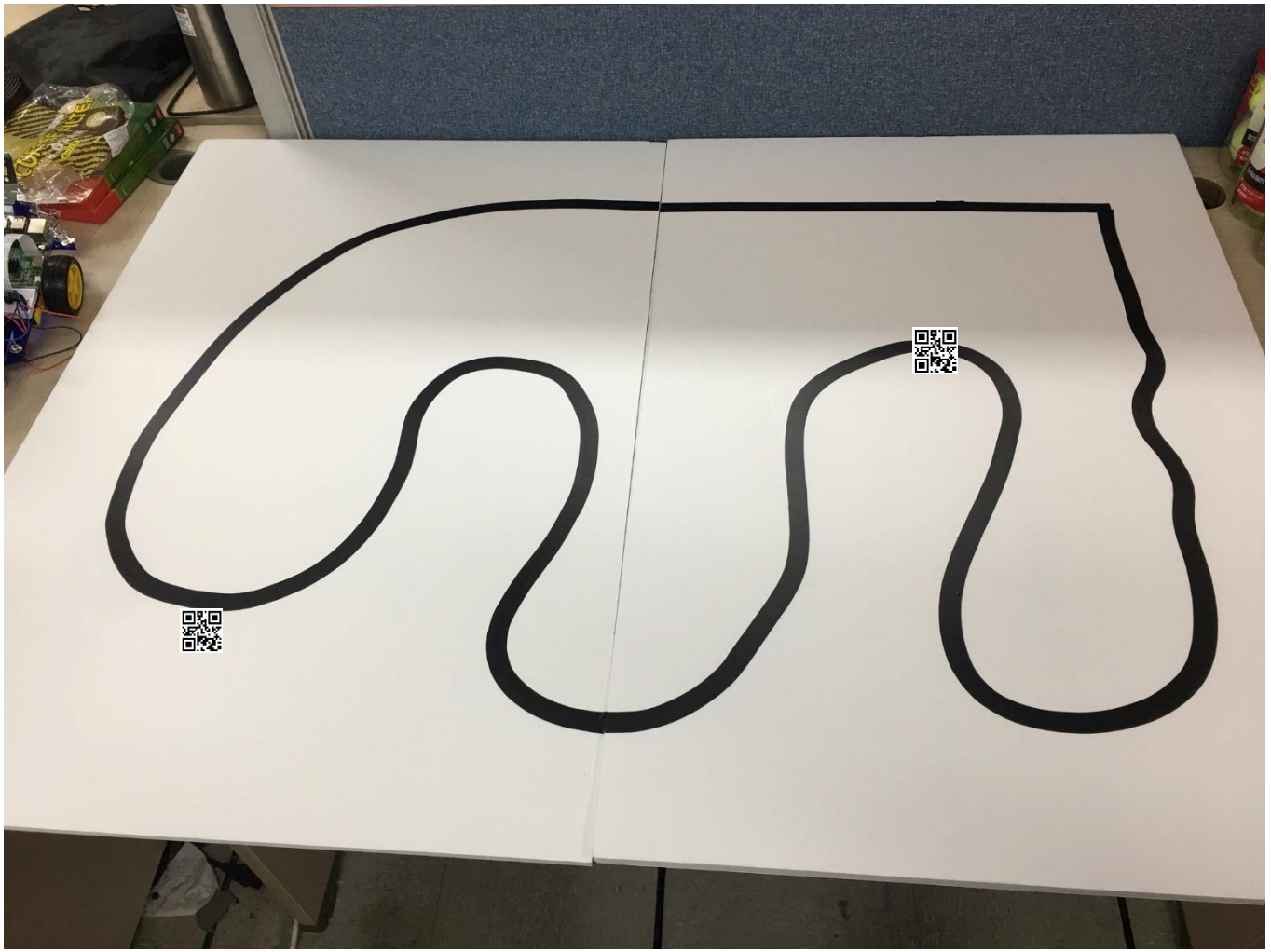








자율주행자동차 트랙



자율주행자동차 평가 기준

- 트랙을 다음과 같이 5단계로 구분
- 트랙의 각 단계 통과여부로 평가

트랙	직선	곡선	지그재그	직각	QR
통과여부 (O,X)