소규모
프로젝트

OPENCV를 이용한
차선감지 자율주행 RC카

김규현 김희수 조영재

목차

개요

OPENCV로 차선 Detection을 수행하는 라즈베리파이와

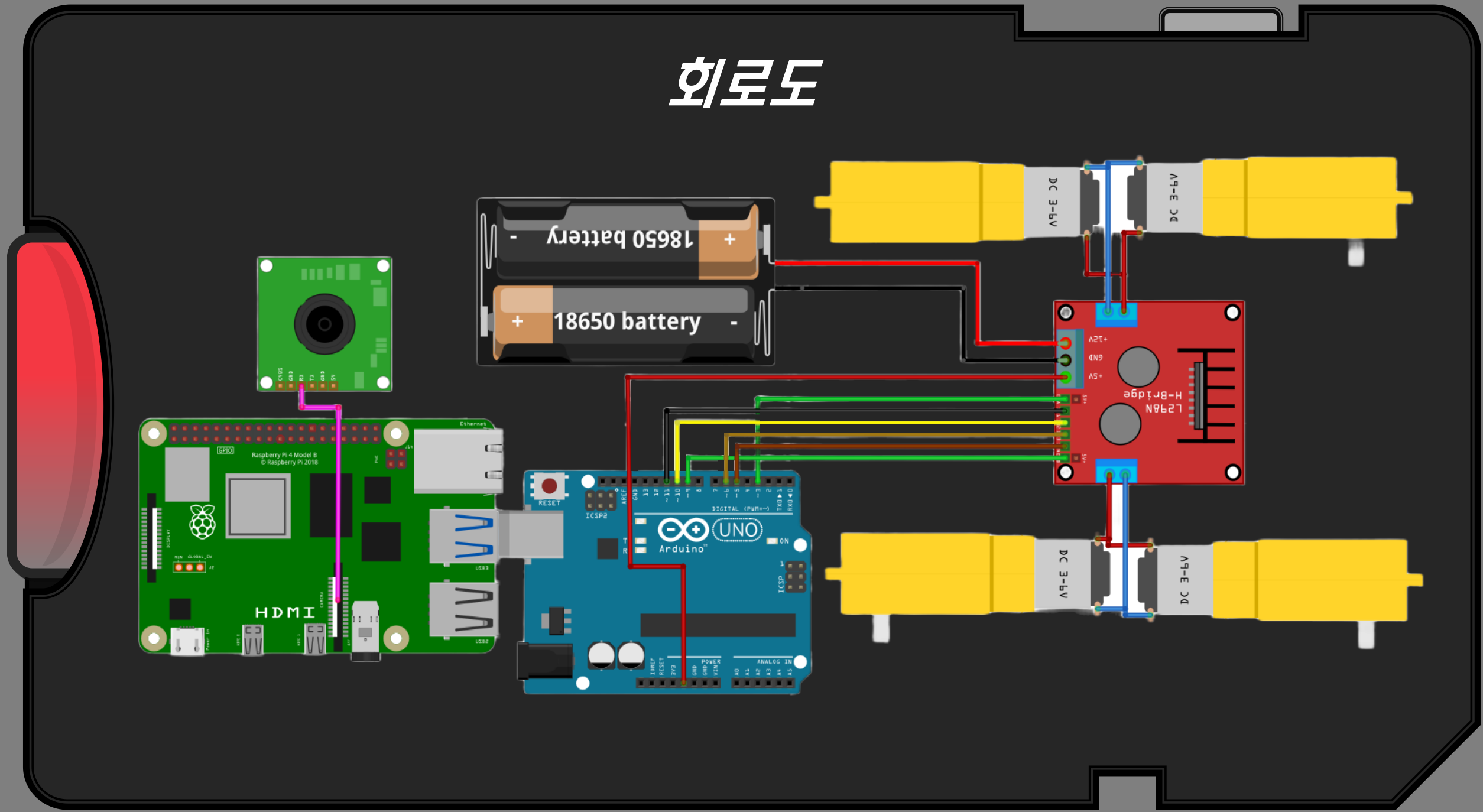
L298N 모듈 사용해 속도와 방향을 제어하는 아두이노를 이용한 자율주행 RC카

