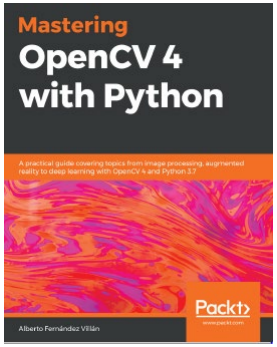
Face Recognition

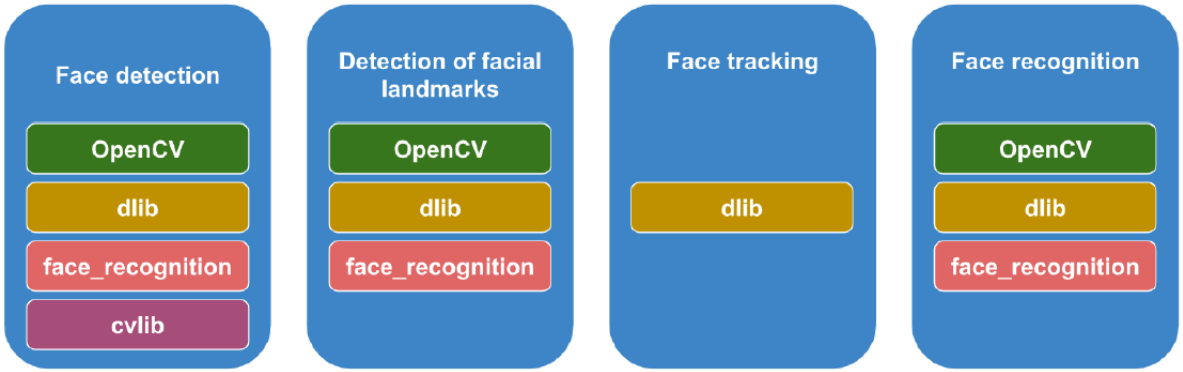
-dlib 중심으로..(교재 11장)



Mastering OpenCV 4 with Python, Alberto, Packt, Pub. 2019

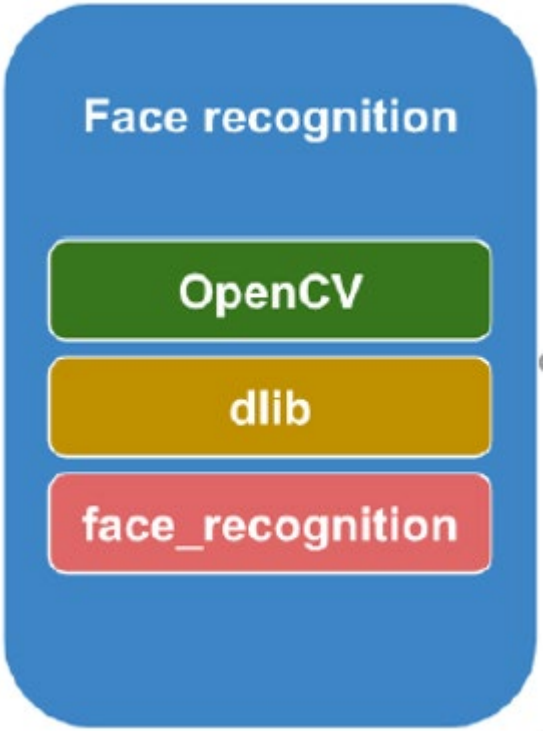
서경대학교 김진헌

얼굴 영상 처리 기술의 장르



- Face detection
  - 얼굴의 위치와 크기 검출
- Detection of facial landmark
  - 얼굴의 주요 부위 위치/크기 검출: 눈, 입, 코, 뺨 등..
- Face tracking
  - 움직이는 얼굴의 위치와 크기 검출
- Face recognition
  - Face identification(1: N): 등록된 얼굴 중에 누구인지를 맞추는 처리=>Access Control
  - Face verification(1:1): 자신이라고 주장하는 사람이 맞는지 맞추는 처리 => ATM

얼굴 인식 함수



- OpenCV
  - Eigenfaces
  - Fisherfaces
  - Local Binary Patterns Histograms(LBPH)
    - 조명조건을 고려한 설계로 실제 상황에서 3개의 접근 중 가장 우수. 새로운 얼굴을 등록하는 update() 메소드를 지원.
- Dlib(documentation and API reference, github)
  - DNN을 이용하여 얼굴을 128개의 벡터로 수치화. 미지의 얼굴을 128 벡터와의 유클리디안 거리로 동일 인물을 판단.
- Face_recognition
  - dlib의 인코딩과 face 비교 함수를 사용한다.

