REPORT3

제목 : 사진속의 런닝맨 멤버 찾기

과목명	디지털영상처리
담당 교수님	김진헌

전공	컴퓨터공학과
학번	2019305059
이름	이현수

제출일	2021. 12. 18. 토

목차

Ⅰ.문제정의

1. 문제

사진속에 있는 런닝맨 멤버를 찾는다. 런닝맨 멤버가 포함된 사진에서 멤버를 찾아 그 인물의 얼굴에 사각형을 그리고 그 상단에 인물의 영문 이름을 적어 넣는다. 이 프로그램은 런닝맨 멤버 7인(유재석, 지석진, 김종국, 하하, 송지효, 전소민, 양세찬)에 대해 총 5개의 검색 사진 파일에 대해 프로그램을 수행한다.

2. 프로그램 출력 결과물

- matplotlib창 5개 - 기본
  프로그램이 시작되면 별도의 키보드 개입없이 matplotlib 창을 5개 열어 결과를 출력하고 종료한다. 이때 창은 최대크기로 띄운다.

- matplotlib창 5개 - 추가 정보
  검출된 인물 별로 그 사람이라고 판단하게 된 얼굴을 3순위까지 선택해 유클리디언 거리 순으로 표시. 이때 타이틀에 유클리디언 거리를 함께 보인다.

- 출력창 - 추가 정보
  matplotlib창과는 별도로 출력창에는 얼굴 검출 번호 별 판단한 멤버의 이니셜(혹은 이름)과 그때의 유클리디언 거리를 표시한다.

```
[File: r2.png] ----------------------------
face 0: Jeon=0.369
face 1: yang=0.370
face 2: ha=0.379
face 3: Jeon=0.431
[File: r2.png] 4 faces found including 4 members
```

2) 추가 가점 사항(1)

3. 본인 여부 판단법

128벅터의 유클리디안 거리가 0.6이하로 정한다. 같은 사람이 2인 이상 검출되더라도 같은 기준을 유지한다.

Ⅱ. 프로그램

1. 사용기술

■ 얼굴탐색 모듈에는 OpenCV, Dlib, face_recognition, cvlib가 있다.

□ OpenCV
- Haar cascade calssifier
- Deep learning based classifier

□ Dlib
- Hog 기반
- CNN 기반

□ face_recognition
- 내부적으로 dlib Hog, CNN detector 사용

□ cvlib
- 내부적으로 OpenCV DNN을 사용

■ 얼굴인식 모듈에는 OpenCV, Dlib, Face_recognition이 있다.

□ Opencv
- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms(LBPH)

□ Dlib
- DNN을 이용하여 얼굴을 128개의 벡터로 수치화한다. 미지의 얼굴을 128벡터와의 유클리디안 거리로 동일인물을 판단한다.

□ Face_recogntion
- Dlib의 인코딩과 face 비교 함수를 사용한다.

이번 과제에서는 **Dlib Hog 기반**으로 얼굴을 탐색하고, **Dlib**를 사용해 얼굴을 인식한다.

1) Dlib face detection - Hog

- frontal face detector를 생성해 사용할 수 있다.
- Histogram of Oriented Gradients(HOG)는 feature + SVM기반 학습알고리즘.
- sliding window detection approach
- 3000개의 “Labeled Faces in the Wild” DB를 사용해 학습
- 불과 몇 개의 영상으로 학습이 가능

2) Dlib 얼굴 인식

- Deep learning 기반으로 고품질 얼굴인식 지원으로 실세계의 데이터베이스에 대해 99.38%의 정확도를 구현.
- ResNet 34 뉴럴 모델을 기반으로 300만개의 얼굴에 대해 학습(21.4MB)
- 얼굴 특징을 나타내는 128차원의 descriptor 생성. 유클리디언 거리가 0.6이하이면 같은 사람으로 간주한다. 모델 자체가 다른 사람과 구분되는 얼굴의 특징벡터를 추출하므로 사전에 특징 모델을 학습할 필요가 없다.

■ Dlib 기반 얼굴 인식 학습과정

1. Triplets 단위로 학습한다. 즉 3개 단위 영상단위로 학습해 그중 2개는 같은 사람으로 128개의 벡터가 같은 사람에 대해서는 가까워지도록 하고 이 같은 사람의 벡터들이 다른 사람에 대해서는 각각 멀어지도록 학습한다.
2. 이러한 학습을 수천명 사람에 대해서 수백만개의 영상의 영상에 대해 반복학습.
3. 그 결과 생성되는 디스크립터는 다음 특징을 갖게 된다.
   - The generated 128D descriptors of two images of the same person are quite similar to each other.
   - The generated 128D descriptors of two images of different people are very different

■ Dlib 기반 얼굴 인식 활용과정

1. 인식하고자 하는 인물들의 영상을 이용해 사전에 각 인물들의 descriptor들을 확보한다.
2. 미지의 인물에 대한 디스크립터를 구하여 이 벡터와 기존에 확보하고 있는 인물의 벡터와의 비교를 통해 어떤 인물인지 판단한다. 이때 판단 방법은 간단하게는 유클리디어 거리를 사용할 수 있다.
3. 인식하고자 하는 인물에 대해 dlib 디스크립터를 구한 후 동일인인지 구별하고자하는 인물에 대한 디스크립터를구해 두 디스크립터의 거리가 일정 기준 이하이면 동일인으로 처리한다.

2. 사용데이터

■ 얼굴 랜드마크 검출

8.72MB

■ 얼굴인식

21.4MB

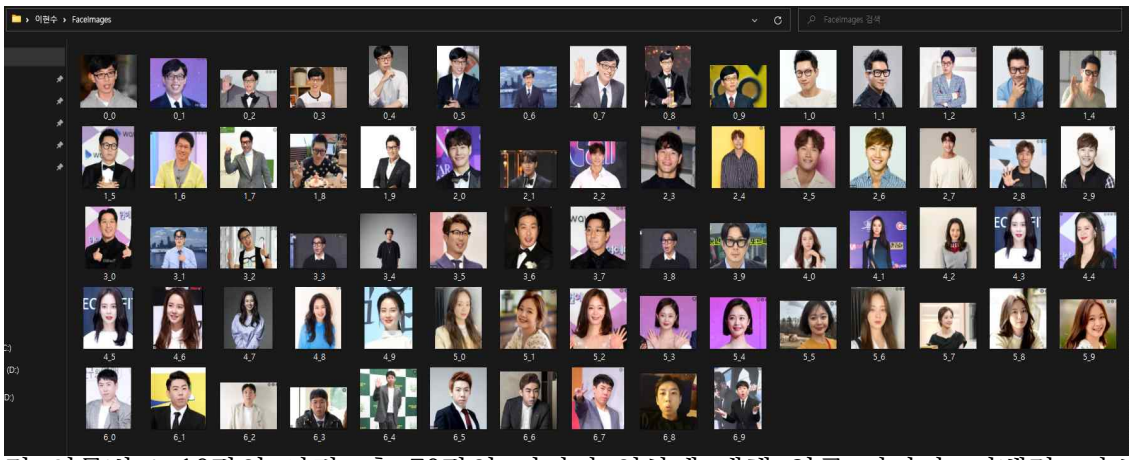
■ 얼굴인식 DB 파일

5.71MB

■ 인식하고자 하는 인물들의 영상

이번 과제에서 인식하고자 하는 인물은 런닝맨 멤버 7인이다.

- 유재석, 지석진, 김종국, 하하, 송지효, 전소민, 양세찬



각 인물별로 10장의 사진, 총 70장의 이미지 영상에 대해 얼굴 이미지, 라벨링, 디스크립터를 저장한다.

