[2019] 임베디드시스템 설계

자율주행 자동차

보고서

- 28조 -

컴퓨터과학부 2014920015 박승록
컴퓨터과학부 2014920030 육영훈

1. 개요

라즈베리파이와 아두이노 보드를 사용하여 자율주행 자동차를 만든다. 라즈베리파이에는 파이카메라와 보조배터리를 부착하고 아두이노에는 건전지가 부착된 모터 드라이버 쉴드 L293D를 연결한다. 라즈베리파이는 카메라로 사진을 찍어 방향과 속도를 결정하고, 이를 USB 케이블을 통해 아두이노에게 Serial 통신 방법으로 전달한다. 아두이노는 명령 문자열을 받아서 모터 드라이버 쉴드를 통해 모터를 제어하고, 그 결과로 자율주행 자동차가 움직이게 된다.

2. 동작 원리

2.1 모터 제어 코드 (아두이노)

```
#include <SoftwareSerial.h>
#include <AFMotor.h>

#define L_BASE 200  // left motor base speed
#define R_BASE 200  // right motor base speed

AF_DCMotor motor_L(3);  // left motor
AF_DCMotor motor_R(4);  // right motor

String Speed;
char  LorR;
int  i, s;
int  v1, v2;
char DataToRead[6];


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  motor_L.run(RELEASE); // motor stop
  motor_R.run(RELEASE);
}

void loop() {
  DataToRead[5] = '\n';
  Serial.readBytesUntil(char(13), DataToRead, 5);

/* For Debugging, send string to RPi */
  for (i = 0; i < 6; i++) {
    Serial.write(DataToRead[i]);
    if (DataToRead[i] == '\n') break;
  }
/* End of Debugging */
  LorR = DataToRead[0];
  Speed = "";
  for (i = 1; ((DataToRead[i] != '\n') || (DataToRead[i] != '\r')) && (i < 6); i++) {
    Speed += DataToRead[i];
  }
  if(Speed.length() > 0 && (LorR == 'L' || LorR == 'R')){
    s = Speed.toInt();
    //Serial.println(LorR);
    //Serial.print("s= ");
    //Serial.println(s);
    if (LorR == 'L') {
      // Turn left wheel with speed s
      if(s<=5) {
        v1 = L_BASE;
        v2 = R_BASE + 10*s;
      }
      else if(s<10) {
        v1 = L_BASE - 5*s;
        v2 = R_BASE + 5*s;
```

```
      }
      else {
        v1 = 0;
        v2 = R_BASE + 3*s;
      }

      motor_L.setSpeed(v1);
      motor_R.setSpeed(v2);
      motor_L.run(FORWARD); // motor run
      motor_R.run(FORWARD);
    }
    else if (LorR == 'R') {
      // Turn right wheel with speed s
      if(s<=5) {
        v1 = L_BASE + 10*s;
        v2 = R_BASE;
      }
      else if(s<10) {
        v1 = L_BASE + 5*s;
        v2 = R_BASE - 5*s;
      }
      else {
        v1 = L_BASE + 3*s;
        v2 = 0;
      }
      motor_L.setSpeed(v1);
      motor_R.setSpeed(v2);
      motor_L.run(FORWARD); // motor run
      motor_R.run(FORWARD);
    }
  }

  delay(100);
  motor_L.run(RELEASE); // motor stop
  motor_R.run(RELEASE);
  delay(200);
}

```

라즈베리파이에서 입력받은 명령을 수행하는 아두이노 코드이다. 예를 들어 “R30\n”라는 문자열을 받으면 아두이노는 오른쪽 방향으로 정수 값 30의 속도로 전진한다고 판단한다. 사용한 키트의 모터는 setSpeed()의 parameter 값으로 약 120에서 255까지 동작한다. 120보다 더 낮은 값은 모터의 힘이 부족한 지 바퀴가 잘 굴러가지 않는다.

