결과 보고서

과제. 10주차 homework

과 목 명	임베디드응용및실습
학 번	2020161024
이 름	김 진 혁
제 출 일	2024년 11월 8일

1-1. OpenCV를 사용하여 라즈베리파이 카메라에서 받은 실시간 영상으로 얼굴 검출

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

1-2. 소스코드

```
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("카메라에서 프레임을 가져올 수 없습니다.")
        break

    flipped_frame = cv2.flip(frame, 0)

    gray = cv2.cvtColor(flipped_frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30))

    for (x, y, w, h) in faces:
        cv2.rectangle(flipped_frame, (x, y), (x + w, y + h), (255, 0, 0), 2)  # 파란색
박스

    cv2.imshow('Face Detection (Flipped)', flipped_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

1-3. 풀이과정

haarcascade_frontalface_default.xml 파일을 사용하여 얼굴을 검출하는 Haar Cascade 분류기를 로드한다. cv2.VideoCapture(0) 으로 카메라와의 연결을 설정한 후 카메라를 통해 영상을 받아온다. flip 함수를 통해 상하 반전 되어 있는 이미지를 상하 반전 시켜주었다. 그레이스케일로 프레임을 바꿔주고 얼굴크기, 최소 검출 얼굴 크기 등을 설정 해준 뒤 for문을 활용해서 검출된 얼굴 위치에 파란색 사각형을 그려주었고 사각형의 두께는 2로 설정하였다. 그리고 마지막에 imshow 함수를 통해 상하 반전된 얼굴 검출 결과를 화면에 표시하였다.

2-1. 첨부된 4장의 이미지를 라인 트레이서 용도로 얻었다고 가정하고, 4장의 영상에서 노란색 또는 흰색선을 추출하여 표기하시오.

- 영상 표기하는 방법은 자유롭게 한다.

- 사각형, 라인, 선만 남기고 다 검게 등등…

- 제안 알고리즘을 4장의 영상에 동일하게 적용 시, 선응이 보장되도록 한다.

- 영상 크기 변경, 크롭, 컬러 변경 등 자유롭게 할 수 있다.

- 소스 코드와 라인 표기된 4장의 영상 제출

2-2. 소스코드

```
import cv2
import numpy as np

yellow_lower = np.array([20, 100, 100], dtype=np.uint8)
yellow_upper = np.array([30, 255, 255], dtype=np.uint8)

white_lower = np.array([0, 0, 200], dtype=np.uint8)
white_upper = np.array([180, 25, 255], dtype=np.uint8)

image_files = ["./imgs/1.jpg", "./imgs/2.jpg", "./imgs/3.jpg", "./imgs/4.jpg"]

for file in image_files:
    img = cv2.imread(file)
    img = cv2.resize(img, (640, 480))

    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    yellow_mask = cv2.inRange(hsv, yellow_lower, yellow_upper)
    white_mask = cv2.inRange(hsv, white_lower, white_upper)

    combined_mask = cv2.bitwise_or(yellow_mask, white_mask)

    result = cv2.bitwise_and(img, img, mask=combined_mask)

    cv2.imshow("Yellow and White Lines", result)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

3. 풀이과정

노란색과 흰색의 HSV 색상 범위를 설정한 후 image_files 리스트에 이미지 파일 경로를 지정해주었다. for문을 사용하여 image_files 리스트에 있는 각 이미지 파일을 순회하여 이미지를 읽어오고 resize를 통해 크기를 조정하였다. 노란색 마스크와 흰색 마스크를 생성하고 두 개를 OR 연산으로 결합하여 combined_mask를 생성하였다. 마지막으로 원본 이미지와 combined_mask를 AND 연산하여 result 이미지를 생성하고 imshow 함수를 통해 출력하였다.