3. 전체적인 구조

■ unknown 이미지 정보 및 dlib 객체 생성

rpt_face.py db.py

```
import ...
# 필수 사항 ----------------------------
# 아래의 변수이름과 표현식은 바꾸지 말고 소스 그대로 상단에 노출되게 해 주십시오.

# 1) 테스트할 영상 파일이 있는 드라이브와 폴더. 실제 평가에서는 바꾸어 사용할 수 있습니다.
test_img_path = 'data/'#'d:/dip/images/'

# 2) 그 폴더 안에 있는 테스트 대상 파일 이름
# 이 파일은 런닝맨 멤버와 아닌 사람들이 섞여 있습니다.
# 실제 평가에서는 바꾸어 사용할 수 있습니다. 테스트 파일이 10여개가 될 수도 있습니다.
test_file_list = ['t1.jpg', 't2.jpg', 't3.jpg', 't4.jpg', 't5.jpg']

# 3) path for dlib model: 위치 바꾸지 말고 제출해 주세요.
# 평가자의 PC에 설치해 놓고 테스트 할 것입니다.
model_path = 'data/'#'d:/dip/data/dlib_face_recog/'

# 4) shape predictor와 사용하는 recognition_model
# 아래 정의된 것을 그대로 사용해 주세요.
pose_predictor_5_point = dlib.shape_predictor(model_path + "shape_predictor_5_face_landmarks.dat")
face_encoder = dlib.face_recognition_model_v1(model_path + "dlib_face_recognition_resnet_model_v1.dat")

# 검색하고자 하는 인물 7인
mb_han_lst = ['유재석', '지석진', '김종국', '하하', '송지효', '전소민', '양세찬']
mb_eng_lst = ['Yoo', 'ji', 'kim', 'ha', 'song', 'jeon', 'yang']
md_dic = {'Yoo':'유재석', 'ji':'지석진', 'kim':'김종국', 'ha':'하하',
          'song':'송지효', 'jeon':'전소민', 'yang':'양세찬'}

# threshold보다 큰 유클리디언 거리는 동일 인물로 판단하지 않음.
threshold = 0.6

# 얼굴검출 dlib hog face detector 사용.
detector = dlib.get_frontal_face_detector()
```

과제에서 주어진 코드를 상단에 작성. 데이터 파일 경로, 검색 대상 파일 이름, 검색 인물 정보와 dlib 기반 얼굴 검출, 랜드마크 검출, 얼굴 인식 객체를 불러온다.

■ 인코딩, 비교, 출력, 데이터파일load 함수 정의

rpt_face.py db.py

```
# face_encodings은 128차원의 ndarray 데이터들을 사람 얼굴에 따라 리스트 자료형과 얼굴 위치좌표를 반환
def face_encodings(face_image, number_of_times_to_upsample=1, num_jitters=2):
    """Returns the 128D descriptor for each face in the image"""
    # Detect faces:
    # hog 기반의 dlib face detector. gray 변환된 영상을 사용.
    gray = cv.cvtColor(face_image, cv.COLOR_RGB2GRAY)
    face_locations = detector(gray, number_of_times_to_upsample)
    # Detected landmarks:
    raw_landmarks = [pose_predictor_5_point(face_image, face_location) for face_location in face_locations]
    # Calculate the face encoding for every detected face using the detected landmarks for each one:
    return face_locations, [np.array(face_encoder.compute_face_descriptor(face_image, raw_landmark_set, num_jitters))
                            for raw_landmark_set in raw_landmarks]

# unknown encoding을 known_encoding list에 비교해 유클리디어 거리를 계산해 이름과 반환하고, 거리는 오름차순 정렬해 반환한다.
def compare_faces_ordered(dbfaces, encodings, face_names, encoding_to_check):
    """Returns the ordered distances and names when comparing a list of face encodings against a candidate to check"""
    distances = list(np.linalg.norm(encodings - encoding_to_check, axis=1))
    return zip(*sorted(zip(distances,dbfaces, face_names)))

# show_detection은 이미지에 있는 사람 얼굴에 빨간 사각형과 검색된 사람의 영문 이름을 출력해 반환
def show_detection(image, face, text, num):
    """Draws a rectangle and Eng name over each detected face"""
    cv.rectangle(image, (face.left(), face.top()), (face.right(), face.bottom()), (0, 0, 255), 2)
    cv.putText(image,text, (face.left()-10,face.top()-3), cv.FONT_HERSHEY_DUPLEX, 1, (255,0,0),2)
    cv.putText(image, str(num), (face.left() - 30, face.top() + 30), cv.FONT_HERSHEY_DUPLEX, 1, (255, 0, 0), 2)
    return image

def loadObjects(file_name):             # 지정한 파일, file_name에서 읽은 객체를 반환하기
    with open(file_name, "rb") as file:
        return(pickle.load(file))
```

face_encodings, compare_faces_ordered, show_detection, loadObjection 함수를 정의한다. 이 과제의 핵심 소스코드라고 할 수 있다.

face_encodings함수는 얼굴 검출, 랜드마크 검출, 128차원의 ndarray 데이터와 얼굴 검출 위치 좌표를 반환한다.

compare_face_ordered 함수는 비교대상 사진과 미리 준비된 얼굴인식 DB 정보와 유클리디언거리를 구하고, 유클리디언거리를 기준으로 오름차순 정렬한 결과를 (거리, 얼굴사진, 이름)을 반환한다.

show_detection 함수는 사진에 인식된 사람 얼굴에 파란 네모상자와 인식된 사람의 영문이름과 인식된 얼굴을 0번부터 출력한다.

■ unknown 이미지에 대해 encoding, db.bin 파일에서 인물정보 로드

```
69
70  start = time.time()
71
72  # 검색 대상 파일 즉 미지의 사진을 로드,
73  unknown_images = []
74  for i in range(len(test_file_list)):
75      unknown_image = cv.imread(test_img_path+test_file_list[i]) # image read
76      unknown_image = unknown_image[:, :, ::-1]         # Convert image from BGR (OpenCV format) to RGB (dlib format)
77      unknown_images.append(unknown_image)              # append image to unknown_images list
78
79  # 검색 대상 파일 사진에 대해 encoding
80  unknown_encodings = []
81  unknown_rects = []
82  for image in unknown_images:
83      unknown_image_rects, unknown_image_encoding = face_encodings(image)
84      unknown_encodings.append(unknown_image_encoding)  # 검색 대상 파일 사진에 여러사람이 존재 가능
85      unknown_rects.append(unknown_image_rects)         # 찾은 얼굴의 위치 좌표값 삽입
86
87  # 인물 데이터 읽기
88  all_member_face, all_member_encodings1d_lst, all_member1d_label_lst = loadObjects("db.bin")
89
```

known이미지들과 unknown이미지들을 cv모듈을 통해 읽고, RGB형태로 바꾼다. 그 후 known이미지들과 unknown 이미지들을 encoding한다.

“db.bin” 파일을 로드한다. 여기에는 인물 인식 대상 7인에 대해서 각각 10개의 얼굴 사진, 디스크립트, 라벨을 얻는다.