Programming with OpenCV

- 3가지(Eigenfaces, Fisherfaces, LBPH) 기술 모두 공통으로 recognizer를 생성하는 것으로 시작한다.

```
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
face_recognizer = cv2.face.EigenFaceRecognizer_create()
face_recognizer = cv2.face.FisherFaceRecognizer_create()
```

- 다음 2가지 메소드로 학습과 테스팅을 시행한다.
  - train() and predict()
  - face_recognizer.train(faces, labels)
    - LBPH의 경우: cv2.face_FaceRecognizer.update(src, labels)
      - 학습이 완료된 모델에 새로운 데이터를 기반으로 추가 학습
  - label, confidence = face_recognizer.predict(face)
- 학습된 모델을 저장하거나 로드하는 함수: write() and read() methods
  - cv2.face_FaceRecognizer.write(filename)
  - cv2.face_FaceRecognizer.read(filename)

Programming with dlib

1_encode_face_dlib.py

- Deep learning 기반으로 고품질 얼굴인식을 지원.
  - 실세계의 데이터베이스에 대해 99.38%의 정확도를 구현
- ResNet 34 뉴럴 모델을 기반으로 300만개의 얼굴에 대해 학습(21.4MB): 모델 다운로드
- 얼굴 특징을 나타내는 128 차원의 descriptor 생성. 유클리디언 거리가 0.6이하이면 같은 사람으로 간주한다. 모델 자체가 다른 사람과 구분되는 얼굴의 특징벡터를 출력하므로 사전에 특징 모델을 학습할 필요가 없다.
- 학습과정 요약
  - Triplets 단위로 학습: 3개의 영상 단위로 학습. 그중 2개는 같은 사람. 128개의 벡터가 같은 사람에 대해서는 가까워지도록 하고 이 같은 사람의 벡터들이 다른 사람에 대해서는 각각 멀어지도록 학습시킨다.
  - 이러한 학습을 수천명의 사람에 대해서 수백만개의 영상에 대해 반복하여 학습시킨다.
  - 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.

- 활용 방법
  - 1) 인식하고자 하는 인물들의 영상을 이용해 사전에 각 인물들의 descriptor들을 확보한다.
  - 미지의 인물에 대한 디스크립터를 구하여 이 벡터와 기존에 확보하고 있는 인물의 벡터와의 비교를 통해 어떤 인물인지 판단한다.
    - 판단 방법은 간단하게는 유클리디어 거리를 사용할 수 있다. ➔ 설계자의 역량 판단과 역량
  - 2) 인식하고자 하는 인물에 대한 dlib 디스크립터를 구한 후 동일인인지 판별하고자하는 인물에 대한 디스크립터를 구해 두 디스크립터의 거리가 일정 기준이하이면 동일인으로 처리한다.

dlib face descriptor로 identification

face_encodings은 128차원의 ndarray 데이터들을 사람 얼굴에 따라 리스트 자료형으로 반환하는데 그중 0번째를 반환 받는다. known 영상이 한 사람이므로...