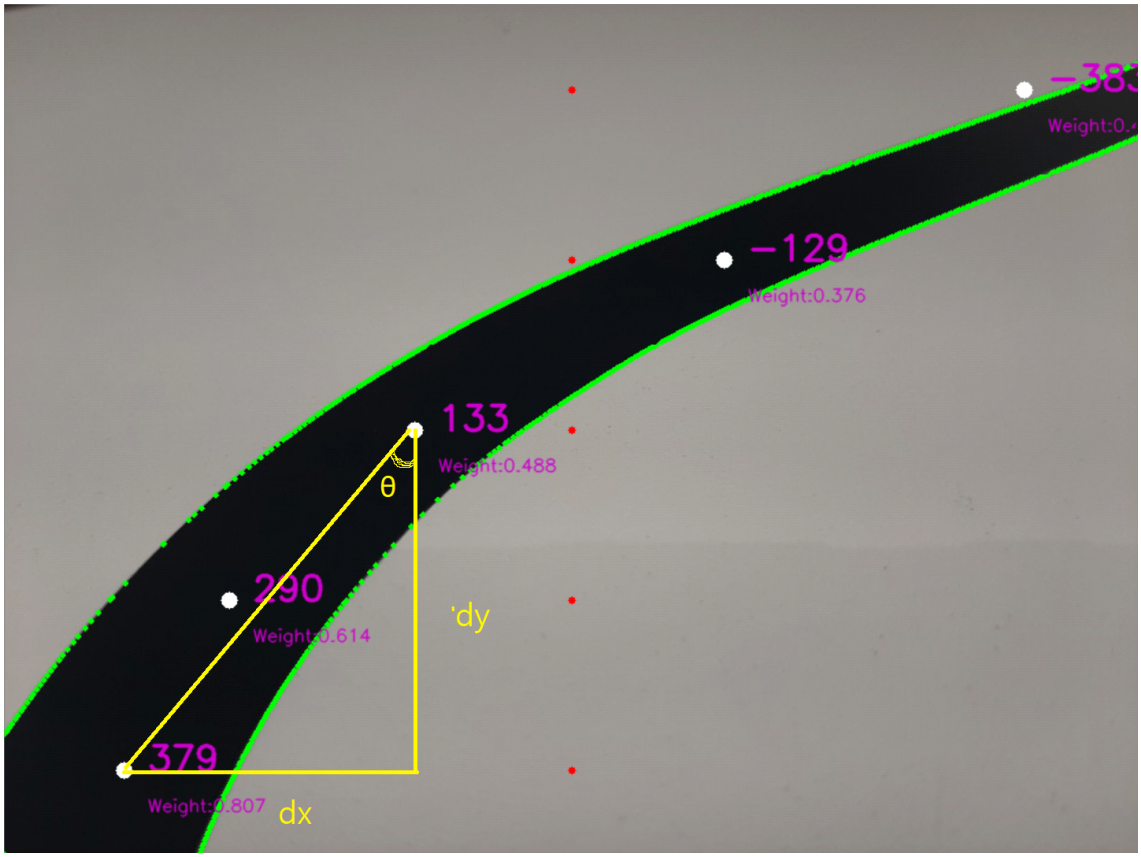
모터 방향을 결정하는 LorR과 속도 s에 따라 왼쪽 바퀴 속도 v1, 오른쪽 바퀴 속도 v2가 결정된다. 좌회전은 오른쪽 바퀴 속도를 좀 더 주고 우회전은 왼쪽 바퀴 속도를 좀 더 주는 식이다. s가 10보다 커지면 한쪽 바퀴 속도를 0으로 두어 회전률을 높였다.

2.2 판단 알고리즘 (Rpi)

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

from Image import *
from Utils import *

from picamera import PiCamera
from picamera.array import PiRGBArray

N_SLICES = 5  # number of image slices
IMAGE_WIDTH = 1088  #image width
IMAGE_HEIGHT = 608  #image height

CROP_HEIGHT = int(IMAGE_HEIGHT/N_SLICES) # crop height

#num = 1 # image number

WEIGHT_CONSTANT = 0.6  # ratio of weight

ser = serial.Serial('/dev/ttyUSB0', 9600)   # serial
if ser is None:
    ser = serial.Serial('/dev/ttyUSB1', 9600)
```

카메라로 캡처한 이미지를 5등분으로 나눈다. CROP_HEIGHT는 5등분된 이미지의 높이이다. WEIGHT_CONSTANT는 후에 설명할 상수 값이다.

```python
def main():

    ser = serial.Serial('/dev/ttyUSB0', 9600)
    if ser is None:
        ser = serial.Serial('/dev/ttyUSB1', 9600)

    camera = PiCamera()
    camera.resolution = (IMAGE_WIDTH, IMAGE_HEIGHT)
    rawCapture = PiRGBArray(camera, size=(IMAGE_WIDTH, IMAGE_HEIGHT))

    camera.start_preview(fullscreen = False, window = (600, 200, IMAGE_WIDTH, IMAGE_HEIGHT)) # show current camera
    time.sleep(0.1) # sleep for loading camera


    while True:
        try:
            camera.capture(rawCapture, format="bgr", use_video_port=True)   # capture
            Images = []
            for _ in range(N_SLICES):
                Images.append(Image())
            frame = rawCapture.array

            points = SlicePart(frame, Images, N_SLICES) # crop image and do PROCESS

            topPoint = points[2][0]     # x of third crop_image
            botPoint = points[-1][0]    # x of fifth crop_image
            dx = topPoint - botPoint
            dy = 2*CROP_HEIGHT
                                        # LINE : line for 3rd_image center and 5th_image center
            theta = math.atan2(dy, dx)  # angle for vertical line and LINE
            angle = theta - math.pi/2
            max_angle = math.atan(IMAGE_WIDTH/CROP_HEIGHT/2)
            curveRatio = angle/max_angle    # ratio of curve

