Implementing one of face recognition algorithms

- PC 카메라 영상에 나타나는 사람을 인식하여 정보 표시 -

▷ **이름: 손 수 지**

▷ **학번: 202011933**

▷ **학과: 전자정보통신공학부 전자공학 전공**

▷ **과목(분반): 객체지향프로그래밍 (108)**

▷ **담당교수: 박 한 훈 교수님**

▷ **제출일: 2022.06.18**

- 목 차 -

Ⅰ. 서론 ····· 3
1. 과제 수행 배경(동기)과 목적 ····· 3
2. 과제 수행 방향 및 방법 ····· 3
2-1. 과제 수행 방향 ····· 3
2-2. 방법 및 관련 내용 ····· 3

Ⅱ. 본론(구현) ····· 6
1. 얼굴 데이터 수집하기 ····· 6
2. 수집한 dataset으로 학습시키기 ····· 8
3. 실시간 얼굴 인식 및 정보 표시 ····· 10
4. 전체 코드 ····· 12

Ⅲ. 결과 및 분석 ····· 15
1. 실험(시뮬레이션) 환경 ····· 15
2. 실험 결과 및 분석 ····· 15
3. 한계 및 개선 방향 ····· 19

Ⅳ. 결론 ····· 20
1. 과제 수행 결과 요약 ····· 20
2. 과제 수행을 통해 느낀 점 ····· 20

Ⅴ. 참고문헌 ····· 21

Ⅰ. 서론

1. 과제 동기 및 목적

객체지향프로그래밍의 두 번째 과제 'PC 카메라 영상에 나타나는 사람을 인식하여 정보 표시하기'를 수업시간에 학습한 Python과 OpenCV를 활용하여 구현해보고자 하였습니다.

2. 과제 수행 방향 및 방법

2-1. 과제 수행 방향

이 과제의 목표는 PC 카메라 영상에 나타나는 사람을 인식하여 정보를 표시할 수 있도록 프로그래밍 하는 것입니다.
먼저, PC 카메라 영상에 나타나는 사람을 컴퓨터가 인식하기 위해서는, 얼굴을 감지해야 합니다. 예를 들어, 제가 어떤 얼굴도 모르던 새로운 사람을 만나게 되었는데 이름이 '가연'이라는 것을 알게 되었다고 합시다. 그러면 이후에 이 '가연'이라는 사람 얼굴을 보았을 때, 이사람은 '가연이다'라는 생각을 떠올릴 수 있습니다. 마찬가지로, 컴퓨터가 얼굴을 감지하기 위해서는 얼굴 이미지와 그 정보를 함께 학습시켜 주면 됩니다. 따라서 저는 컴퓨터의 웹캠을 활용해 영상에 인식되는 얼굴마다 id를 부여하고, 얼굴을 캡쳐해 data set을 만들어 recognizer을 학습시킨 뒤, 학습된 recognizer을 사용해 얼굴을 인식하고, 정보를 표시할 수 있도록 프로그래밍 해보고자 합니다.

2-2 방법 및 관련 내용

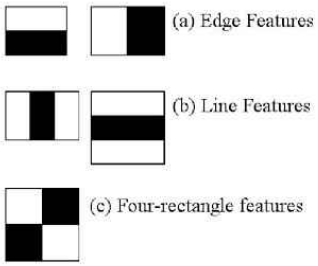
'PC 카메라 영상에 나타나는 사람을 인식하여 정보를 표시'하게 하기 위한 알고리즘은 세 개의 큰 틀로 정리할 수 있습니다.
1. 먼저, 인식할 얼굴의 dataset을 만들어 컴퓨터를 학습시켜야합니다. 영상처리 프레임워크 OpenCV에서 제공하는 Haar Cascade Classifier를 사용해 영상에서 얼굴을 검출한 뒤, 이미지를 캡쳐해 data set을 만듭니다.
2. OpenCV 패키지에 포함된 인식기 LBPH를 얼굴 recognizer로 사용하고, data set에서 인식되는 얼굴과 id를 반환하며 recognizer를 훈련시키고, yml파일로 저장합니다.
3. 훈련된 인식기를 불러와 현재 캠에서 인식되는 얼굴의 id를 예측합니다. id보단 이름을 나타내는 것이 더 편리하다고 생각하여 이름에 관한 list를 만들어두고, id에 해당하는 인덱스값을 반환하여, 해당 인덱스값에 해당하는 list의 요소를 출력할 수 있도록 하였습니다.

