결과 보고서

임베디드응용및실습.
10주차 실습 과제

과 목 명	임베디드응용및실습
학 번	2019161068
이 름	백승진
제 출 일	2024년 11월 13일

1. OpenCV를 사용하여 라즈베리파이 카메라에서 받은 실시간 영상으로 얼굴을 검출하시오.

- 얼굴 검출 시 사각형 박스가 표시되도록 함
- 소스 코드 및 나의 얼굴 검출 영상 제출

1) 코드

```
import cv2
import numpy as np

# Face cascade classifier 초기화
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

# 웹캠 초기화
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

while True:
    # 프레임 읽기
    ret, frame = cap.read()

    # 프레임 상하좌우 반전
    frame = cv2.flip(frame, -1)

    # 그레이스케일 변환
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 얼굴 검출
    faces = face_cascade.detectMultiScale(gray)

    # 검출된 얼굴 수 표시
    cv2.putText(frame, f'Faces detected: {len(faces)}',
                (10, 30), cv2.FONT_HERSHEY_DUPLEX, 0.3, (0, 255, 0), 2)

    # 각 얼굴에 대해 사각형 검출
    for (x, y, w, h) in faces:
        # 얼굴 사각형 그리기
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

    # 결과 표시
    cv2.imshow('Face Detection', frame)

    # ESC 키를 누르면 종료
    if cv2.waitKey(1) & 0xFF == 27:
        break

# 자원 해제
cap.release()
cv2.destroyAllWindows()
```

2) 결과

영상 참조

3) 해석

이번 얼굴 검출 과제에서는 harr cascade 분류기를 사용한다. 이를 위해 먼저 cv2의 해당 분류기를 초기화한다. 다음으로 웹캠도 초기 설정을 진행한다.

다음은 본격적으로 while 문을 이용하여 실시간으로 계속해서 프레임을 읽고, flip 함수를 이용하여 상하좌우 반전시켜준다. 이는 카메라에서 얻은 영상이 상하좌우 반전된 채로 초기 설정되어 있기 때문이다. 다음으로 cascade 분류를 진행하기 위해 해당 이미지를 1채널로 변환한 후, 얼굴을 검출한다. 이때, 검출된 얼굴 수를 왼쪽 위(10, 30)에 표시한다. 다음으로 얼굴의 가로와 세로의 길이를 이용하여 얼굴에 사각형을 그려준다. 사각형을 그렸으면 마지막으로 해당 영상을 실시간으로 출력한다. 이 반복문은 ESC 키를 누르면 종료되도록 작성하였다.

반복문이 종료되면 할당된 자원인 웹캠과 윈도우를 모두 종료한다.

처음에 640, 480 크기의 영상으로 시도하였는데, 프레임의 반응 속도가 느린 것을 확인할 수 있었다. 하드웨어의 한계로 연산 속도가 느리기 때문이며, 이 때문에 320, 240 크기의 영상으로 줄일 수 밖에 없었다.

2. 첨부된 4장의 이미지를 라인 트레이서 용도로 얻었다고 가정하고, 4장의 영상에서 노란색 또는 흰색 선을 추출하여 표기하시오.

- 영상을 표기하는 방법은 자유롭게 함
  - 사각형, 라인 선만 남기고 다 검게 하는 방법 등
- 영상 크기 변경, 크롭, 컬러 변경 등 자유롭게 할 수 있음
- 제안 알고리즘을 4장의 영상에 동일하게 적용 시, 성능이 보장되도록 함
- 소스 코드와 라인 표기된 4장의 영상을 제출

