라인 트레이싱 코드

```
import cv2 as cv
import numpy as np
import threading, time
import SDcar
speed =100
grid =4
def func_thread():
    i =0
    while True:
        print("alive!!")
        time.sleep(1)
        i =i+1
        if is_running is False:
            break
def key_cmd(which_key):
    global enable_linetracing  # 전역 변수 선언
    print('which_key', which_key)
    is_exit =False
    if which_key &0xFF ==184:
        print('up')
        car.motor_go(speed)
    elif which_key &0xFF ==178:
        print('down')
        car.motor_back(speed)
    elif which_key &0xFF ==180:
        print('left')
        car.motor_left(speed)
    elif which_key &0xFF ==182:
        print('right')
        car.motor_right(speed)
    elif which_key &0xFF ==181:
        car.motor_stop()
        print('stop')
    elif which_key &0xFF ==ord('q'):
        car.motor_stop()
        print('exit')
        is_exit =True
    elif which_key &0xFF ==ord('e'):
```

```
        enable_linetracing =True
        print('Line tracing enabled:', enable_linetracing)
    elif which_key &0xFF ==ord('w'):
        enable_linetracing =False
        car.motor_stop()
        print('Line tracing disabled:', enable_linetracing)

    return is_exit
def cal_moment(img, i, j):
    M =0
    for a in range(img.shape[0]):
        for b in range(img.shape[1]):
            M +=(a)**j*(b)**i*img[a,b]
    return M
img =cv.imread("line1.png", cv.IMREAD_GRAYSCALE)
if img is not None:
    print('img shape : ', img.shape)
    cv.imshow("img", img)
    y, x =img.shape
    print('x={}, y={}'.format(x,y))
    y_u =int(y/3)
    img_seg1 =img[:y_u,:]
    img_seg2 =img[y_u:y_u*2,:]
    img_seg3 =img[y_u*2:y_u*3,:]
    cv.imshow("img_seg1", img_seg1)
    cv.imshow("img_seg2", img_seg2)
    cv.imshow("img_seg3", img_seg3)
    sum_val =0
       M_seg1   =[cal_moment(img_seg1,   0,   0),   cal_moment(img_seg1,   1,
0)/cal_moment(img_seg1, 0, 0)\
, cal_moment(img_seg1, 0, 1)/cal_moment(img_seg1, 0, 0)]
       M_seg2   =[cal_moment(img_seg2,   0,   0),   cal_moment(img_seg2,   1,
0)/cal_moment(img_seg2, 0, 0)\
, cal_moment(img_seg2, 0, 1)/cal_moment(img_seg2, 0, 0)]
       M_seg3   =[cal_moment(img_seg3,   0,   0),   cal_moment(img_seg3,   1,
0)/cal_moment(img_seg3, 0, 0)\
, cal_moment(img_seg3, 0, 1)/cal_moment(img_seg3, 0, 0)]

    print('M_seg1', M_seg1)
```

```
    print('M_seg2', M_seg2)
    print('M_seg3', M_seg3)
    cv.waitKey(20)
    cv.destroyAllWindows()
def show_grid(img):
    h, _, _=img.shape
    for x in v_x_grid:
        #print('show grid', x)
        cv.line(img, (x, 0), (x, h), (0,255,0), 1, cv.LINE_4)
def line_tracing(cx):
    global moment
    global v_x
    global enable_linetracing  # 전역 변수 선언
    tolerance =0.1
    diff =0
    if not enable_linetracing:
        return  # 자율주행 비활성화 시 동작 중지
    if moment[0] !=0 and moment[1] !=0 and moment[2] !=0:
        avg_m =np.mean(moment)
        diff =np.abs(avg_m -cx) /v_x
    print('diff ={:.4f}'.format(diff))
    if diff <=tolerance:
        moment[0] =moment[1]
        moment[1] =moment[2]
        moment[2] =cx
        if v_x_grid[grid] <=cx <v_x_grid[grid+2]:
            car.motor_go(speed)
            print('go')
        elif cx <v_x_grid[grid]:
            car.motor_left(speed)
            car.motor_left(speed)
            print('turn left')
        elif cx >v_x_grid[grid+2]:
            car.motor_right(speed)
            car.motor_right(speed)
            print('turn right')
    else:
        car.motor_go(speed)
        print('default go')
```

```
        moment =[0, 0, 0]


def detect_maskY_BGR(frame):
    B =frame[:,:,0]
    G =frame[:,:,1]
    R =frame[:,:,2]
    Y =np.zeros_like(G, np.uint8)
    # need to tune params
    Y =G*0.5 +R*0.5 -B*0.7 # 연산 수행 시 float 64로 바뀜
    Y =Y.astype(np.uint8)
    Y =cv.GaussianBlur(Y, (5,5), cv.BORDER_DEFAULT)
    # need to tune params
    _, mask_Y =cv.threshold(Y, 100, 255, cv.THRESH_BINARY)
    return mask_Y
def main():
    global enable_linetracing  # 전역 변수 선언
    camera =cv.VideoCapture(0)
    camera.set(cv.CAP_PROP_FRAME_WIDTH, v_x)
    camera.set(cv.CAP_PROP_FRAME_HEIGHT, v_y)
    try:
        while camera.isOpened():
            ret, frame =camera.read()
            frame =cv.flip(frame, -1)
            cv.imshow('camera', frame)
            crop_img =frame[180:, :]
            maskY =detect_maskY_BGR(crop_img)
