

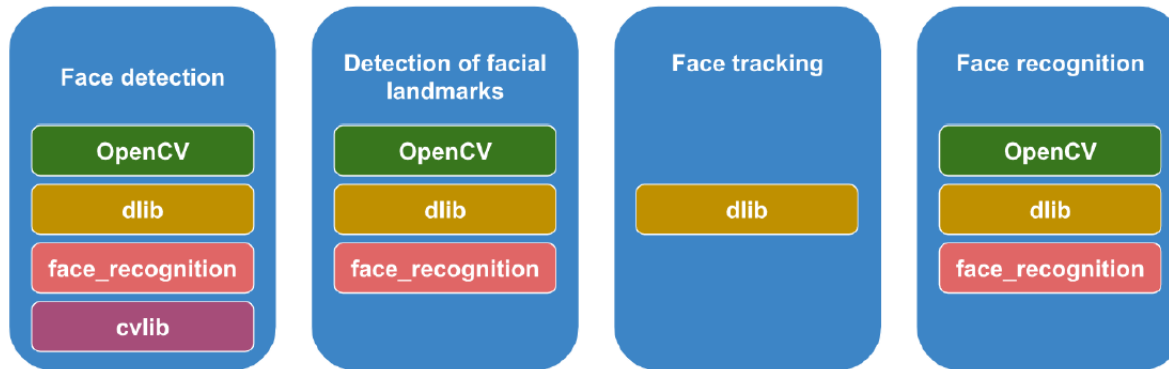
Face Recognition

- Haar Cascades & DNN

서경대학교 김진헌

얼굴 영상 처리 기술의 장르

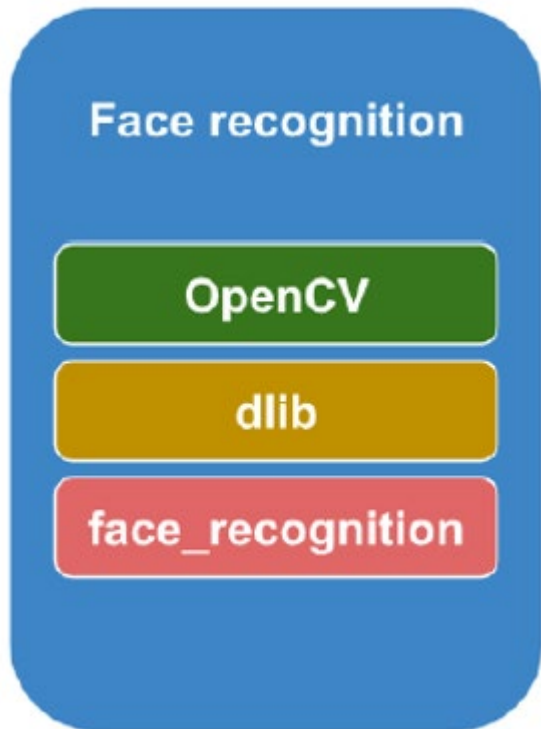
1



- 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

2



- 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

3

- 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 테스트를 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

4

- 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 유클리디언 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.

Programming with dlib

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

5

1_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")
```

```
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]
```

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

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.

Programming with dlib

6

encode_face_dlib.py

실험 2에서 적용해 본 미지 영상들

실험1의 사례

미지영상

jared_1.jpg 0.39983264838593896

jared_2.jpg 0.4104153683230741

jared_3.jpg 0.3913191431497527

obama.jpg 0.9053700273411349

jared_4.jpg 0.4117069012411471

거리가 0.4 이하

거리가 멀다

0.06769105174428224 0.31971101815206227

0.46457711999062107

0.4528653085126009 0.40278126602577996



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
```