a. Haar Cascade Classifier

이 과제에서 얼굴 (또는 객체)을 감지하기 위해 OpenCV에서 제공하는 "Haar Cascade Classifier"를 사용하였습니다. "Haar Cascade Classifier"는 이미지의 밝기 차를 이용

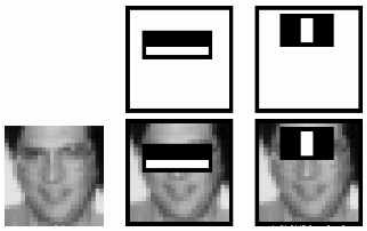
하여 특징을 찾아내고, 특징에 다라 대상의 분류를 해주는 알고리즘입니다. Haar 기능 기반 Cascade 분류기를 사용한 객체 감지는 Paul Viola와 Michael Hones가 2001년 논문 “Rapid Object Detection using a Boosted Cascade of Simple Features”에서 제안한 효과적인 객체 감지 방법입니다.

b. Haar Feature



여기서 Haar filter란 흑백 사각형이 서로 붙어있는 형태롤 구성된 필터입니다. 이런 다양한 작은 네모(Haar feature, kernnel)를 이미지에 옮겨가면서 검은색 부분과 흰색 부분 각각의 밝기 값을 빼서 특징을 찾습니다. 이미지에 필터를 전부 대입하는 것은 쉽고 단순하지만, 연산량이 많아 Integral Images(다음 픽셀에 이전 픽셀까지의 합이 더해진 영상)를 사용하여 빠르게 구할 수 있습니다.

c. Adaboost



사람의 얼굴은 대체적으로 눈이 있는 부분은, 이마와 코, 광대뼈의 굴곡의 아래에 위치하기 때문에 검은색 수평선의 특성을 지니고, 코가 있는 부분은 하얀색 수직선의 특성을 지닙니다. 눈과 코처럼 얼굴에서 의미 있는 Haar Feature은 밝기 차이가 특정 임계값 이상인 값이 됩니다. 즉, 흑과 백으로 이루어진 Haar feature가 처음에는 모두 같은 가중치를 가지고 있다가, 특정한 학습 data set을 이용하여 어떤 네모가 잘 찾아내는지 가중치를 조정하는 방식으로 진행됩니다.
즉, Adaboost라는 과정을 거치면, 얼굴이라는 정보에 해당하는 Haar Feature를 찾을 수 있습니다.

d. Cascade Classifier

위의 내용을 통해, cascade는 안면 검출시, 얼굴이 아닌 범위를 걸러내는 방식이라고 할 수 있습니다. 이미지에서 얼굴에 해당하는 Haar Featuref를 활용하여 얼굴을 찾습니다. 이미지 대부분 공간은 얼굴이 없는 영역이기 때문에, Haar feature 내의 영역에 얼굴이 있는지 빠르게 판별하기 위해 단계별로 진행합니다. 예를 들어, 낮은 단계에서 얼굴이 존재하지 않는다고 판단되면, 다음 단계는 확인하지 않고 바로 넘어가는 방법입니다.

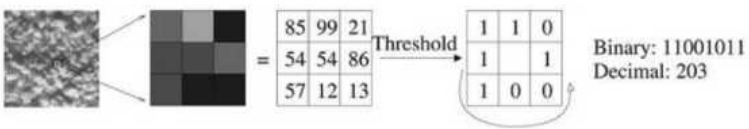
OpenCV에서 제공하는 Haar Cascade Classifier를 보면

haarcascade_eye.xml
haarcascade_eye_tree_eyeglasses.xml
haarcascade_frontalcatface.xml
haarcascade_frontalcatface_extended.xml
haarcascade_frontalface_alt.xml
haarcascade_frontalface_alt_tree.xml
haarcascade_frontalface_alt2.xml
haarcascade_frontalface_default.xml
haarcascade_fullbody.xml
haarcascade_lefteye_2splits.xml
haarcascade_licence_plate_rus_16stages.xml
haarcascade_lowerbody.xml
haarcascade_profileface.xml
haarcascade_righteye_2splits.xml
haarcascade_russian_plate_number.xml
haarcascade_smile.xml
haarcascade_upperbody.xml

이렇게 다양한 분류기를 제공하는 것을 볼 수 있습니다. 이중에서, 저는 얼굴인식을 위한 'haarcascade_frontalface_default.xml'을 사용하겠습니다.

e. LBP(Local-Binary-Pattern)HFace Recognizer

Local-Binary-Pattern 알고리즘은, 지역적인 주변의 값을 2진수로 표현한 뒤 값을 계산합니다.