            diffRatio = Images[-1].diff/IMAGE_WIDTH/2 # difference from center
            v_change = (1-WEIGHT_CONSTANT)*curveRatio + WEIGHT_CONSTANT*diffRatio # velocity change

            print(v_change)
            if v_change > 0:  # turn left
                left(v_change)

            else:             # turn right
                right(-v_change)

            rawCapture.truncate(0) # image clear

        except KeyboardInterrupt:
            print("Keyboard Interrupt")
            camera.close()
            cv2.destroyAllWindows()
            break;
```



[그림 1] Process 처리 이후의 image

매 루프마다 파이 카메라가 사진을 캡처한다. 캡처한 이미지를 SlicePart 함수가 5등분으로 자르고 도로 윤곽선, 도로의 무게중심 등을 표시한다. topPoint와 botPoint는 각각 위에서부터 3번째, 5번째 이미지의 도로 무게중심의 x좌표이다. 이미지가 중심으로부터 떨어진 정도를 dx라고 정의한다. dx>0이면 중심으로부터 왼쪽으로, dx<0이면 중심으로부터 오른쪽으로 치우쳐 있다. dy는 세 번째와 다섯 번째 이미지의 무게중심의 y축 차이이다.

세 번째 무게중심과 다섯 번째 무게중심을 지나는 직선을 LINE이라고 정의하자. 수직선과 LINE이 이루는 각을 theta라고 할 때, 삼각형에서 반대쪽 각이 angle이다. 이 때 angle이 가질 수 있는 최댓값 max_angle과 angle의 비율이 도로가 굽은 정도를 나타내는 curveRatio이다. 중심에서 떨어진 정도를 diffRatio라 할 때, 위에서 정의한 WEIGHT_CONSTANT를 사용하게 되는데 중심에서 떨어진 정도가 도로가 굽은 정도보다 더 중요하게 여겨 0.6으로 주었다. 따라서 계산식을 통해 속도 변화량 v_change를 구할 수 있다.

v_change > 0이면 좌회전, v_change < 0이면 우회전으로 동작한다.

```
27  def cal(c): # coefficient for speed
28      return c*55
29
30  def left(c): # turn left
31      v = int(cal(c))
32      cmd = ("L%d\n" % v).encode("ascii")
33      print("[SEND] %s" % cmd)
34      ser.write(cmd)  # transmit to Arduino
35      try:
36          read_serial = ser.readline()
37          print("[READ] %s" % read_serial) # debug for serial
38      except:
39          print("readline exception")
40
41
42  def right(c): # turn right
43      v = int(cal(c))
44      cmd = ("R%d\n" % v).encode("ascii")
45      print("[SEND] %s" % cmd)
46      ser.write(cmd)  # transmit to Arduino
47      try:
48          read_serial = ser.readline()
49          print("[READ] %s" % read_serial) # debug for serial
50      except:
51          print("readline exception")
52
```

parameter로 전달된 v_change 값에 따라 아두이노로 명령어를 전달한다. L_BASE와 R_BASE를 200으로 설정했기 때문에 최대 속도 255를 넘지 않는 계수 값을 반환하는 cal(c)를 사용하였다. 오버플로우가 일어나면 모터가 동작하지 않는다.

3. 고찰

처음에 OpenCV를 설치하는 것부터 난관이었다. OpenCV 3.3버전을 설치하려 했으나 실패해서 4.1.2버전을 4시간을 걸쳐 힘들게 설치하게 되었다. OpenCV를 설치하고 python에서 잘 작동하는 것까지 확인한 후 아두이노로 모터제어 시뮬레이션을 해보는데, 양쪽 모터의 속도가 달라서 속도 차이를 조절하느라 힘들었다. 나중에 밝혀진 사실이지만, 모터를 사용하지 않을 때 건전지 하나만 빼놔서 나머지 건전지는 대기전력으로 소모돼 일어난 일이었다.

파이카메라 구동에서도 몇 가지 시행착오를 겪었다. 우리가 사용한 파이카메라에는 필름이 붙어있었는데, 떼지 않고 사용했는데 해상도가 매우 나빠서 process 처리 함수가 제대로 작동하지 않았다. 한 번 떼 봤는데 잘 됐다. 카메라 위치 조정과 고정 부분에서도 시간을 많이 들인 것 같다.

모니터에 HDMI로 라즈베리파이를 연결하면서 작업하는 게 힘들어서 편한 작업환경을 가지기 위해 라즈베리파이에 아두이노 프로그램을 설치하고 노트북에 원격으로 라즈베리파이 환경을 제어하는 vnc viewer를 사용했다.

Serial 통신에서도 문제가 많았다. 아두이노 시리얼 모니터에서 확인해본 결과 speed를 나타내는 정수 값 s에 제대로 된 값이 들어가지 않는 경우가 많아서 디버그하는데 어려움이 있었다.

자율주행 자동차 주행 코드를 작동하면서 처리된 image를 화면에 출력하는데 필요한 라이브러리 및 함수를 사용하는 데 실패해서 작업 중에 힘들었다.