활용장비 및 재료

Raspberry Pi 4 Model B, 카메라 모듈 V2

Arduino Uno, L298N

회로도


18650 battery
18650 battery
DC 3-6V
DC 3-6V
DC 3-6V
DC 3-6V
L298N
H-Bridge
+12V
GND
+5V
Raspberry Pi 4 Model B
© Raspberry Pi 2018
HDMI
UNO
Arduino
RESET
ICSP2
ICSP
POWER
ANALOG IN
DIGITAL (PWM=~)

라즈베리파이 코드

```python
import cv2
import numpy as np
import serial
import time

ser = serial.Serial('/dev/ttyACM0', 9600)

trap_top_width_p1 = 0.45
trap_top_width_p2 = 0.55
trap_bottom_width_p1 = 0.0
trap_bottom_width_p2 = 1.0
trap_height_p1 = 0.6
trap_height_p2 = 1.0

rho = 10
theta = 1 * np.pi / 180
threshold = 50
min_line_length = 10
max_line_gap = 30
```

Serial.Serial('/dev/ttyACM0')는

라즈베리파이와 아두이노 USB 시리얼 통신코드

```python
def draw_lines(img, lines, color=[0, 255, 0], thickness=12):
    # 예외처리
    if (lines is None) or (len(lines) == 0):
        return

    # 왼쪽, 오른쪽 라인을 그릴것인지 체크
    left_draw_check = True
    right_draw_check = True

    # 모든선(lines)의 기울기를 체크해서 불필요한 선 제거
    # 기울기 임계값 보다 기울기가 작은 선은 제거
    slope_threshold = 0.5  # 기울기 30도
    slopes = []
    new_lines = []

    # 기울기가 작은 라인 제거
    # lines 변수는 (N, 1, 4)차원 형태를 가짐.
    for line in lines:
        x1, y1, x2, y2 = line[0]  # line = [[x1,y1,x2,y2]]
        # 기울기 계산
        if x2 - x1 == 0.:
            slope = 999
        else:
            slope = (y2 - y1) / (x2 - x1)
        if abs(slope) > slope_threshold:
            slopes.append(slope)
            new_lines.append(line)

    lines = new_lines  # 조건을 만족하는 line만 걸러냄
```

def draw_lines():

Hough 알고리즘으로 추출해낸

여러가지 Edge들을 확률적으로

추출 후 이미지를 절반으로 나누어

두 개의 선으로 도출하는 코드

```
### 오른쪽 / 왼쪽 라인 분리
# 기울기 및 선을 구성하는 두점이 영상의 가운데를 기준으로 좌우에 분포하는지 체크
left_lines = []
right_lines = []

for i, line in enumerate(lines):
    x1, y1, x2, y2 = line[0]  # line = [[x1,y1,x2,y2]]
    img_center = img.shape[1] / 2  # width
    # 기울기 방향이 바뀜 : y의 좌표가 위에서 아래로 내려옴

    if slopes[i] > 0 and x1 > img_center and x2 > img_center:  # right
        right_lines.append(line)
    elif slopes[i] < 0 and x1 < img_center and x2 < img_center:  # left
        left_lines.append(line)

## LEFT / RIGHT 라인을 구성하는 점들을 사용해서 np.polyfit을 적용
# np.polyfit에 사용될 점으로 추가

# LEFT 찾기
left_lines_x = []
left_lines_y = []

for line in left_lines:
    x1, y1, x2, y2 = line[0]  # line = [[x1,y1,x2,y2]]
    left_lines_x.append(x1)
    left_lines_x.append(x2)
    left_lines_y.append(y1)
    left_lines_y.append(y2)

if len(left_lines_x) > 0:
    left_m, left_b = np.polyfit(left_lines_x, left_lines_y, 1)  # y = m*x + b
else:
    left_m, left_b = 1, 1
    left_draw_check = False
```

- 영상 가운데를 기준으로 절반으로 나누어
  왼쪽은 left_lines, 오른쪽은 right_lines 로 지정
- Left에 속해 있는 (x, y)좌표를 left_lines 리스트에 추가

```
right_lines_x = []
right_lines_y = []

for line in right_lines:
    x1, y1, x2, y2 = line[0]
    right_lines_x.append(x1)
    right_lines_x.append(x2)
    right_lines_y.append(y1)
    right_lines_y.append(y2)

if len(right_lines_x) > 0:
    right_m, right_b = np.polyfit(right_lines_x, right_lines_y, 1)
else:
    right_m, right_b = 1, 1
    right_draw_check = False

y1 = int(img.shape[0])
y2 = int(img.shape[0] * trap_height_p1)

right_x1 = int((y1 - right_b) / right_m)
right_x2 = int((y2 - right_b) / right_m)
left_x1 = int((y1 - left_b) / left_m)
left_x2 = int((y2 - left_b) / left_m)

if right_draw_check:
    cv2.line(img, (right_x1, y1), (right_x2, y2), color, thickness)
if left_draw_check:
    cv2.line(img, (left_x1, y1), (left_x2, y2), color, thickness)
```