3*3 픽셀에서 정중앙을 기준으로 주위 픽셀 8개와 크기를 비교합니다. 중심 픽셀의 값보다 크거나 같으면 1, 작으면 0으로 설정하는 것이다. 위의 경우 오른쪽과 같은 값을 얻을 수 있고, 이진수로 11001011로 표현할 수 있습니다. 이렇게 모든 픽셀에 대해 계산해줍니다. 그리고 히스토그램(분포표)을 만들어, 값의 비교(제곱오차)를 통해 유사한 얼굴을 찾아냅니다.

Ⅱ. 본론(구현)

1. 얼굴 데이터 수집하기

1) 필요한 라이브러리 가져오기

```
import cv2  #opencv 모듈/프레임워크
```

2) 카메라 정보 받아오기

```
cam = cv2.VideoCapture(0)
cam.set(3, 640) #너비 초기값 설정
cam.set(4, 480) #높이 초기값 설정
#opencv에서 제공하는 분류기 중 얼굴을 인식하는 분류기 가져오기
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

3) USER ID 입력받기

```
face_id = input('\n USER ID 입력하고 엔터 누르세요 ')
#id를 입력받고 카메라로 얼굴을 캡처하기 전, 출력문구 작성
print("\n [INFO] Initializing face capture. Look the camera and wait ...")
```

4) 영상 처리 및 출력

```
count = 0  #count 변수 정의, 초기화
while(True):  #while 반복문 활용 -> 각 프레임마다 영상 처리 및 출력 반복
    ret, frame = cap.read()   #카메라 상태 및 프레임
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)   #흑백으로
    faces = faceCascade.detectMultiScale(
        gray,  #검출하고자 하는 원본이미지
        scaleFactor = 1.2,  #검색 윈도우 확대 비율, 1보다 커야 한다
        minNeighbors = 6,  #얼굴 사이 최소 간격(픽셀)
        minSize=(20,20)   #얼굴 최소 크기. 이것보다 작으면 무시
    )
    #얼굴에 대해 rectangle 출력
```

```
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (255,0,0), 2)
        #inputOutputArray, point1 , 2, colorBGR, thickness)
        count += 1
        #만들어 둔 dataset 폴더에 사진 저장
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg",
        gray[y:y+h,x:x+w])
    cv2.imshow('image', frame)
#while문 종료 조건
    k = cv2.waitKey(100) & 0xff
    if k == 27:                 # ESC 눌렀을 때 종료
        break
    elif count >= 100:          #100장의 얼굴 sample을 캡처했을 때 종료
        break
#메모리 해제
cap.release()
#모든 윈도우 창 닫기
cv2.destroyAllWindows()
```

2. 수집한 dataset으로 학습시키기

1) 필요한 라이브러리 가져오기

```
import cv2     #opencv 모듈
import numpy as np     #배열 계산 용이
from PIL import Image     #파이썬 이미지 라이브러리
import os    #os모듈

path = 'dataset'  #이미지 dataset 경로 (dataset 폴더)
#LBPHFaceRecognizer 객체 생성
recognizer = cv2.face.LBPHFaceRecognizer_create()
#detector 객체 생성
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

2)이미지와 레이블 데이터를 가져오는 함수 정의

```
# 이미지 불러와서 라벨링하기
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    #listdir : 해당 디렉토리 내 파일 리스트
    #path + file Name : 경로 list 만들기

    faceSamples = []
    ids = []
    for imagePath in imagePaths:  #각 파일마다
        #흑백 변환
        PIL_img = Image.open(imagePath).convert('L')
                    #convert() : 8 bit pixel 이미지로 바꿔줌.
        img_numpy = np.array(PIL_img, 'uint8')
        # user id (정수로만 입력받았기 때문에, int 사용하면 편리)
        id = int(os.path.split(imagePath)[-1].split(".")[1])
```