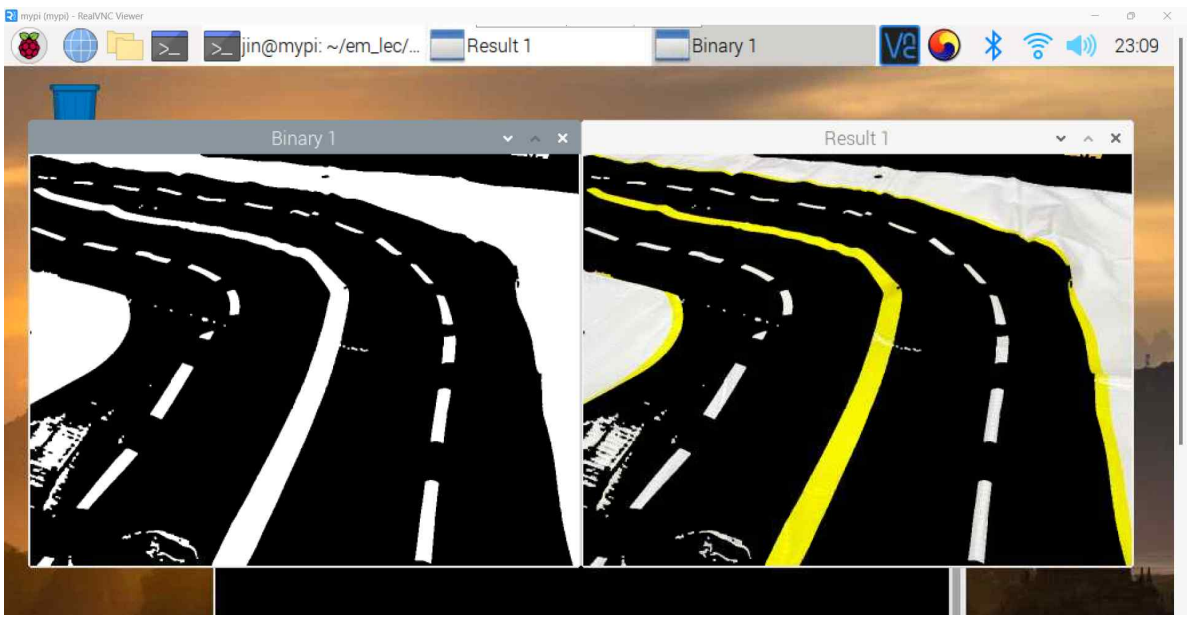
1) 코드

```
import cv2
import numpy as np

def detect_lines(image_path):
    # 이미지 읽기
    image = cv2.imread(image_path)

    # 그레이스케일로 변환
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 이진화
    _, binary = cv2.threshold(gray, 180, 255, cv2.THRESH_BINARY)

    # 모폴로지 연산으로 노이즈 제거
    kernel = np.ones((3, 3), np.uint8)
    binary = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

    # 결과 이미지 생성
    result = image.copy()
    result[binary == 0] = 0

    return result, binary

image_files = [
    '/home/jin/em_lec/week10/assignment/img/img1.jpg',
    '/home/jin/em_lec/week10/assignment/img/img2.jpg',
    '/home/jin/em_lec/week10/assignment/img/img3.jpg',
    '/home/jin/em_lec/week10/assignment/img/img4.jpg'
]

num = 1
for img_path in image_files:
    result, binary = detect_lines(img_path)

    if result is not None and binary is not None:
        width = 480
        ratio = width / result.shape[1]
        dim = (width, int(result.shape[0] * ratio))

        resized_result = cv2.resize(result, dim, interpolation=cv2.INTER_AREA)
```

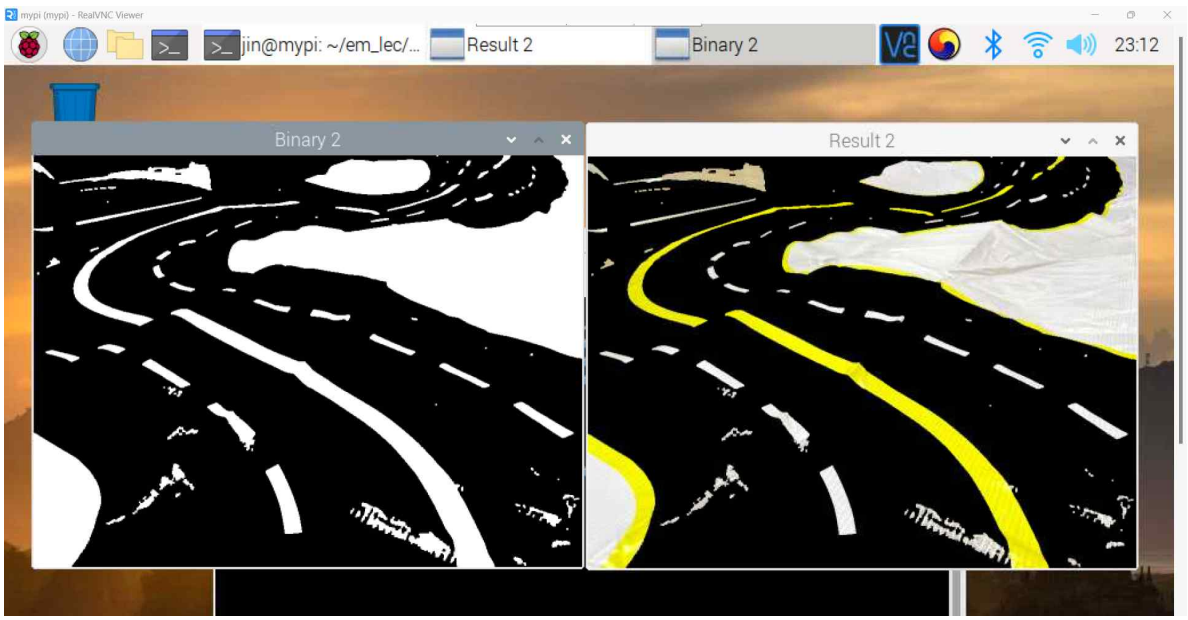
```
        resized_binary = cv2.resize(binary, dim, interpolation=cv2.INTER_AREA)

        cv2.imshow(f'Result {num}', resized_result)
        cv2.imshow(f'Binary {num}', resized_binary)
        cv2.waitKey(0)
    num += 1

cv2.destroyAllWindows()
```