5

2_compare_faces_dlib.py

```
known_image_1 = cv2.imread("jared_1.jpg")
known_image_2 = cv2.imread("jared_2.jpg")
known_image_3 = cv2.imread("jared_3.jpg")
known_image_4 = cv2.imread("obama.jpg")
unknown_image = cv2.imread("jared_4.jpg")
```

실험1

실험2

```
known_image_1 = cv2.imread("jared_1.jpg")
known_image_2 = cv2.imread("jared_2.jpg")
known_image_3 = cv2.imread("jared_3.jpg")
known_image_4 = cv2.imread("obama.jpg")
#unknown_image = cv2.imread("jared_4.jpg")
unknown_image = cv2.imread("obama2.jpg")
```

미지의 영상을 오바마2를 입력했을 때 위의 4개 영상 중 누구와 거리가 가까울 것인가?

```
known_image_1_encoding = face_encodings(known_image_1)[0]
known_image_2_encoding = face_encodings(known_image_2)[0]
known_image_3_encoding = face_encodings(known_image_3)[0]
known_image_4_encoding = face_encodings(known_image_4)[0]
known_encodings = [known_image_1_encoding, known_image_2_encoding,
                   known_image_3_encoding, known_image_4_encoding]
unknown_encoding = face_encodings(unknown_image)[0]
```

영상을 128개의 벡터로 인코딩하기

```
# unknown_encoding을 known_encodings list에 비교했을 때 유클리디어 거리를 계산하여 반환한다.
computed_distances = compare_faces(known_encodings, unknown_encoding)
```

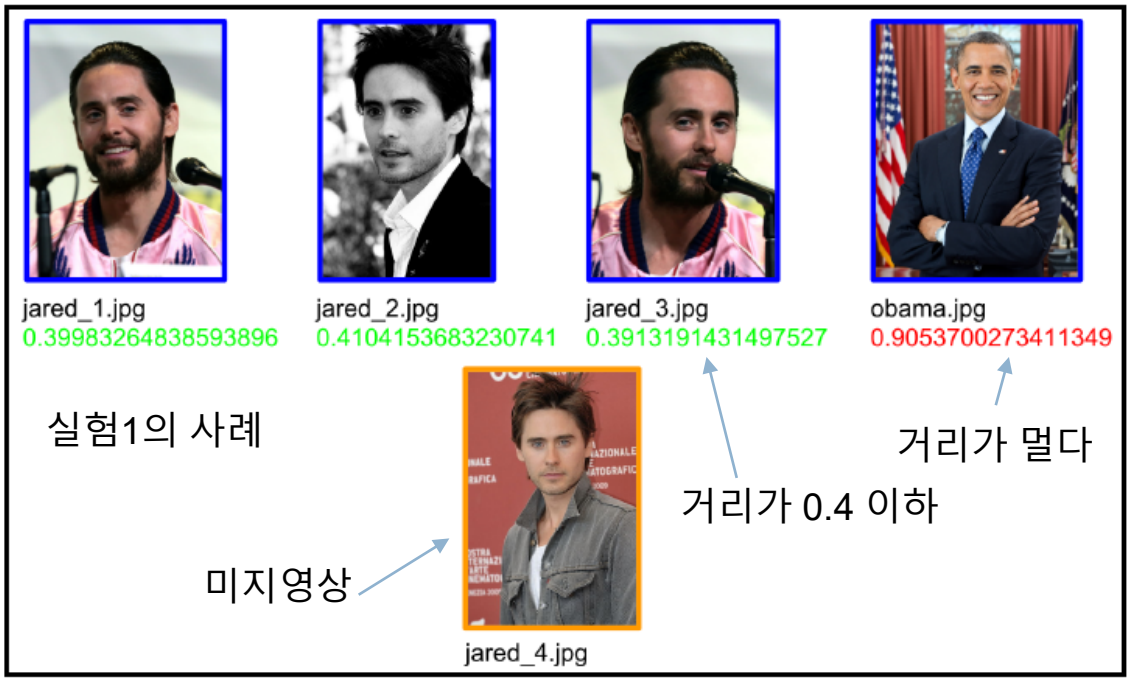
```
name            =['jared_1', 'jared_2', 'jared_3', 'obama']
matching distance=[0.39983264838593896, 0.4104153683230741, 0.3913191431497527, 0.9053700273411349]
```

실험1 결과 — 자레드와 거리가 가깝다.

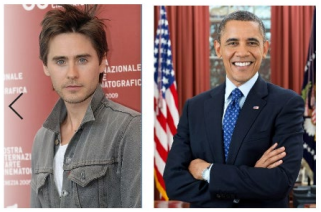
```
name            =['jared_1', 'jared_2', 'jared_3', 'obama']
matching distance=[0.8270609062142131, 0.8179289853067652, 0.8494330745541205, 0.46457711999062107]
```

실험2 결과 — 오바마와 거리가 가깝다.