■ 얼굴 인식 및 결과 출력

```
# 5장의 검색 대상 파일에 대해 얼굴 인식, 얼굴 비교 후 matplotlib창 띄위기 및 출력창에 정보 출력
for i ,ue in enumerate(unknown_encodings):
    total_count = len(ue) # 총 감지된 얼굴 수
    member_count=0        # 인식된 런닝맨 멤버수
    img_faces = unknown_images[i].copy()
    rects = unknown_rects[i]
    text=[]
    extra=[]
    for m, ue_face in enumerate(ue): # 감지된 얼굴에 대해
        computed_distances_ordered, faces, ordered_names = compare_faces_ordered(all_member_face,
                                    all_member_encodings1d_lst, all_member1d_label_lst, ue_face) # 비교
        if(computed_distances_ordered[0]<threshold): # 유클리디안 거리가 0.6인 경우
            text.append([mb_eng_lst[int(ordered_names[0])],computed_distances_ordered[0]])      # 그 사람 이름 추가
            member_count+=1                    # 인식된 멤버수 변수 1 증가
            unface = unknown_images[i][rects[m].top():rects[m].bottom(),rects[m].left():rects[m].right()]
            info = [[unface,-1],[faces[0], computed_distances_ordered[0]],[faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)
        else: # unknown인 경우
            text.append(['unknown',-1])
            unface = unknown_images[i][rects[m].top():rects[m].bottom(), rects[m].left():rects[m].right()]
            info = [[unface, -1], [faces[0], computed_distances_ordered[0]], [faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)

    for j, ue_rect in enumerate(unknown_rects[i]):              # 인식된 사람 얼굴에 대해
        img_faces = show_detection(img_faces, ue_rect, text[j][0], j) # 사각형과 이름 출력
    plt.figure(num=f'{test_file_list[i]}')
    plt.imshow(img_faces)
    title = f'[File: {test_file_list[i]}] {total_count} faces found including {member_count} members'
    plt.title(title, fontsize=20)
    plt.axis('off')
    mng = plt.get_current_fig_manager()
    mng.window.state('zoomed')           # 창 최대화
    # 이미지 분석 창
    plt.figure(num=f'{test_file_list[i]} - analysis')
    for index, j in enumerate(extra,1):
        for k in range(4):
            img_index = index+k*(total_count)
            plt.subplot(4,total_count,img_index)
            if k!=0: plt.title(round(j[k][1],3))
            plt.axis('off')
            plt.imshow(j[k][0])
    mng = plt.get_current_fig_manager()
    mng.window.state('zoomed')           # 창 최대화
    # 주어진 형식으로 정보 출력
    print(f'[File: {test_file_list[i]}] ------------------------------')
    for k, info in enumerate(text):
        if info[1]==-1:
            print(f'face {k}: {info[0]}')
        else: print(f'face {k}: {info[0]}={round(info[1],3)}')
    print(title)
    print()
print("\n실행시간 : ", time.time()-start)
plt.show()
```

compare_face_ordered함수를 이용해 얼굴 인식을 하고 show_detection함수를 통해 결과를 사진에 반영하고 matplotlib 창 5개 + 이미지 분석 matplotlib 창 5개와 과제에서 제시한 출력 조건으로 출력하는 코드이다.

■ 인식대상 멤버들의 디스크립터 저장 코드 - ‘db.py’

rpt_face.py db.py

```python
import numpy as np
import dlib
import cv2 as cv
import pickle
import os

model_path = 'data/'#'d:/dip/data/dlib_face_recog/'
pose_predictor_5_point = dlib.shape_predictor(model_path + "shape_predictor_5_face_landmarks.dat")
face_encoder = dlib.face_recognition_model_v1(model_path + "dlib_face_recognition_resnet_model_v1.dat")
detector = dlib.get_frontal_face_detector()

# 검색하고자 하는 인물 7인
mb_han_lst = ['유재석', '지석진', '김종국', '하하', '송지효', '전소민', '양세찬']
mb_eng_lst = ['Yoo', 'ji', 'kim', 'ha', 'song', 'jeon', 'yang']
mb_index = [0,1,2,3,4,5,6]

# face_encodings은 128차원의 ndarray 데이터들을 사람 얼굴에 따라 리스트 자료형과 얼굴 위치좌표를 반환
def face_encodings(face_image, number_of_times_to_upsample=1, num_jitters=2):
    """Returns the 128D descriptor for each face in the image"""
    # Detect faces:
    # hog 기반의 dlib face detector. gray 변환된 영상을 사용.
    gray = cv.cvtColor(face_image, cv.COLOR_RGB2GRAY)
    face_locations = detector(gray, number_of_times_to_upsample)
    # Detected landmarks:
    raw_landmarks = [pose_predictor_5_point(face_image, face_location) for face_location in face_locations]
    # Calculate the face encoding for every detected face using the detected landmarks for each one:
    return face_locations, [np.array(face_encoder.compute_face_descriptor(face_image, raw_landmark_set, num_jitters))
                            for raw_landmark_set in raw_landmarks]


def saveObjects(file_name, data):       # data를 지정한 이름의 파일 file_name에 저장하기
    with open(file_name, "wb") as file:
        pickle.dump(data, file)

def loadObjects(file_name):             # 지정한 파일, file_name에서 읽은 객체를 반환하기
    with open(file_name, "rb") as file:
        return(pickle.load(file))

db = []
db_faces=[]
db_encoding1d=[]
db_labels=[]
for i in os.listdir('FaceImages/'):
    path = 'FaceImages/'+i
    img = cv.imread(path)
    img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    faces, known_image_encoding = face_encodings(img)
    img_face = img[faces[0].top():faces[0].bottom(), faces[0].left():faces[0].right(),:]
    db_faces.append(img_face)
    db_encoding1d.append(known_image_encoding[0])
    db_labels.append(i[0:i.find('_')])
    print(i)

db.append(db_faces)
db.append(db_encoding1d)
db.append(db_labels)

# 저장
saveObjects("db.bin", db)
```

각 멤버별로 고유 인덱스를 정해 이미지를 읽고 인코딩해 pickle파일로 저장한다.

4. 소스코드

[rpt_face.py]