- Left에 속해 있는 (x, y)좌표를 right_lines 리스트에 추가

✓ np.polyfit에 의해 출력되는 값은 기울기와 y절편

left_lines, right_lines 에 할당된 기울기 값과 y절편값을 이용해서 (X, Y) 좌표 값을 연결해 선을 그림

```python
if abs(left_x1) ≤ 100 or abs(right_x1) ≥ 500:
    if left_x1 > 0 or right_x1 > 0:
        if left_x1 < 10 and right_x1 > 500:
            # print('left')
            ser.write(b'a')

        # elif left_x1 > 100 and right_x1 > 450:
        elif left_x1 > 50 and right_x1 > 450:
            # print('right')
            ser.write(b'd')

        else:
            # print('go')
            ser.write(b'w')
```

- 앞서 도출한 left_x1, right_x1의 좌표를 기준으로 주행방향을 아두이노로 전송하는 코드

```
capture = cv2.VideoCapture(0)

capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
codec = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')  # .mp4

width = int(capture.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(capture.get(cv2.CAP_PROP_FPS))

pts = np.array([[int(width * trap_bottom_width_p1), int(height * trap_height_p2)],
                [int(width * trap_top_width_p1), int(height * trap_height_p1)],
                [int(width * trap_top_width_p2), int(height * trap_height_p1)],
                [int(width * trap_bottom_width_p2), int(height * trap_height_p2)]], dtype=np.int32)

lower_black = (0, 0, 0)
upper_black = (0, 3, 3)

img_mask = np.zeros((height, width), dtype=np.uint8)
img_mask = cv2.fillPoly(img_mask, [pts], 255)
```

cv2.Fillpoly()를 통해 관심영역을 설정

```python
while True:
    ret, frame = capture.read()
    if ret == False:
        print('동영상 종료')
        break
    frame = cv2.flip(frame, -1)
    frame_bgr = frame.copy()
    mask_black = cv2.inRange(frame_bgr, lower_black, upper_black)

    # mask_lane의 성능을 개선하기 위해서 모폴로지 적용
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    mask_lane = cv2.morphologyEx(mask_black, cv2.MORPH_CLOSE, kernel, iterations=2)  # mask_lane → (720,1280,1)

    # 외각선 구하기 : Canny엣지사용
    frame_canny = cv2.Canny(mask_lane, 50, 150)
    # frame_canny = cv2.bitwise_and(frame_canny, frame_canny, mask=img_mask)

    # 외곽선을 기준으로 선을 추출
    lines = cv2.HoughLinesP(frame_canny, rho, theta, threshold, minLineLength=min_line_length, maxLineGap=max_line_gap)
    draw_lines(frame_bgr, lines)

    cv2.polylines(frame_bgr, [pts], True, (255, 0, 0), 1)
    ################################
    cv2.imshow('dst', frame_bgr)
    ################################

    key = cv2.waitKey(25)
    if key == 27:  # Esc키
        break;

capture.release()
cv2.destroyAllWindows()
```

While문을 이용해서 영상출력

1. 커널을 만들어서 모폴로지화 (노이즈 제거)

