Facial Landmarks Detection

using ~~OpenCV~~, dlib, face_recognition, ~~cvlib~~ modules

2021년 2학기

서경대학교 김진헌

face_recognition 모듈

1_landmarks_detection_fr.py

- face_recognition module은 내부적으로 dlib의 HOG, CNN detector 사용
- 실험에 의하면 pyCharm의 Setting에서는 설치할 수 없었다.
  - 설치 방법
    - 성공 사례(추천): pip로 설치 가능: pip install face_recognition
      - 사전에 cmake가 설치되어 있어야 하는데 이것도 pip 명령으로 설치하여 성공하였다: pip install cmake
    - 다른 사례: 수동으로 설치하는 방법(링크)도 있다.
      - 역시 cmake를 먼저 설치해야 한다.
        - Cmake 설치가 필요하고, visual studio가 미리 설치되어 있어야 할 것으로 추정된다. (확실치 않음).
      - C++ 컴파일러가 있어야 하는 것으로 알고 있음

face_recognition.face_landmarks(face_image, face_locations=None, model='large')

face_image – image to search

face_locations – Optionally provide a list of face locations to check.

model – Optional - which model to use. "large" (default)

or "small" which only returns 5 points but is faster.

large: 9개의 key(landmark)가 검출. small: 3개(양눈, 코)의 key가 검출

랜드 마크 검출 face_recognition 함수

1_landmarks_detection_fr.py

- *형식: face_landmarks_list = **face_recognition.face_landmarks**(face_image, face_locations=None, model='large')*
- 입력 파라미터
  - *face_locations: 얼굴이 있는 영역*
  - *model: 검출하는 랜드마크의 규모(크면 시간 소요)*
    - large: 9개의 key(landmark)를 검출. 왼쪽/오른쪽은 바라보는 사람의 입장에 따름.
      - [' chin ' , ' left_eyebrow ' , ' right_eyebrow ' , ' nose_bridge ' , ' nose_tip ' , ' left_eye ' , ' right_eye ' , ' top_lip ' , ' bottom_lip ']
    - small: 3개의 key가 검출
      - ['nose_tip', 'left_eye', 'right_eye']
- 반환값: 모델에 따라 9 혹은 3개의 검출된 얼굴의 랜드마크로 구성된 사전형 자료 x 검출된 얼굴의 수
  - 반환값 = [얼굴 0 랜드마크 사전, 얼굴 1 랜드마크 사전, 얼굴 2 랜드마크 사전, …]
  - 얼굴 랜드마크 사전 = {key=landmark: value=[(x1, y1), (x2, y2), …]}