rpt_face.py × db.py ×

```python
import ...
# 필수 사항 ----------------------------
# 아래의 변수이름과 표현식은 바꾸지 말고 소스 그대로 상단에 노출되게 해 주십시오.

# 1) 테스트할 영상 파일이 있는 드라이브와 폴더. 실제 평가에서는 바꾸어 사용할 수 있습니다.
test_img_path = 'data/'#'d:/dip/images/'

# 2) 그 폴더 안에 있는 테스트 대상 파일 이름
# 이 파일은 런닝맨 멤버와 아닌 사람들이 섞여 있습니다.
# 실제 평가에서는 바꾸어 사용할 수 있습니다. 테스트 파일이 10여개가 될 수도 있습니다.
test_file_list = ['t1.jpg', 't2.jpg', 't3.jpg', 't4.jpg', 't5.jpg']

# 3) path for dlib model: 위치 바꾸지 말고 제출해 주세요.
# 평가자의 PC에 설치해 놓고 테스트 할 것입니다.
model_path = 'data/'#'d:/dip/data/dlib_face_recog/'

# 4) shape predictor와 사용하는 recognition_model
# 아래 정의된 것을 그대로 사용해 주세요.
pose_predictor_5_point = dlib.shape_predictor(model_path + "shape_predictor_5_face_landmarks.dat")
face_encoder = dlib.face_recognition_model_v1(model_path + "dlib_face_recognition_resnet_model_v1.dat")

# 검색하고자 하는 인물 7인
mb_han_lst = ['유재석', '지석진', '김종국', '하하', '송지효', '전소민', '양세찬']
mb_eng_lst = ['Yoo', 'ji', 'kim', 'ha', 'song', 'jeon', 'yang']
md_dic = {'Yoo':'유재석', 'ji':'지석진', 'kim':'김종국', 'ha':'하하',
          'song':'송지효', 'jeon':'전소민', 'yang':'양세찬'}

# threshold보다 큰 유클리디언 거리는 동일 인물로 판단하지 않음.
threshold = 0.6

# 얼굴검출 dlib hog face detector 사용.
detector = dlib.get_frontal_face_detector()

# face_encodings은 128차원의 ndarray 데이터들을 사람 얼굴에 따라 리스트 자료형과 얼굴 위치좌표를 반환
def face_encodings(face_image, number_of_times_to_upsample=1, num_jitters=2):
    """Returns the 128D descriptor for each face in the image"""
    # Detect faces:
    # hog 기반의 dlib face detector. gray 변환된 영상을 사용.
    gray = cv.cvtColor(face_image, cv.COLOR_RGB2GRAY)
    face_locations = detector(gray, number_of_times_to_upsample)
    # Detected landmarks:
    raw_landmarks = [pose_predictor_5_point(face_image, face_location) for face_location in face_locations]
    # Calculate the face encoding for every detected face using the detected landmarks for each one:
    return face_locations, [np.array(face_encoder.compute_face_descriptor(face_image, raw_landmark_set, num_jitters))
                            for raw_landmark_set in raw_landmarks]

# unknown encoding을 known_encoding list에 비교해 유클리디어 거리를 계산해 이름과 반환하고, 거리는 오름차순 정렬해 반환한다.
def compare_faces_ordered(dbfaces, encodings, face_names, encoding_to_check):
    """Returns the ordered distances and names when comparing a list of face encodings against a candidate to check"""
    distances = list(np.linalg.norm(encodings - encoding_to_check, axis=1))
    return zip(*sorted(zip(distances,dbfaces, face_names)))

# show_detection은 이미지에 있는 사람 얼굴에 빨간 사각형과 검색된 사람의 영문 이름을 출력해 반환
def show_detection(image, face, text, num):
    """Draws a rectangle and Eng name over each detected face"""
    cv.rectangle(image, (face.left(), face.top()), (face.right(), face.bottom()), (0, 0, 255), 2)
    cv.putText(image,text, (face.left()-10,face.top()-3), cv.FONT_HERSHEY_DUPLEX, 1, (255,0,0),2)
    cv.putText(image, str(num), (face.left() - 30, face.top() + 30), cv.FONT_HERSHEY_DUPLEX, 1, (255, 0, 0), 2)
    return image

def loadObjects(file_name):             # 지정한 파일, file_name에서 읽은 객체를 반환하기
    with open(file_name, "rb") as file:
        return(pickle.load(file))
```

```
start = time.time()

# 검색 대상 파일 즉 미지의 사진을 로드.
unknown_images = []
for i in range(len(test_file_list)):
    unknown_image = cv.imread(test_img_path+test_file_list[i]) # image read
    unknown_image = unknown_image[:, :, ::-1]         # Convert image from BGR (OpenCV format) to RGB (dlib format)
    unknown_images.append(unknown_image)              # append image to unknown_images list

# 검색 대상 파일 사진에 대해 encoding
unknown_encodings = []
unknown_rects = []
for image in unknown_images:
    unknown_image_rects, unknown_image_encoding = face_encodings(image)
    unknown_encodings.append(unknown_image_encoding)  # 검색 대상 파일 사진에 여러사람이 존재 가능
    unknown_rects.append(unknown_image_rects)         # 찾은 얼굴의 위치 좌표값 삽입

# 인물 데이터 읽기
all_member_face, all_member_encodings1d_lst, all_member1d_label_lst = loadObjects("db.bin")

# 5장의 검색 대상 파일에 대해 얼굴 인식, 얼굴 비교 후 matplotlib창 띄워기 및 출력창에 정보 출력
for i ,ue in enumerate(unknown_encodings):
    total_count = len(ue) # 총 감지된 얼굴 수
    member_count=0        # 인식된 런닝맨 맴버수
    img_faces = unknown_images[i].copy()
    rects = unknown_rects[i]
    text=[]
    extra=[]
    for m, ue_face in enumerate(ue): # 감지된 얼굴에 대해
        computed_distances_ordered, faces, ordered_names = compare_faces_ordered(all_member_face,
                                    all_member_encodings1d_lst, all_member1d_label_lst, ue_face) # 비교
        if(computed_distances_ordered[0]<threshold): # 유클리디안 거리가 0.6인 경우
            text.append([mb_eng_lst[int(ordered_names[0])],computed_distances_ordered[0]])      # 그 사람 이름 추가
            member_count+=1                    # 인식된 맴버수 변수 1 증가
            unface = unknown_images[i][rects[m].top():rects[m].bottom(),rects[m].left():rects[m].right()]
            info = [[unface,-1],[faces[0], computed_distances_ordered[0]],[faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)
        else: # unknown인 경우
            text.append(['unknown',-1])
            unface = unknown_images[i][rects[m].top():rects[m].bottom(), rects[m].left():rects[m].right()]
            info = [[unface, -1], [faces[0], computed_distances_ordered[0]], [faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)

    for j, ue_rect in enumerate(unknown_rects[i]):              # 인식된 사람 얼굴에 대해
        img_faces = show_detection(img_faces, ue_rect, text[j][0], j) # 사각형과 이름 출력
    plt.figure(num=f'{test_file_list[i]}')
    plt.imshow(img_faces)
    title = f'[File: {test_file_list[i]}] {total_count} faces found including {member_count} members'
    plt.title(title, fontsize=20)
    plt.axis('off')
    mng = plt.get_current_fig_manager()
    mng.window.state('zoomed')           # 창 최대화
    # 이미지 분석 창
    plt.figure(num=f'{test_file_list[i]} - analysis')
    for index, j in enumerate(extra,1):
        for k in range(4):
            img_index = index+k*(total_count)
            plt.subplot(4,total_count,img_index)
            if k!=0: plt.title(round(j[k][1],3))
            plt.axis('off')
            plt.imshow(j[k][0])
    mng = plt.get_current_fig_manager()
    mng.window.state('zoomed')           # 창 최대화
    # 주어진 형식으로 정보 출력
    print(f'[File: {test_file_list[i]}] ------------------------------')
    for k, info in enumerate(text):
        if info[1]==-1:
            print(f'face {k}: {info[0]}')
        else: print(f'face {k}: {info[0]}={round(info[1],3)}')
    print(title)
    print()
print("\n실행시간 : ", time.time()-start)
plt.show()
```

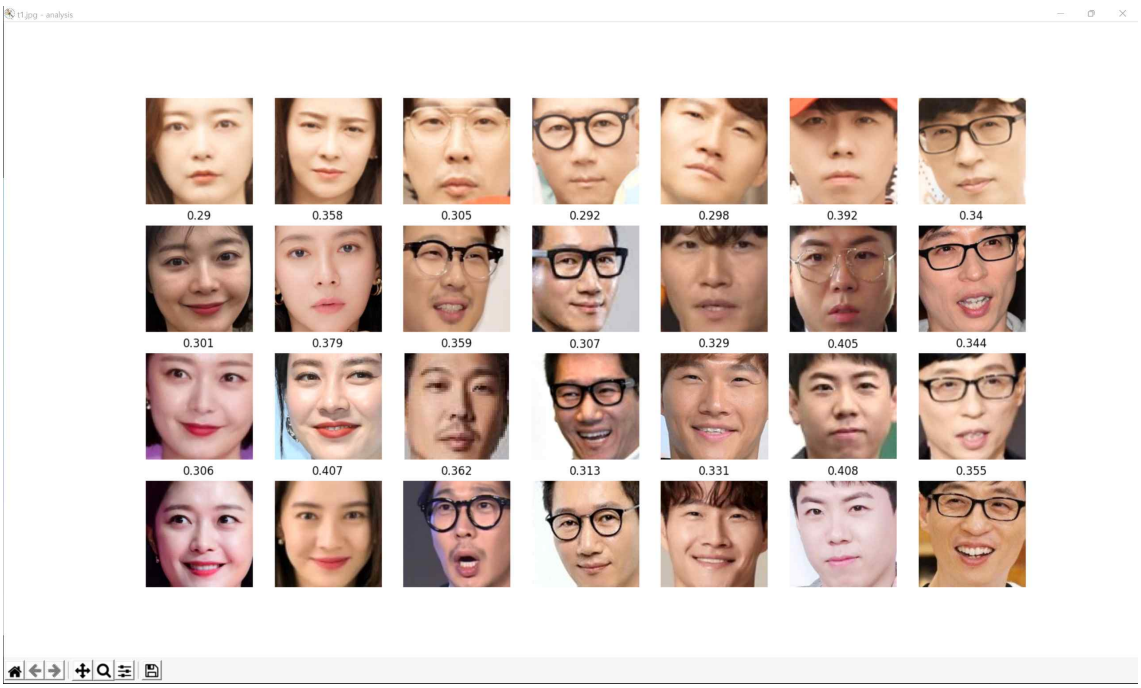
[db.py] - 'rpt_face.py'파일 하단에 주석처리해서 제출함.