#split(imagePath)[-1] : user.1.99.jpg 의 형식으로 되어 있는 파일을 처리하기 위하여 파일 이름을 먼저 split분리해내고,

#split(file full path)[-1].split(".")[1] : (0번째 user) 1번째 id값을 가져온다.

```
        #학습을 위한 얼굴 샘플
        faces = detector.detectMultiScale(img_numpy)
        for(x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)

    return faceSamples, ids

print('\n [INFO] Training faces. It will take a few seconds. Wait ...')
faces, ids = getImagesAndLabels(path)

recognizer.train(faces,np.array(ids)) #학습

# trainer.yml로 모델 저장하기
recognizer.write('trainer/trainer.yml')
#학습된 얼굴 수 뽑고, 종료
print('\n [INFO] {0} faces trained. Exiting Program'.format(len(np.unique(ids))))
```

3. 실시간 얼굴 인식 및 정보 표시

1) 필요한 라이브러리 가져오기

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

2) 2에서 학습시킨 recognizer 가져오기

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = 'haarcascade_frontalface_default.xml'
faceCascade = cv2.CascadeClassifier(cascadePath)

font = cv2.FONT_HERSHEY_SIMPLEX  #폰트 설정

id = 0  #id 카운터 초기화
#user id에 해당하는 이름 배열 만들기
names = ['None', 'suji','jaeseong','eunjeong','soeun']

cam = cv2.VideoCapture(0)
cam.set(cv2.CAP_PROP_FRAME_WIDTH, 1980)
cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
# 얼굴로 인식 될 최소 창 크기 정의
minW = 0.1 * cam.get(cv2.CAP_PROP_FRAME_WIDTH)
minH = 0.1 * cam.get(cv2.CAP_PROP_FRAME_HEIGHT)
```

3) 얼굴 검출하고, 예측을 하는 while문 작성

```
while True:
    ret, frame = cam.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
```

```
        minNeighbors=6,
        minSize=(int(minW), int(minH))
    )
    # 얼굴에 대한 예측
    for(x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0),2)
        id, confidence = recognizer.predict(gray[y:y+h, x:x+w])  #얼굴부분만 가져옴
# recognizer.predict(src) : 얼굴을 예측. user id와, 확률값(confidence 값)을 가져온다.
                            (confidence: 0에 가까울수록 label과 일치)
        # 신뢰도가 작으면 100 ==> "0" 는 완벽하게 일치함을 의미
        if confidence < 100 :
            id = names[id]  #name 리스트의 id 인덱스 값을 id로 함
        else:
            id = "unknown"  #신뢰도가 0이면 id를 “unknown”

        confidence = "  {0}%".format(round(100-confidence))

        # id의 label과 예측값을 이미지에 폰트로 출력
        cv2.putText(frame,str(id), (x+5,y-5),font,1,(255,255,255),2)
        cv2.putText(frame,str(confidence), (x+5,y+h-5),font,1,(255,255,0),1)

    cv2.imshow('camera', frame)
    if cv2.waitKey(1) > 0 : break  # 캠 종료 시 ‘ESC’ 누르기

print("\n [INFO] Exiting Program and cleanup stuff")  #종료문구
cam.release()
cv2.destroyAllWindows()
```

4. 전체 코드

01_face_dataset.py | 03_face_recognition.py | 02_face_training.py

```python
import cv2
import os

cam = cv2.VideoCapture(0)
cam.set(3, 640) # 너비 설정
cam.set(4, 480) # 높이 설정

face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# 사람 얼굴 마다 숫자로 ID를 입력
face_id = input('\n enter user id end press <return> ==>  ')

print("\n [INFO] Initializing face capture. Look the camera and wait ...")
# 얼굴 카운트를 초기화한다
count = 0

while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1

        # 만들어둔 dataset 폴더에 사진이 저장된다
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('image', img)

    k = cv2.waitKey(100) & 0xff # 'ESC'를 눌러 종료한다
    if k == 27:
        break
    elif count >= 30: # 30장의 얼굴 샘플을 캡쳐하고 비디오를 중지
         break