Identification 실험 결과



실험 2에서 적용해 본 미지 영상들



두 인물을 묶어 미지 영상으로 입력(크기를 줄여 거리가 조금 달라졌음)

```
unknown_encoding = face_encodings(unknown_image)[0]
0.4117069012411471, 0.42171084693250943, 0.400383990511587, 0.9248907396564583

unknown_encoding = face_encodings(unknown_image)[1]
0.9076341335340578, 0.8717065226598436, 0.8927595657189682, 0.10158945347645236
```

과제 3: 사진속의 러닝맨 멤버 찾기

rpt_face.py

```
3차 과제: 사진 속에서 러닝맨 멤버찾기
    러닝맨 멤버가 포함된 사진에서 멤버를 찾아 그 인물의 얼굴에 사각형을
    그리고 그 상단에 해당 인물의 영문 이름을 적어 넣는다.

"""
# 1) 검색할 영상 파일이 있는 드리이브와 폴더
path = 'd:/dip/images/'
# 2) 그 폴더 안에 있는 검색 대상 파일 이름
test_file_list = ['t1.jpg', 't2.jpg', 't3.jpg', 't4.jpg', 't5.jpg']
# 3) 검색인물: 총 6인
mb_han_lst = ['유재석', '지석진', '김종국', '하하', '송지효', '전소민', '양세찬']
mb_eng_lst = ['Yoo', 'ji', 'kim', 'ha', 'song', 'jeon', 'yang']
# 4) path for dlib model
model_path = 'd:/dip/data/dlib_face_recog/'
# 5) shape predictor와 사용하는 recognition_model
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")
```

rpt_face.py

0. 소스 프로그램 작성 조건:

    위 1)~5)까지의 코드 부분은 위치는 재량이지만 수정없이 그대로 사용해야 합니다.

1. 출력하는 방법

    프로그램이 시작되면 별도의 키보드의 개입없이 **matplotlib** 창을 5개 열어 결과를 출력하고 종료합니다.

    이와는 별도로 출력창에는 검색된 얼굴의 수와 신원이 확인된 사람의 이름을 다음 사례와 같이 출력합니다.

[t1.jpg]
검색된 사람의 얼굴 수= 12, 러닝맨 멤버 수: 3, 확인된 인물=유재석, 지석진, 전소민
[t2.jpg]
검색된 사람의 얼굴 수= 3, 러닝맨 멤버 수: 1, 확인된 인물=양세찬
[t3.jpg]
검색된 사람의 얼굴 수= 0, 러닝맨 멤버 수: 0
[t4.jpg]
검색된 사람의 얼굴 수= 4, 러닝맨 멤버 수: 0
[t5.jpg]
검색된 사람의 얼굴 수= 8, 러닝맨 멤버 수: 3, 확인된 인물=양세찬, 김종국, 하하

rpt_face.py

K유클리디언 외에 NN을 시도해서 더 좋은 결과가 있었다면 가점합니다.

2. 본인 여부의 판단법

   128벡터의 유클리디안 거리가 0.6이하로 정합니다.

   같은 사람이 2인 이상 검출되더라도 같은 기준을 유지합니다.

3. 평가 요소

   1) 화면 출력이 올바르게 되고 있는가?

   2) 수행창 출력이 올바르게 되고 있는가?

4. 기타

   1) 활동된 알고리즘(얼굴 검출, 디스크립터 산출 관계)들의 성능한계는 배려합니다.

      한편, 주어진 조건하에서 성능을 높이기 위한 노력은 인정합니다.

      (예: 남들은 잘 안되었는데 자신의 프로그램은 인식을 잘 한다다던가...)

   2) 기타 궁금한 사항은 구글 클래스 룸에 문의 주세요..

마감일자: 12월 22일 23시 59분
제출물: 성명.zip(rpt_face.py+PDF 보고서+발표 동영상)
제출처: 구글 클래스룸 3차 과제