rpt_face.py db.py

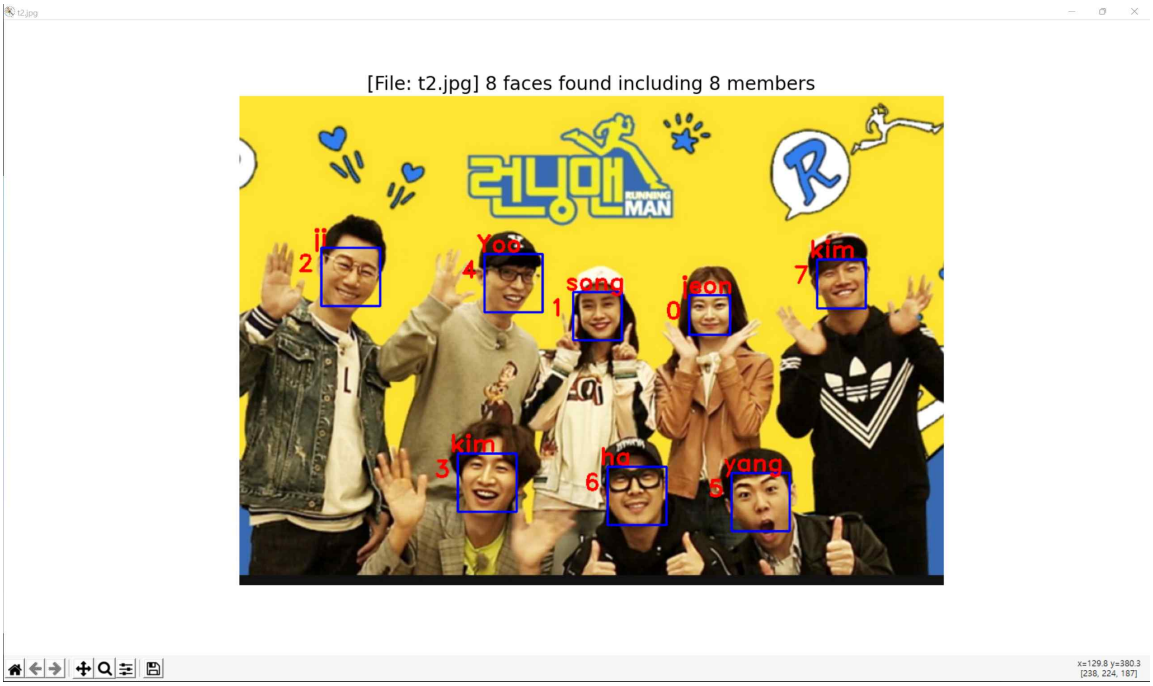
```python
import numpy as np
import dlib
import cv2 as cv
import pickle
import os

model_path = 'data/'#'d:/dip/data/dlib_face_recog/'
pose_predictor_5_point = dlib.shape_predictor(model_path + "shape_predictor_5_face_landmarks.dat")
face_encoder = dlib.face_recognition_model_v1(model_path + "dlib_face_recognition_resnet_model_v1.dat")
detector = dlib.get_frontal_face_detector()

# 검색하고자 하는 인물 7인
mb_han_lst = ['유재석', '지석진', '김종국', '하하', '송지효', '전소민', '양세찬']
mb_eng_lst = ['Yoo', 'ji', 'kim', 'ha', 'song', 'jeon', 'yang']
mb_index = [0,1,2,3,4,5,6]

# face_encodings은 128차원의 ndarray 데이터들을 사람 얼굴에 따라 리스트 자료형과 얼굴 위치좌표를 반환
def face_encodings(face_image, number_of_times_to_upsample=1, num_jitters=2):
    """Returns the 128D descriptor for each face in the image"""
    # Detect faces:
    # hog 기반의 dlib face detector. gray 변환된 영상을 사용.
    gray = cv.cvtColor(face_image, cv.COLOR_RGB2GRAY)
    face_locations = detector(gray, number_of_times_to_upsample)
    # Detected landmarks:
    raw_landmarks = [pose_predictor_5_point(face_image, face_location) for face_location in face_locations]
    # Calculate the face encoding for every detected face using the detected landmarks for each one:
    return face_locations, [np.array(face_encoder.compute_face_descriptor(face_image, raw_landmark_set, num_jitters))
                            for raw_landmark_set in raw_landmarks]


def saveObjects(file_name, data):       # data를 지정한 이름의 파일 file_name에 저장하기
    with open(file_name, "wb") as file:
        pickle.dump(data, file)

def loadObjects(file_name):             # 지정한 파일, file_name에서 읽은 객체를 반환하기
    with open(file_name, "rb") as file:
        return(pickle.load(file))

db = []
db_faces=[]
db_encoding1d=[]
db_labels=[]
for i in os.listdir('FaceImages/'):
    path = 'FaceImages/'+i
    img = cv.imread(path)
    img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
    faces, known_image_encoding = face_encodings(img)
    img_face = img[faces[0].top():faces[0].bottom(), faces[0].left():faces[0].right(),:]
    db_faces.append(img_face)
    db_encoding1d.append(known_image_encoding[0])
    db_labels.append(i[0:i.find('_')])
    print(i)

db.append(db_faces)
db.append(db_encoding1d)
db.append(db_labels)

# 저장
saveObjects("db.bin", db)
```

Ⅲ. 실행결과

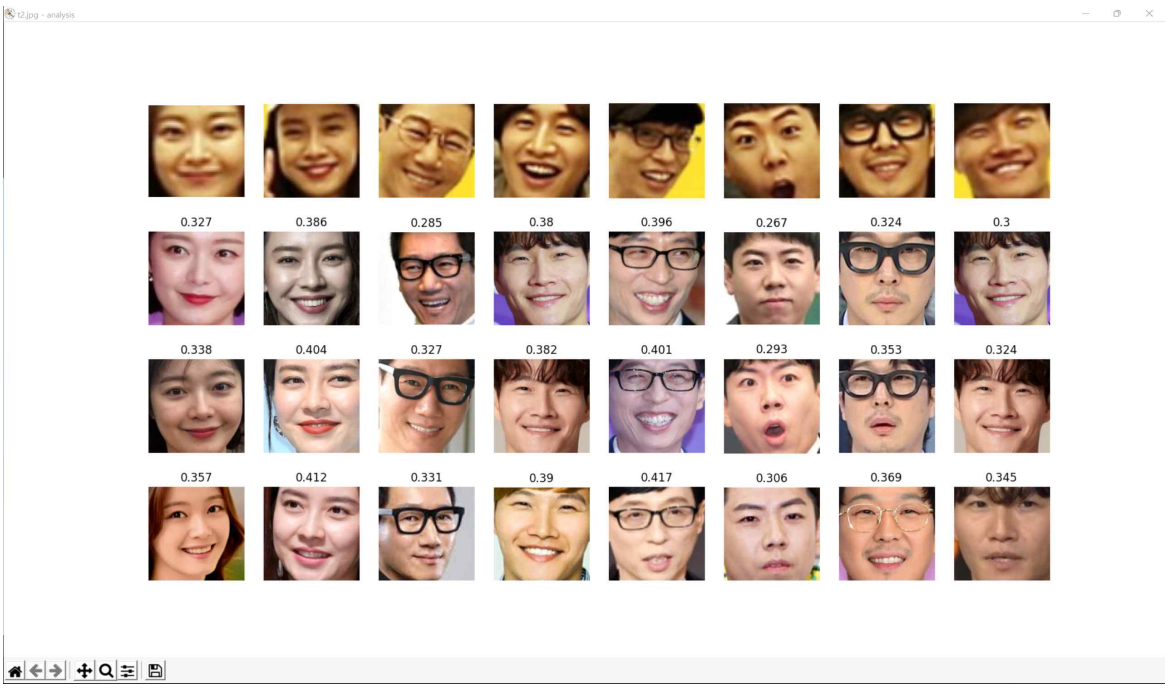
얼굴인식 수, 멤버 얼굴 인식 수, 각 인물 별 모두 정확히 인식을 한다.