# 정리
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

01_face_dataset.py | 03_face_recognition.py | 02_face_training.py*

```python
import cv2
import numpy as np
from PIL import Image
import os

# 이미지 데이터베이스 경로
path = 'dataset'

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

# 이미지와 레이블 데이터를 가져오는 함수
def getImagesAndLabels(path):

    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []

    for imagePath in imagePaths:

        PIL_img = Image.open(imagePath).convert('L') #  grayscale로 변환
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)

    return faceSamples,ids

print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))

# 모델을 trainer/trainer.yml에 저장
recognizer.write('trainer/trainer.yml')

# 훈련된 얼굴 수를 print하고 종료
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

01_face_dataset.py | 03_face_recognition.py*

```python
import cv2
import numpy as np
import os

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#id 카운터 초기화
id = 0

# id와 관련된 이름 불이기: example ==> yeojin: id=0,  etc
names = ['', 'su', 'suji']

# 실시간 영상 캡쳐 초기화 및 사이즈
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height

# 윈도우 인식 될 최소 창 크기를 정의한다.
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:

    ret, img =cam.read()

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

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
       )

    for(x,y,w,h) in faces:

        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        # 신뢰도가 작으면 100 ==> "0" 는 완벽하게 일치함을 의미
        if (confidence < 100):
            id = names[id]
            confidence = "  {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = "  {0}%".format(round(100 - confidence))

        cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
        cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

    cv2.imshow('camera',img)

    k = cv2.waitKey(10) & 0xff # 'ESC'를 눌러 종료한다
    if k == 27:
        break

# 정리
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

Ⅲ. 결과 및 분석

Python 3.6

spyder

1. 얼굴 데이터 수집하기

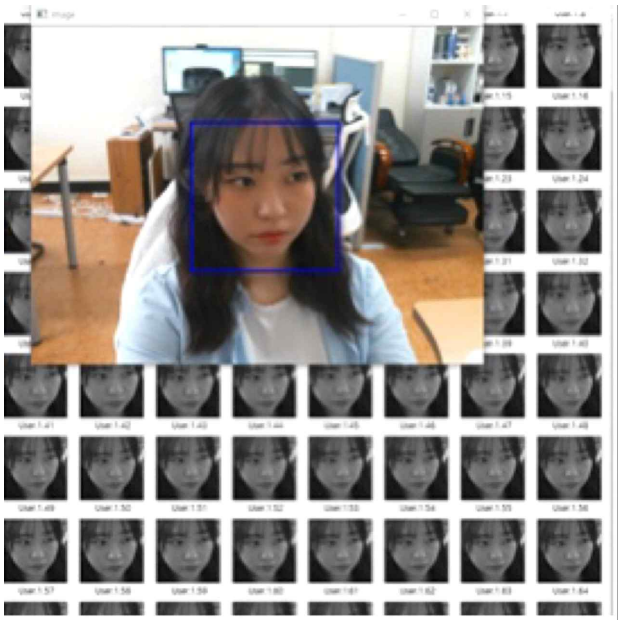
```
In [2]: runfile('F:/OpenCV-Face-Recognition-master/FacialRecognition/01_face_dataset.py
OpenCV-Face-Recognition-master/FacialRecognition')


 enter user id end press <return> ==>  2
```

user id 입력하라는 input 함수가 실행된 것을 확인 할 수 있었습니다.

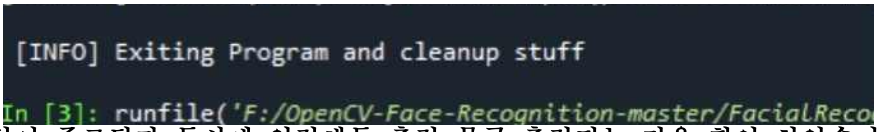
제 얼굴은 user id 1로, 다른 친구를 user id 2로 입력한 후, 엔터 눌렀습니다.

```
In [2]: runfile('F:/OpenCV-Face-Recognition-master/FacialRecognition/01_face_dataset.py
OpenCV-Face-Recognition-master/FacialRecognition')


 enter user id end press <return> ==>  2

 [INFO] Initializing face capture. Look the camera and wait ...
```

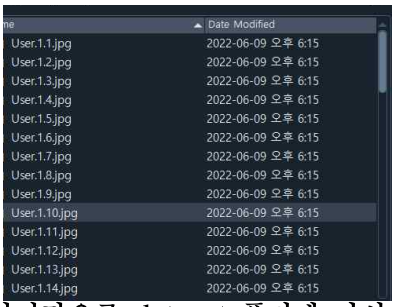
입력해뒀던 print 문구가 출력되면서



캠을 통해 100개의 캡쳐본을 저장한 뒤 창이 종료되었습니다.