                        contours,   _=cv.findContours(maskY,   cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
            if len(contours) >0:
                c =max(contours, key=cv.contourArea)
                m =cv.moments(c)

                epsilon =1
                cx =int(m['m10']/(m['m00']+epsilon))
                cy =int(m['m01']/(m['m00']+epsilon))
                cv.circle(crop_img, (cx, cy), 3, (0, 0, 255), -1)
                cv.drawContours(crop_img, contours, -1, (0, 255, 0), 3)
                                cv.putText(crop_img,   str(cx),   (10,   10),
```

```
cv.FONT_HERSHEY_DUPLEX, 0.5, (0, 255, 0))
                if enable_linetracing:
                    line_tracing(cx)
                show_grid(crop_img)
                   cv.imshow('crop_img', cv.resize(crop_img, dsize=(0, 0),
fx=2, fy=2))
            which_key =cv.waitKey(20)
            if which_key >0:
                is_exit =key_cmd(which_key)
                if is_exit:
                    break
    except Exception as e:
        print(e)
    finally:
        camera.release()
        cv.destroyAllWindows()
        global is_running
        is_running =False
if __name__=='__main__':
    v_x =320
    v_y =240
    v_x_grid =[int(v_x*i/10) for i in range(1, 10)]
    moment =np.array([0, 0, 0])
    print(v_x_grid)
    t_task1 =threading.Thread(target=func_thread)
    t_task1.start()
    car =SDcar.Drive()
    is_running =True
    enable_linetracing =False
    main()
    is_running =False
    car.clean_GPIO()
    print('end of program')
```

Motor 제어 코드

```
import threading
import time
import RPi.GPIO as GPIO
class Drive:
    def __init__(self):
                                                                self.pins
={"SW1":5,"SW2":6,"SW3":13,"SW4":19,"PWMA":18,"AIN1":22,"AIN2":27,"PWMB":
23,"BIN1":25,"BIN2":24}
        self.config_GPIO()
        self.L_Motor =GPIO.PWM(self.pins["PWMA"],500)
        self.L_Motor.start(0)
        self.R_Motor =GPIO.PWM(self.pins["PWMB"],500)
        self.R_Motor.start(0)
    def config_GPIO(self):
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.pins["SW1"],GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(self.pins["SW2"],GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(self.pins["SW3"],GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(self.pins["SW4"],GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(self.pins["PWMA"],GPIO.OUT)
        GPIO.setup(self.pins["AIN1"],GPIO.OUT)
        GPIO.setup(self.pins["AIN2"],GPIO.OUT)
        GPIO.setup(self.pins["PWMB"],GPIO.OUT)
        GPIO.setup(self.pins["BIN1"],GPIO.OUT)
        GPIO.setup(self.pins["BIN2"],GPIO.OUT)
    def clean_GPIO(self):
        GPIO.cleanup()

    def motor_go(self, speed):
        GPIO.output(self.pins["AIN1"],0)
        GPIO.output(self.pins["AIN2"],1)
        self.L_Motor.ChangeDutyCycle(speed)
        GPIO.output(self.pins["BIN1"],0)
        GPIO.output(self.pins["BIN2"],1)
        self.R_Motor.ChangeDutyCycle(speed)
    def motor_back(self, speed):
```

```
        GPIO.output(self.pins["AIN1"],1)
        GPIO.output(self.pins["AIN2"],0)
        self.L_Motor.ChangeDutyCycle(speed)
        GPIO.output(self.pins["BIN1"],1)
        GPIO.output(self.pins["BIN2"],0)
        self.R_Motor.ChangeDutyCycle(speed)

    def motor_left(self, speed):
        GPIO.output(self.pins["AIN1"],1)
        GPIO.output(self.pins["AIN2"],1)
        self.L_Motor.ChangeDutyCycle(speed)
        GPIO.output(self.pins["BIN1"],0)
        GPIO.output(self.pins["BIN2"],1)
        self.R_Motor.ChangeDutyCycle(speed)

    def motor_right(self, speed):
        GPIO.output(self.pins["AIN1"],0)
        GPIO.output(self.pins["AIN2"],1)
        self.L_Motor.ChangeDutyCycle(speed)
        GPIO.output(self.pins["BIN1"],1)
        GPIO.output(self.pins["BIN2"],1)
        self.R_Motor.ChangeDutyCycle(speed)
    def motor_stop(self):
        GPIO.output(self.pins["AIN1"],0)
        GPIO.output(self.pins["AIN2"],1)
        self.L_Motor.ChangeDutyCycle(0)
        GPIO.output(self.pins["BIN1"],0)
        GPIO.output(self.pins["BIN2"],1)
        self.R_Motor.ChangeDutyCycle(0)
if __name__=='__main__':
    drive =Drive()
    drive.motor_go(100)
    time.sleep(2)
    drive.motor_left(100)
    time.sleep(2)
    drive.motor_right(100)
    time.sleep(2)
    drive.motor_back(100)
    time.sleep(2)
```

```
    drive.clean_GPIO()
```