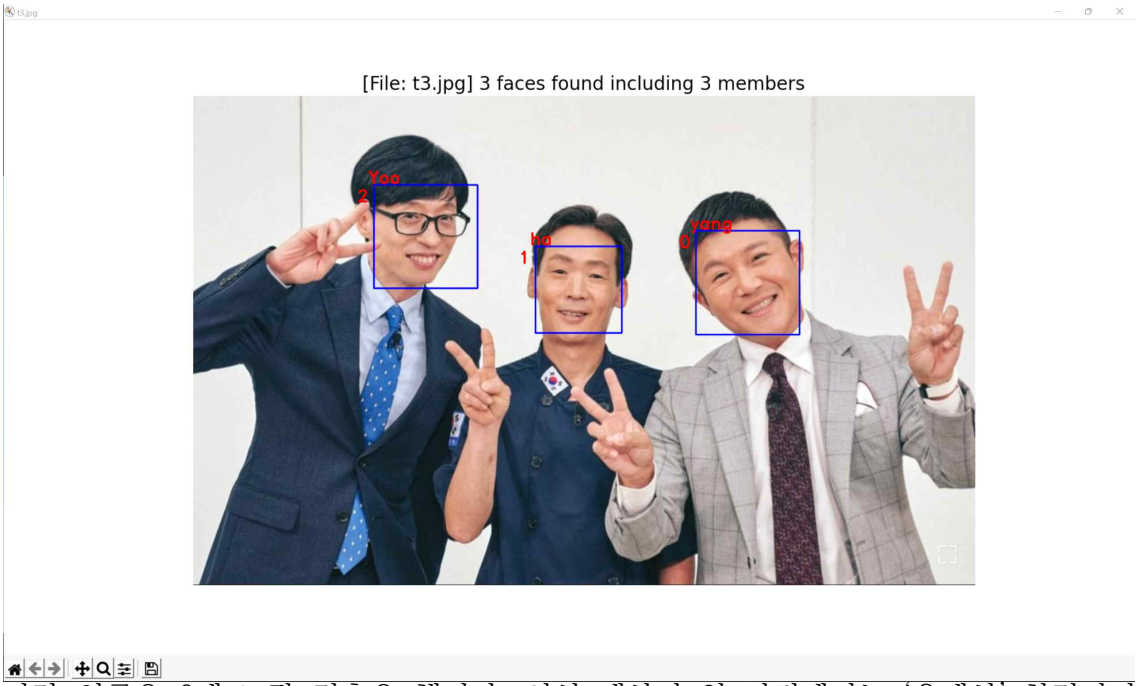
검출된 인물 별로 그 사람이라고 판단하게 된 얼굴을 3순위까지 선택해 유클리디언 거리 순으로 표시했다. 정상적으로 출력된다. 모든 사진이 유클리디언 거리 0.4이하로 매우 좋은 성능으로 인물을 인식한다.



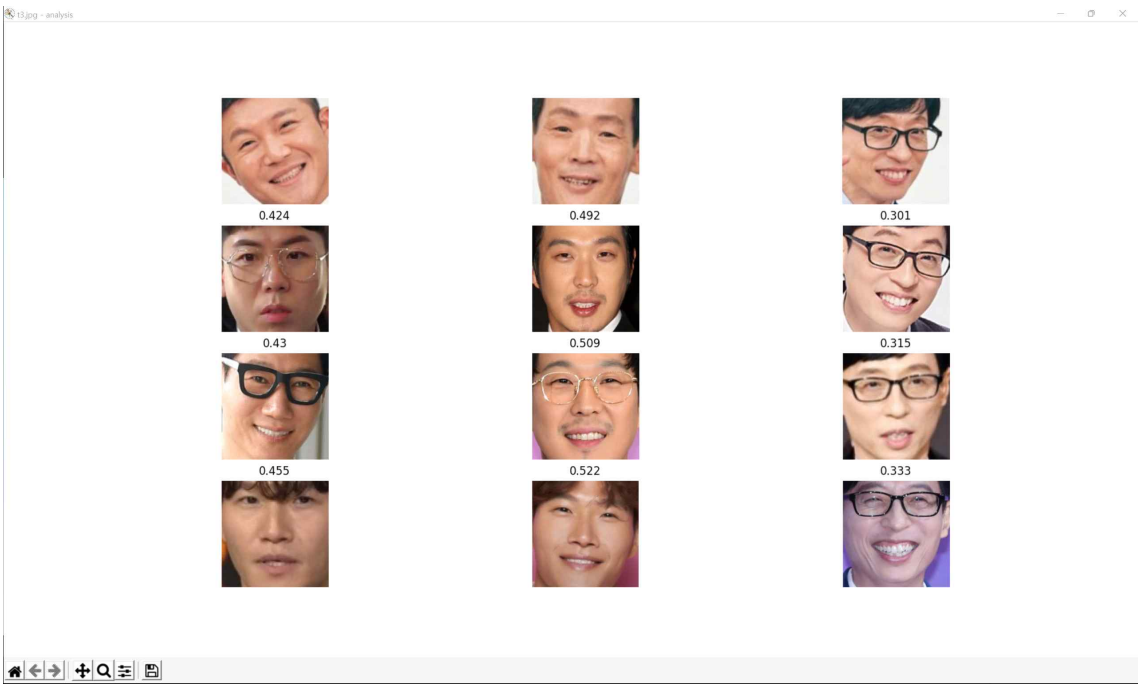
8명 사람에 대해 얼굴인식을 잘 한다. 하지만 인식 대상아 아닌 '이광수'가 '김종국'으로 인식되는 한계점이 있다. 나머지 멤버에 대해서는 정확하게 사람 매칭을 한다.

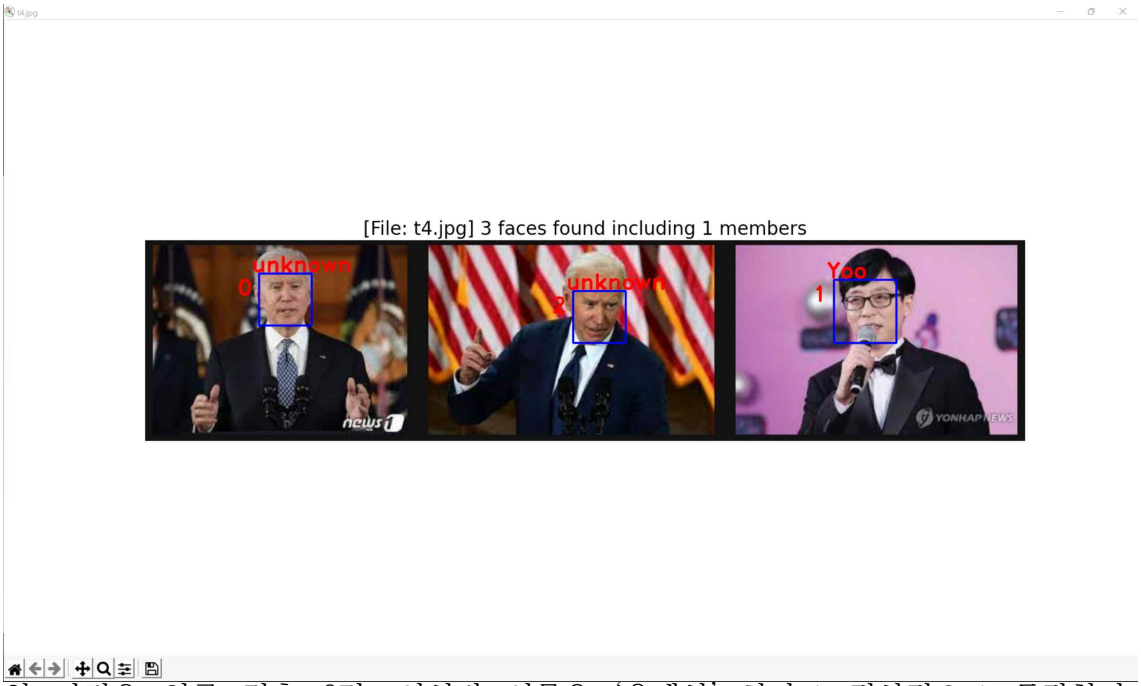


잘못인식한 '이광수'를 보면 김종국으로 잘못 인식하는 한계점이 있다.