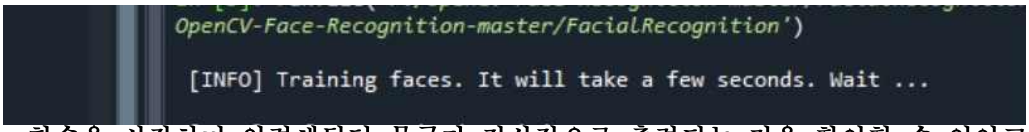


창이 종료됨과 동시에 입력해둔 출력 문구 출력되는 것을 확인 하였습니다.

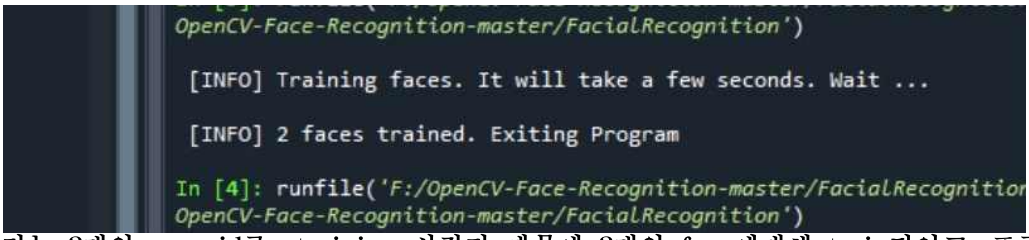


마지막으로 dataset 폴더에 가서user id 1과 2에 대해 100개의 dataset이 생성된 것을 확인함으로써, '얼굴 데이터 수집하기'의 구현이 설계한 대로 잘 진행됨을 확인하였습니다.

2. 수집한 dataset으로 학습시키기

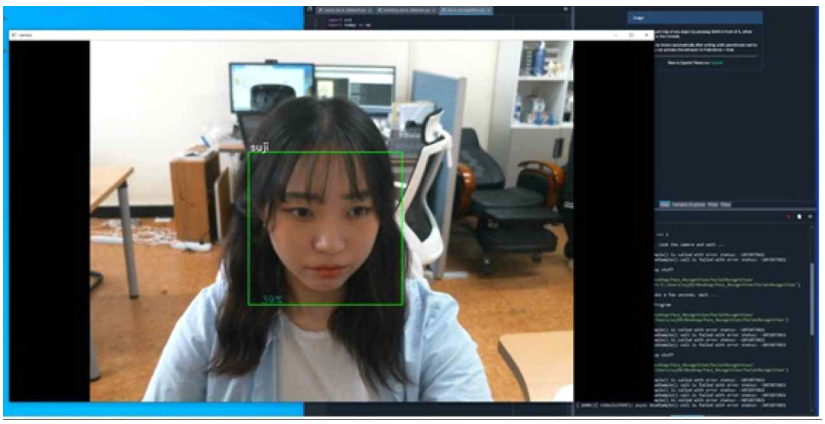


학습을 시작하며 입력해뒀던 문구가 정상적으로 출력되는 것을 확인할 수 있었고,



저는 2개의 user id로 training 시켰기 때문에 2개의 face에대해 train되었고, 프로그램이 종료되었다는 문구가 설계와 같이 출력되는 것을 확인 할 수 있었습니다.

3. 실시간 얼굴 인식 및 정보 표시



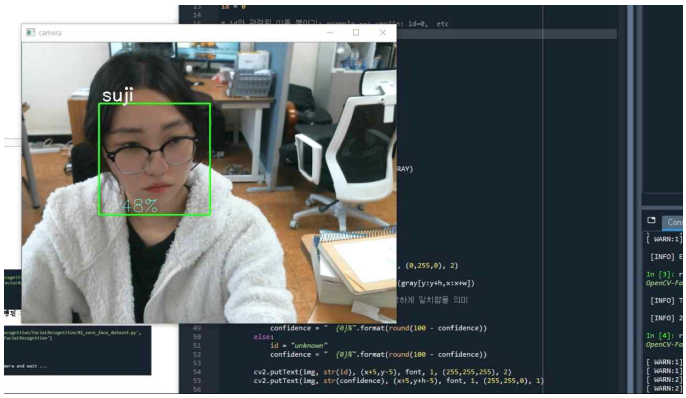
실행 결과, 설계해 둔 대로 opencv를 통해 연결된 캠이 정해둔 크기대로 열렸으며, 사각형의 위쪽에는 예측된 id의 인덱스에 관련한 user names를, 아래쪽에는 id에 관한 얼굴과 일치하는 (100-confidence)의 퍼센테이지를 잘 나타내는 것을 볼 수 있었습니다.

