연구 주제명:
opencv 를 이용한 실시간 눈동자 추적 프로그램

연구 목표: 실시간 얼굴 검출 및 눈동자 추적 프로그램 작성

연구 요약:

[프로그램 요약]

1. OpenCV 의 dlib 얼굴 검출 모듈을 사용하여 영상 속 얼굴의 68 개의 랜드마크를 추출한다.

2. 두 눈에 해당하는 랜드마크(각 6 개 * 2)의 좌표를 기반으로 눈 영역의 마스크를 생성한다. (얼굴 전체가 아닌 두 눈 영역에 대해서만 효율적으로 영상처리하기 위함.)

3. 각 마스크 이미지는 인식률을 높이기 위한 전처리 과정을 거친 후, Hough Tranform 알고리즘을 사용하여 이미지 내 원을 검출한다.

5 월 4 일

국민대학교 전자공학부 전자종합설계 I

연구원: 김정민(20162934) , 신미나(20201463)

연구 내용

영상 전체 영역을 대상으로 눈동자를 검출하려고 하면 정확도가 떨어짐
-> dlib 모듈을 통해 영상 내 얼굴로 인식된 영역 내에서만 눈동자를 검출한다.

[윈도우, vs code 환경에서 dlib 모듈 사용하기]

구글 colab 환경에서 명령어 단 몇 줄로 모듈을 불러와 사용했던 것과 달리 vs code 에서 dlib 모듈을 사용하기 위해서 아래 과정을 거쳐야 한다.

1) visual studio 의 c++ 컴파일러가 없다면 설치
2) CMake 설치
    - CMake 오피셜 사이트에서 알맞은 플랫폼 버전의 CMake 를 다운로드
    - 설치 마법사 창에서 "Add CMake to the PATH for all users"체크 후 설치 진행
    - 명령 프롬프트 창에 "pip install cmake" 입력하여 모듈 설치
3) 마찬가지로 "pip install dlib"입력하여 dlib 라이브러리 설치
    - python 창에 "import dlib" 오류가 없다면 설치 성공
    * 위 CMake 가 빌드되어있지 않으면 반드시 설치 오류가 난다.
4) 얼굴 랜드마크 인식기 적용하기
    - dlib 폴더에 따로 인식기 파일이 없음
    - shape_predictor_68_face_landmarks.dat.bz2 파일을 따로 다운로드
    - 압축 해제하여 shape_predictor_68_face_landmarks.dat 파일을 저장(윈도우에서 BZ2 파일을 압축 해제하기 위해 응용 프로그램 7z2201-x64.exe 사용)

위 과정을 통해 정상적으로 dlib 모듈을 불러와 얼굴 검출 코드를 작성할 수 있다.
다만 Colab 을 쓰거나 아나콘다를 설치하는 방법은 따로 고려하지 않았음.

[참고] How to Install dlib Library for python in windows 10 - GeeksforGeeks

결과물(코드) & 설명 :

```
import cv2
import dlib
import numpy as np
```

✓ 0.0s Python

필요한 모듈 import.

```
# 육각형 영역에서 원을 검출하여 반환하는 함수
def detect_circle_in_hexagon(img, hexagon):
    # 육각형 좌표를 이용해 마스크 생성
    mask = np.zeros_like(img)
    cv2.fillPoly(mask, np.int32([hexagon]), (255, 255, 255))

    # 마스크 적용 및 전처리
    masked = cv2.bitwise_and(img, mask)
    gray = cv2.cvtColor(masked, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(blur, 30, 100)
    #kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)) # (전처리:검출된 엣지를 닫힘 연산)
    #edges = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)

    # 원 검출
    circles = cv2.HoughCircles(edges, cv2.HOUGH_GRADIENT, dp=1, minDist=20, param1=50, param2=20,
                               minRadius=5, maxRadius=20)

    # 원 검출 결과 반환
    return circles
```

✓ 0.1s Python

매개 변수로 이미지와 육각형의 꼭지점 좌표 리스트가 주어지면 육각형 영역에서 Circle 을 검출하여 반환하는 함수.

68_face_landmarks 중 왼쪽 눈 인덱스: `[36, 37, 38, 39, 40, 41]` , 오른쪽 눈 인덱스 : `[42, 43, 44, 45, 46, 47]`
각 눈의 랜드마크 점이 6 개이므로 눈 영역을 자르는 마스크의 모양은 사각형보다 육각형이 더 효율적일 것이라 생각하여 육각형 마스크를 사용하였다.

이미지와 같은 사이즈의 빈 이미지인 mask 를 생성하고, cv2.fillPoly 함수를 통해 매개변수로 주어진 6 개의 좌표를 이용하여 육각형 부분만 흰색`(255, 255, 255)` 인 마스크 이미지를 생성한다. 그런 다음, cv2.bitwise_and 함수로 전체 이미지 중 마스크의 흰색에 해당하는 부분만 추출할 수 있다. 추출된 이미지(양 눈)에 적당한 전처리과정을 거친 후, Hough 변환 알고리즘을 통해 원을 검출한다.