2. Canny Edge detecting을 통해서 외곽선을 추출

3. Hough 변환을 통하여 Canny Edge 에서 검출한 외곽선 중 상대적으로 확률이 높은 선에 대해서만 추출을 함

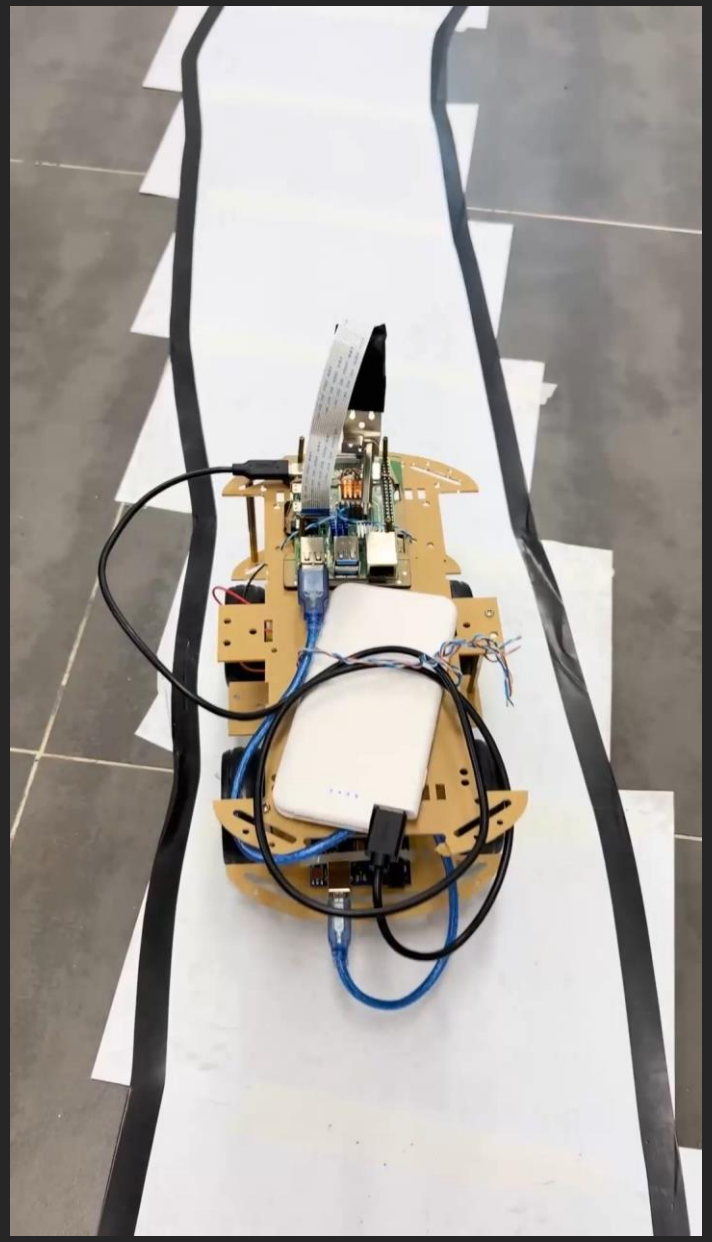
4. Hough변환으로 추출된 선들 중 두개만 뽑아내는 함수가 draw_lines()

아두이노 코드

```
#define EN1 11 // 좌 전진
#define EN2 10 // 좌 후진
#define EN3 6 //우 전진
#define EN4 5 //우 후진
#define ENAPin 3
#define ENBPin 9

#include <SoftwareSerial.h>

void setup() {
  Serial.begin(9600);
  pinMode(EN1,OUTPUT);
  pinMode(EN2,OUTPUT);
  pinMode(EN3,OUTPUT);
  pinMode(EN4,OUTPUT);
  pinMode(ENAPin, OUTPUT);
  pinMode(ENBPin, OUTPUT);
}

void loop() {
  while (Serial.available() > 0) {
    int cmd = Serial.read();

    switch (cmd) {
      case 'w':
        moveForward();
        break;
      case 'a':
        turnLeft();
        break;
      case 'd':
        turnRight();
        break;
    }
  }
}
```

```
void moveForward() {
  analogWrite(ENAPin, 150);
  analogWrite(ENBPin, 150);
  digitalWrite(EN1, HIGH);
  digitalWrite(EN2, LOW);
  digitalWrite(EN3, HIGH);
  digitalWrite(EN4, LOW);

}

void turnLeft() {
  analogWrite(ENAPin, 150);
  analogWrite(ENBPin, 150);
  digitalWrite(EN1, LOW);
  digitalWrite(EN2, HIGH);
  digitalWrite(EN3, HIGH);
  digitalWrite(EN4, LOW);

}

void turnRight() {
  analogWrite(ENAPin, 150);
  analogWrite(ENBPin, 150);
  digitalWrite(EN1, HIGH);
  digitalWrite(EN2, LOW);
  digitalWrite(EN3, LOW);
  digitalWrite(EN4, HIGH);

}
```

주행영상



차선감지 주행영상

감사합니다