위에 첨부한 결과 이미지는 얼굴이 인식되는 범위 내에서 가장 안좋은 결과값을 나타내는 시점을 캡쳐한 사진입니다. 32~68%의 confidence 값을 출력해 내는 것을 확인할 수 있었습니다.

‘1. 얼굴 데이터 수집하기’에서 캡쳐 할 때의 얼굴 각도와 시선 처리가 비슷할수록 출력되는 confidence 값이 커지는 것을 확인할 수 있었습니다.

4. 한계 및 개선방향

제 얼굴로 학습되어있는 이 프로그램을 안경과 같은 액세서리를 착용하고 실행 해보았을 때 결과가 더 나빠지지 않을까?하는 호기심에 며칠 뒤 안경을 끼고, 다른 옷을 입고 프로그램을 실행시켜봤습니다.



훨씬 저조한 confidence 값을 나타낼거라 생각하였지만 실험 결과, 22~55% 정도의 confidence 값을 출력하는 것을 보고 생각보다 잘 구현되는 것을 확인할 수 있었습니다. 안경으로 인한 변화보다는 얼굴의 각도에 더 많은 영향을 받는다는 것을 직접 실험해봄으로써 알 수 있었습니다.

recognizer의 인식 결과가 제 예상보다는 저조했습니다. 채 5분도 걸리지 않는 시간동안 제 얼굴의 data set을 저장하고, 학습시키고, 학습된 recognizer을 사용해 바로 얼굴 인식을 하였지만 70%도 되지 않는 유사율에 조금 놀랐습니다.
100개라는 적은 dat set과 좀 더 밝지 못한 환경이 이유라고 생각했습니다. 종강 후, 훨씬 많은 data set과 얼굴이 좀 더 잘 보일 수 있도록 밝은 환경에서 실험을 진행해보려고 합니다.

Ⅳ. 결론

1. 요약

Haar Cascade 검출기를 사용하여, 실시간 웹캠으로 인식되는 얼굴을 프레임 단위로 캡쳐하여 100개의 이미지 데이터를 수집합니다. 수집한 dataset을 Haar Cascade 학습기로 학습시킨 후, yml 파일로 모델을 저장합니다. (이름은 'trainer.yml') 학습한 yml 파일 모델을 불러와서, 캠을 통해 실시간으로 얼굴을 인식하고, 인식한 얼굴을 예측하여 라벨링 결과를 사각형 위, 아래에 각각 id에 해당하는 이름과, 유사율을 나타내도록 하였습니다. 실험 결과를 보면 전체적으로는 제 예상보다는 저조한 성능을 보이지만, 안경이라는 악세사리를 착용했음에도 인식율이 비슷하다는 부분에서는 예상보다는 얼굴이 잘 인식되는 것을 알 수 있었습니다.

2. 느낀점

과제를 처음 시작할땐 막상 어디서부터 시작해야 할지 막막했지만, 구글이나 깃허브에 opencv를 활용한 소스코드가 몇가지 있었습니다. 처음엔 코드를 그대로 가져와 실험을 진행했지만 제 컴퓨터에 맞지 않았는지 자꾸 오류가 나서 소스코드를 수정하는 방법으로 프로그래밍을 진행했습니다.
어떤 부분에서는 기대에 못미치는 결과를 보여줬고, 어떤 부분에서는 예상보다 더 좋은 결과를 나타내는 것을 직접 실험을 통해 확인하게 되었습니다.
비대면으로 인해 수업시간에는 파이썬을 활용해 직접 구현해 볼 기회가 없었지만, 과제를 통해 함수, 리스트, 객체, 클래스등을 사용해 직접 실습 해 볼 수 있는 좋은 기회였습니다.

Ⅴ. 참고문헌

[1] Marcelo Rovai / Real-Time Face Recognition: An End-To-End Project/ 18.02.23 (https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826)