2) 결과

img1

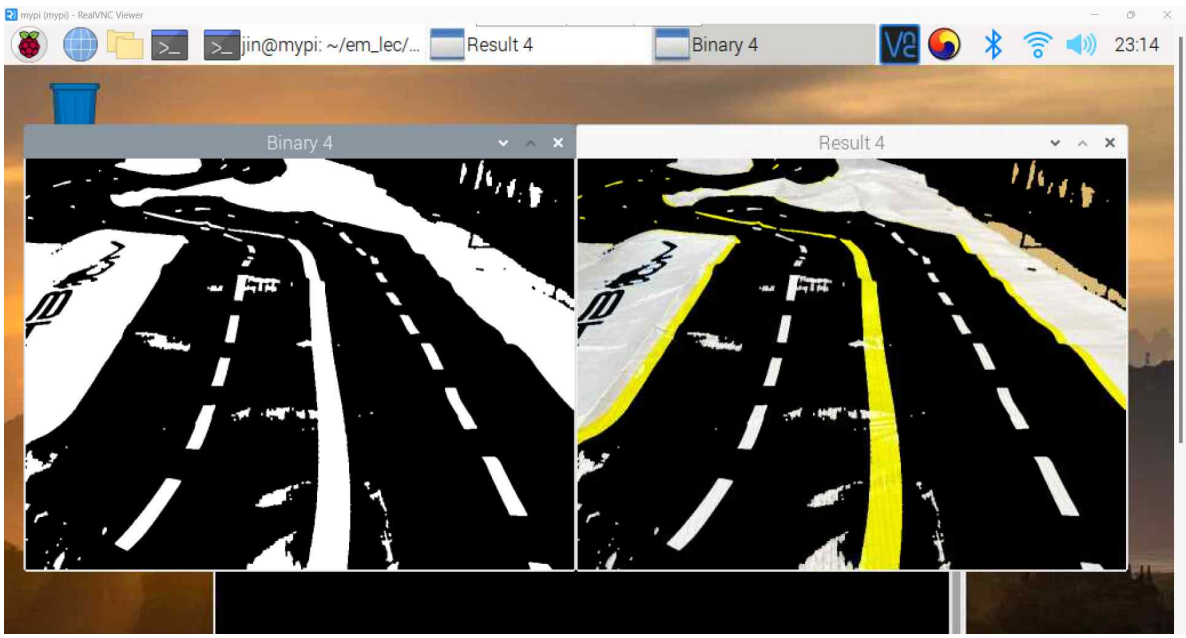


img2



img3

img4



3) 해석

우선 이미지를 읽어 차선을 검출하고, 결과 이미지를 생성하는 detect_lines 함수를 정의한다. 인자는 이미지의 경로이며, 이를 통해 이미지를 읽는다. 이후, 이진화를 위해 단일 채널로 변환한 후 밝기가 180에서 255 사이인 픽셀을 1로, 그 외는 0으로 이진화 처리를 진행한다. 이후 노이즈 제거를 위해 모폴로지 연산을 진행한다. 여기서는 흰색의 노이즈를 지우기 위해 열림 연산을 통해 침식 후에 팽창을 진행하였다. 다음으로 결과 이미지를 생성하여 반환한다.

다음으로 4개의 이미지 파일에 대한 경로를 리스트로 생성한 후, 각 이미지 파일의 차선을

검출하기 위해 반복문을 이용한다.
반복분 내부에서는 바로 각 이미지를 detect_lines 함수를 호출하여 결과를 도출한다. 이때 결과가 반환되는 경우에만 해당 결과 이미지 크기를 재조정한 후 확인할 수 있게 plot하고, 각 이미지를 출력한 후에는 모든 창을 닫는다.
결과를 보면 완벽히 깨끗하게 차선만을 검출해내진 못하고, 다소 노이즈가 보이는데, 이는 이진화의 임곗값을 조정하거나 모폴로지 연산을 더 적절히 사용하여 해결할 수 있을 것이다.