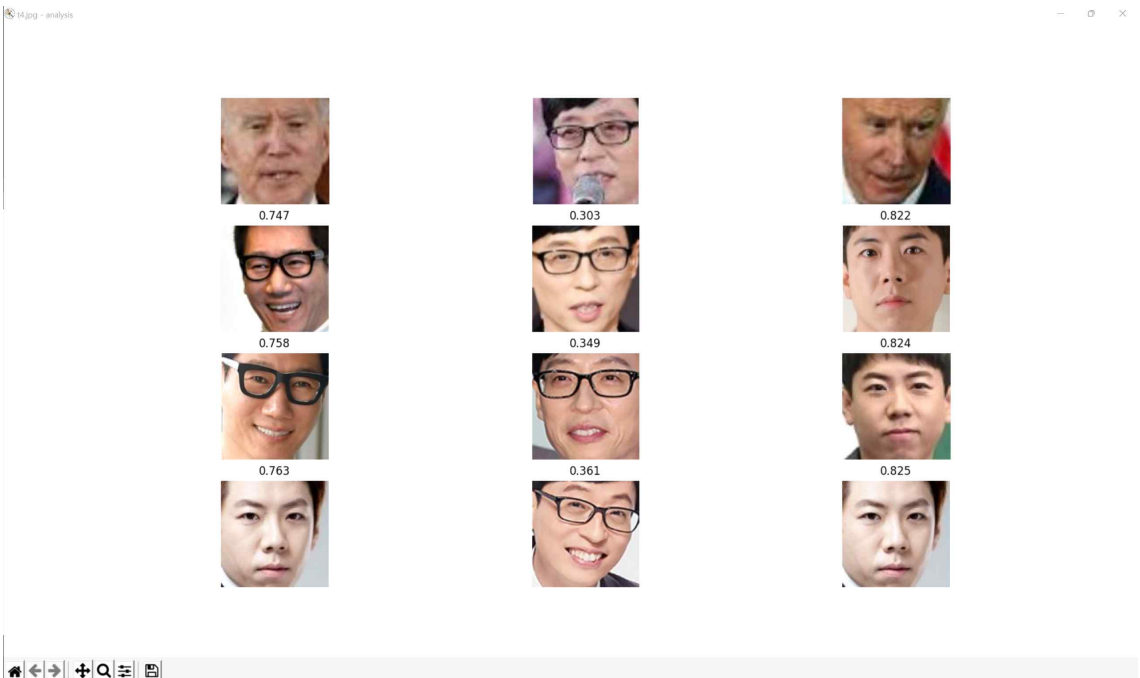


사람 얼굴은 3개로 잘 검출은 했지만, 인식 대상이 위 사진에서는 '유재석' 한명이지만, 나머지 두 사람에 대해서 '하하'와 '양세찬'으로 잘 못 인식되는 한계점이 있다.





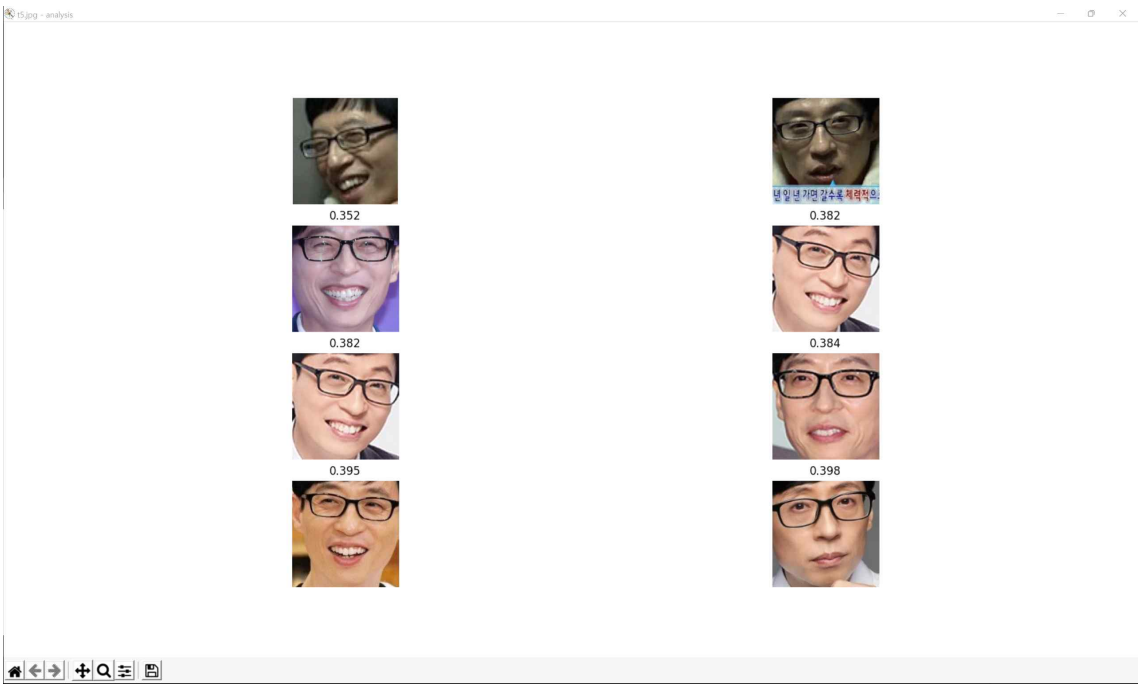
위 사진은 얼굴 검출 3명, 인식된 인물은 '유재석' 하나로 정상적으로 동작한다. '바이든'의 경우 런닝맨 멤버 7인 중 하나가 아니므로 'unknown'문구가 사진에 출력된다.



'unknwon'인 '바이든'의 경우 가장 가깝게 예측한 유클리디안거리가 0.7이상으로 정상적으로 동작하고 있다.

3번 결과창과 같이 한국인을 대상으로 한다면 인식되지 말아야할 대상에 대해 잘못 인식되지만, 외국인을 대상으로 한 사진에서는 원하는 결과가 나온다.

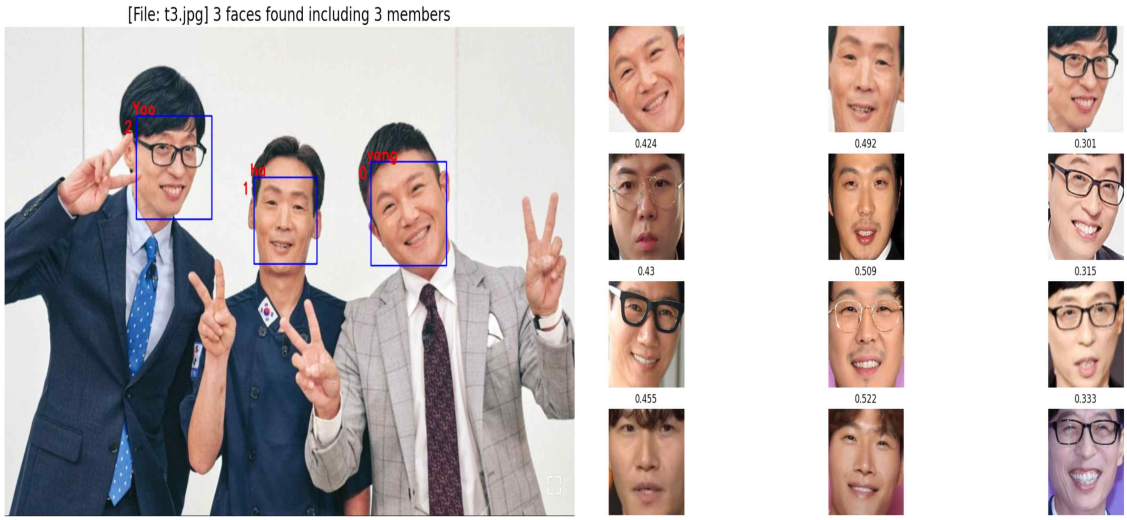
한 사진에 인식 대상인 특정 사람이 2명이상 있는 경우에 정상적으로 출력이 된다. 위 사진의 경우 정상적으로 인식을 한다. 위 사진의 경우 인식 대상 '유재석'이 두명 있는데 두명 모두 '유재석'으로 잘 인식한다.