'눈동자'로 인식하는 기준은 오직 '원 모양의 오브젝트' 뿐이기 때문에 인식률이 저조할 가능성이 높다. 이 함수에서는 Canny 엣지 검출 + Hough 변환 알고리즘과 같은 단순 영상처리 기법을 사용하였지만, 눈동자를 정확하게 추정하기 위한 다른 방법으로 딥러닝, 랜덤 포레스트, 클러스팅 알고리즘 등을 사용할 수도 있겠다.

마스크 이미지의 전처리 과정과 HoughCircles 함수의 파라미터 값에 따라 인식률이 다르게 나타난다. (위 코드에서 닫힘_모폴로지연산은 효과적인 노이즈 제거를 위해 사용하였으나 의외로 없는 경우가 더 인식이 잘 되어 주석처리 하였다.)

```python
# circles의 정보 입력받아 이미지에 원 둘레를 그리는 함수
def draw_circles(img, circles):
        if circles is not None:
            circles = np.round(circles[0, :]).astype("int")
            for (x, y, r) in circles:
                cv2.circle(img, (x, y), r, (0, 255, 0), 2)
```

✓ 0.0s Python

검출된 circles 를 입력 받은 뒤 cv2.circle 함수를 이용해 중심점의 좌표, 반지름으로 line 은 계산해 알맞은 이미지 위치에 원의 윤곽선을 그리는 함수.

```python
# 얼굴 인식 model, 랜드마크 인식기 생성
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('./content/shape_predictor_68_face_landmarks.dat')

# 웹캠 열기
cap = cv2.VideoCapture(0)

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

    # 얼굴 검출
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)

    for face in faces:
        # 얼굴 영역에서 랜드마크 검출
        landmarks = predictor(gray, face)

        # 양 눈 영역을 나타내는 랜드마크 인덱스
        left_eye = [36, 37, 38, 39, 40, 41]
        right_eye = [42, 43, 44, 45, 46, 47]

        # 양 눈의 x, y 좌표를 저장할 리스트 초기화
        left_eye_pts = []
        right_eye_pts = []

        for n in left_eye:
            x = landmarks.part(n).x
            y = landmarks.part(n).y
            left_eye_pts.append((x, y))


        for n in right_eye:
            x = landmarks.part(n).x
            y = landmarks.part(n).y
            right_eye_pts.append((x, y))

        # 양 눈동자를 검출하고 경계선 그리기
        left_circles = detect_circle_in_hexagon(frame, left_eye_pts)
        right_circles = detect_circle_in_hexagon(frame, right_eye_pts)
        draw_circles(frame, left_circles)
        draw_circles(frame, right_circles)

    # 화면에 프레임 출력
    cv2.imshow("frame", frame)

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

# 웹캠 해제 및 창 닫기
cap.release()
cv2.destroyAllWindows()
```

✓ 14.2s Python

Main 코드.

Dlib 라이브러리로부터 얼굴 인식 모델과 인식기를 불러와 초기화.

웹캠을 열고 찍히는 영상의 매 프레임마다 얼굴의 랜드마크를 검출, 양 눈의 좌표 리스트 생성 후 눈동자 검출 함수와 원 그리는 함수를 통해 실시간으로 영상 위에 검출된 눈동자를 그리는 기능을 수행한다. [참고] New chat (openai.com)

실행 결과 :

dlib 모듈로 얼굴의 68 개의 랜드마크를 찍어 실시간으로 보여주는 기능은 높은 인식률과 추적율을 보였지만 위 함수로 구현한 눈동자 검출 기능은 예상한 것과 같이 확연히 낮은 인식률과 추적 정도를 보였다. 이유는 input 의 다양한 또는 복잡한 경우를 고려하지 않고서 Hough 변환 알고리즘에만 의존한 '약한 인식 기준'을 세웠기 때문이라고 생각한다. 눈에서 눈동자가 완전히 드러나지 않을 경우, 안면이 정면에서 벗어났을 경우 등 다양한 경우 인식에 실패하였다.

개선할 점 or 다음 목표 :

인식률 개선을 위해서는 앞서 언급한 딥러닝 또는 클러스터 알고리즘 같은 다른 방식을 사용하는 것도 고려해야 할 것이다.
다음 연구에는 인식률 개선을 위한 다른 객체인식 방법을 찾아보며 프로그램을 수정하고, 연구 목표인 아이트래킹 기술을 이용해 구체적으로 어느 목적으로, 어떻게 활용 시킬지 계획할 예정이다.