출력창을 보면 문제에서 주어진 조건으로 정상적으로 출력된다.

추가적으로 실행시간도 측정해보았다.
Intel i5 cpu, RAM 8GB 기준 22초정도 소요된다.

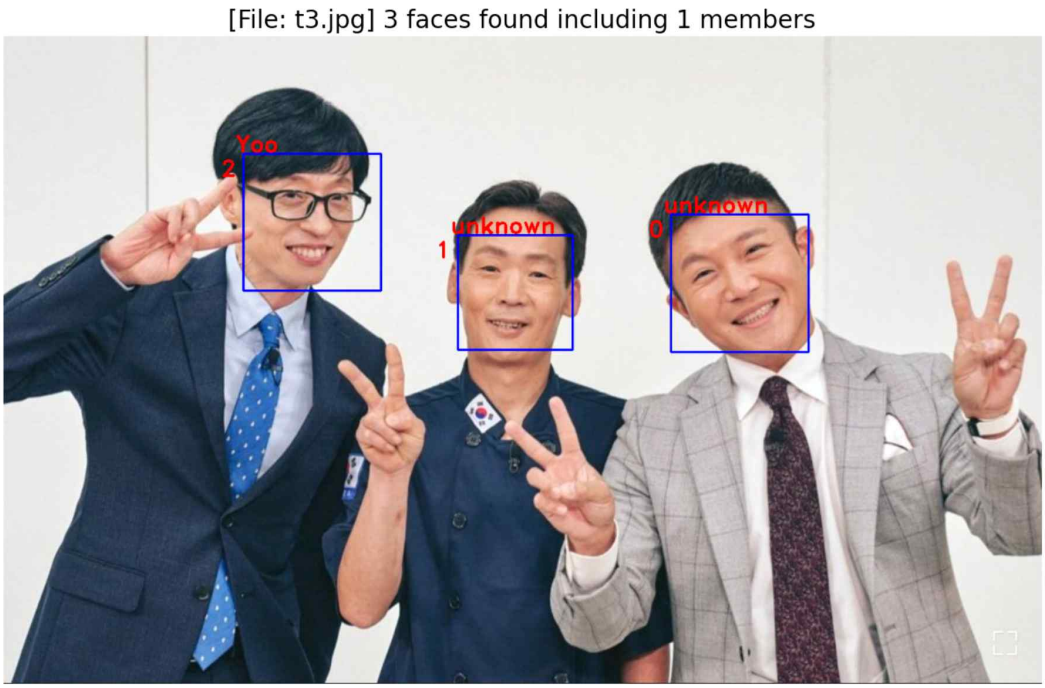
Ⅳ. 알고리즘 개선



위에서 실행결과를 보면 unknown으로 표시해할 인물에 대해서 잘못 인식되고 있다. 이미지 분석결과를 바탕으로 기존의 알고리즘을 수정했다.

```
if computed_distances_ordered[0]<threshold: # 유클리디안 거리가 0.6인 경우
    if computed_distances_ordered[0]>0.4:
        if not (ordered_names[0]==ordered_names[1] and ordered_names[1]==ordered_names[2]):
            text.append(['unknown', -1])
            unface = unknown_images[i][rects[m].top():rects[m].bottom(), rects[m].left():rects[m].right()]
            info = [[unface, -1], [faces[0], computed_distances_ordered[0]],
                    [faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)
        else:
            text.append([mb_eng_lst[int(ordered_names[0])], computed_distances_ordered[0]])  # 그 사람 이름 추가
            member_count += 1  # 인식된 멤버수 변수 1 증가
            unface = unknown_images[i][rects[m].top():rects[m].bottom(), rects[m].left():rects[m].right()]
            info = [[unface, -1], [faces[0], computed_distances_ordered[0]],
                    [faces[1], computed_distances_ordered[1]],
                    [faces[2], computed_distances_ordered[2]]]
            extra.append(info)
    else:
        text.append([mb_eng_lst[int(ordered_names[0])], computed_distances_ordered[0]])  # 그 사람 이름 추가
        member_count += 1  # 인식된 멤버수 변수 1 증가
        unface = unknown_images[i][rects[m].top():rects[m].bottom(), rects[m].left():rects[m].right()]
```

유클리디안 거리가 0.6이하의 설정은 동일하다. 다만 여기서 0.4 이상인 인물에 대해서 그 사람이라고 판단하게 된 얼굴을 3순위까지 라벨을 비교해 동일한 라벨로 판단할 경우에만 그 사람으로 인식하게 했다.



알고리즘을 수정한 결과 원하는 결과가 나왔다.

하지만 아직 한계점도 있다. 여전히 이광수를 김종국으로 인식한다. 그 이유를 보면 애초에 이광수에 대해 유클리디안 거리가 0.4 이하이고 만약 0.4이상이라 하더라도 그 사람이라고 판단하게 된 얼굴을 3순위까지 라벨을 비교해 보면 모두 김종국으로 판단해 결국 김종국으로 잘못 인식된다.

Ⅴ. 시행착오 및 느낀점

■ 시행착오

□ matplotlib창 최대로 키우기

문제에서 matplotlib 창을 띄울 때 결과를 잘 확인하기 위해서 최대 크기로 띄우는 과정에서 시행착오를 겪었다.

```
--------------------------------------
mng = plt.get_current_fig_manager()
mng.window.state('zoomed')
--------------------------------------
```

위 코드가 과제에서 최종적으로 사용한 코드이다. 결과를 정상적으로 matplot창이 최대크기로 띄어진다.

총 3번의 시행착오를 겪었다.

```
--------------------------------------
mng.resize(*mng.window.maxsize())
--------------------------------------
```

위 코드를 실행하면 창이 과도하게 커져 디스플레이 크기보다 더 큰 창이 출력된다.

```
--------------------------------------
mng.window.showMaximized()
mng.frame.Maximize(True)
--------------------------------------
```

위 두 개의 코드 경우 각각 실행해보면 오류가 발생한다.

□ sorted

compare_faces_ordered 메소드에서 sorted를 한다음에 값을 반환하는데, 여기서 zip(*sorted(zip(distances, dbfaces, face_names)))를 해야한다. 하지만 처음에 zip(*sorted(zip(dbfaces, distances, face_names)))를 해서 distances를 기준으로 정렬을 못해 프로그램이 원하는 결과가 나오지 않았다.

■ 배운점

- face_recognition에 대해 알게되었다. 얼굴인식 모듈에는 OpenCV, Dlib, Face_recognition 등 다양한 방법이 가능하다는 것도 알게되었고, 실제로 Dlib를 사용해 얼굴을 인식하는 방법을 알게됨.

- matplotlib 창을 최대로 키우는 방법을 이번 과제를 통해 배움.

- dlib hog기반 detector를 통해 얼굴 검출 좌표에 대한 처리 방법에 대해 배웠다. 교수님이 영상소개 영상에서 알려준 방법도 있지만

```
-------------------------------------
top = max(0, faces[0].top())
bottom = min(faces[0].bottom(), img.shape[0])
left = max(0, faces[0].left())
right = min(faces[0].right(), img.shape[1])
-------------------------------------
```

이런 식으로 구하는 방법이 있다는 것을 알게되었다.

- pickle 모듈을 사용해 데이터를 저장, 불러오는 것을 배웠다.

■ 느낀점

공모전을 통해 Dlib 모듈을 사용한 얼굴 검출, 랜드마크 검출을 해본 경험이 있어 과제에 대한 이해도가 높았다. 다만 특정 사람의 얼굴을 인식해 검색하는 것은 처음 해보는 거라 매우 좋은 경험이라고 생각한다. 이 과제를 통해 알게된 지식으로 다른 공모전이나 프로젝트를 만들 때 유용하게 사용할 것 같다.