Lab 1: 9/3 랜드마크 검출

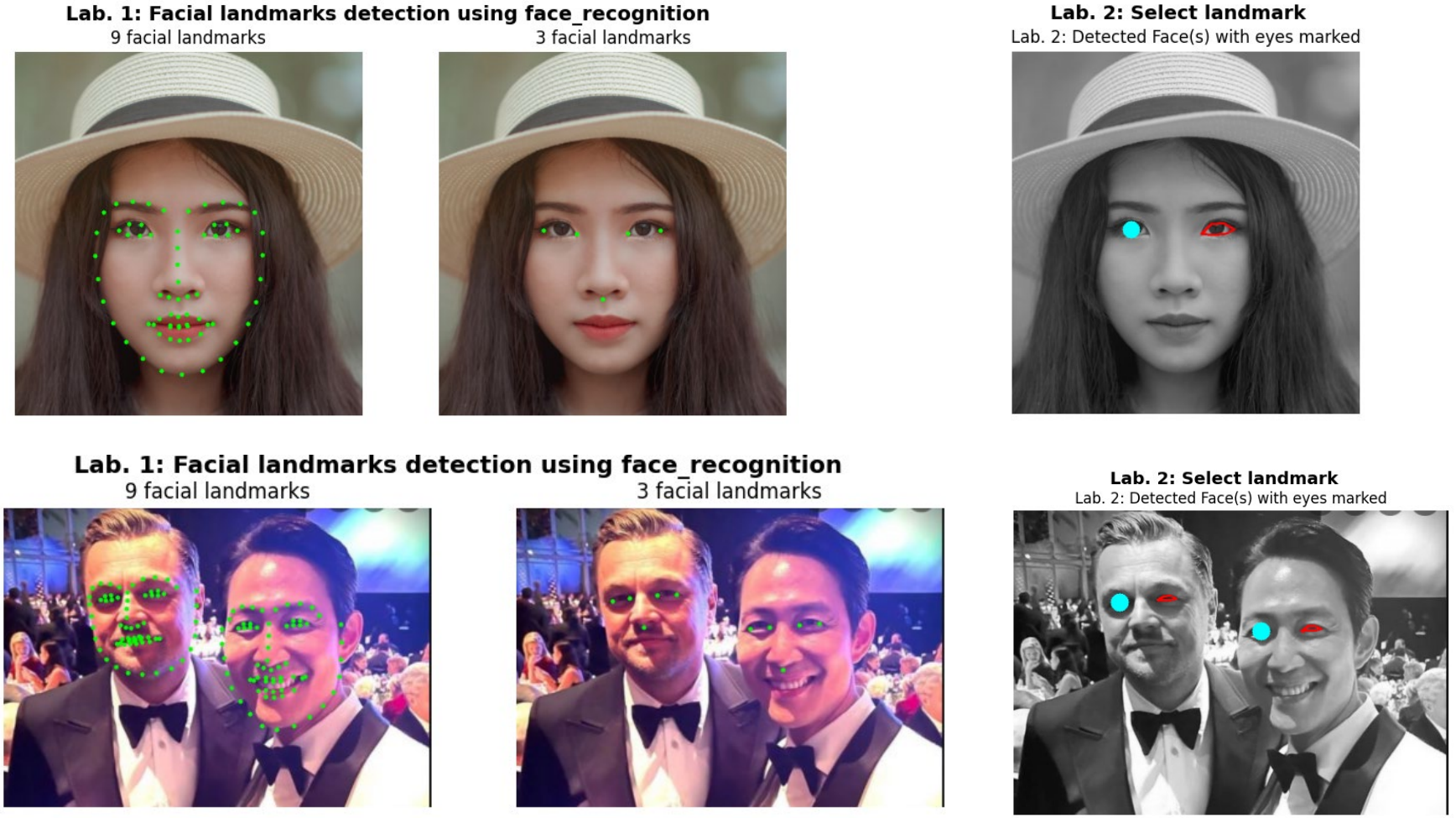
1_landmarks_detection_fr.py

- 9 detected landmark details: 사례 file = "Lee_DiCaprio.JPG"
  - 1) type(face_landmarks_list)=<class 'list'>
  - 2) len(face_landmarks_list)=number of faces detected=2
  - 3) type(face_landmarks_list[0])=<class 'dict'>
  - 4) face_landmarks_list[0].keys()=['chin', 'left_eyebrow', 'right_eyebrow', 'nose_bridge', 'nose_tip', 'left_eye', 'right_eye', 'top_lip', 'bottom_lip']
  - 5) face_landmarks_list[0]['left_eye']=[(315, 162), (322, 156), (331, 156), (340, 161), (331, 161), (323, 162)]

(좌) large model로 검출한 랜드마크 9개
(우) small model로 검출한 랜드마크 3개

Lab 2: 좌우 눈동자 구분 표시

1_landmarks_detection_fr.py



Lab 2: 왼쪽눈은 cyan circle로 오른쪽 눈은 red polygon으로 마킹한다.

dlib 모듈 기반

- **dlib 모듈**
  - *shape detector 데이터 파일이 필요하다.*
    - *=> path + "dlib_shape_predictors/" 아래에 있음*
  - 2_landmarks_detection_dlib_from_camera.py
    - 카메라의 입력영상으로 랜드마크 검출
  - 2_landmarks_detection_dlib_from_videofile.py

USB 카메라 기반 실시간 검출

2_landmarks_detection_dlib_from_camera.py

- 1) 얼굴 객체 생성: detector = dlib.get_frontal_face_detector()
- 2) 랜드마크 검출 객체 생성: predictor = dlib.shape_predictor(p)
  - 이때 p 파일은 랜드마크 객체를 생성하기 위해 정보를 담고 있는 shape predictor 파일이다.
  - p = "../../data/dlib_shape_predictors/shape_predictor_68_face_landmarks.dat"
- 3) 얼굴을 검출한다. => 4각형 영역(rect)을 반환 => 사각형으로 얼굴 영역을 표시한다.
  - rects = detector(gray, 0) # 모노 영상을 사용하였다. 화면 배율은 그대로 사용하는 것으로 설정.
  - cv2.rectangle(frame, (rect.left(), rect.top()), (rect.right(), rect.bottom()), (0, 255, 0), 1)
- 4) 검출된 얼굴 안에서 랜드마크를 검출한다. => 랜드마크 shape descriptor를 반환한다.
  - shape = predictor(gray, rect)
    - 반환받는 shape는 <class '_dlib_pybind11.full_object_detection'>형이다.
    - shape.num_parts 길이(68 혹은 9)의 랜드마크 정보를 담고 있다.
    - (shape.(i).x, shape.(i).y) 좌표 정보를 i=0~shape.num_parts 만큼 갖고 있다.
- 5) 랜드마크 디스크립터를 numpy array 정보로 바꾼다
  - shape = shape_to_np(shape) # shape.shape=(68, 2) 혹은 (9, 2)

USB 카메라 기반 실시간 검출

2_landmarks_detection_dlib_from_camera.py

- 6) 검출된 얼굴 영역과 랜드마크를 화면에 출력한다.

```
draw_shape_lines_range(shape, frame, RIGHT_EYE_POINTS, True)
draw_shape_lines_range(shape, frame, LEFT_EYE_POINTS, True)
draw_shape_lines_range(shape, frame, MOUTH_OUTLINE_POINTS, True)
draw_shape_lines_range(shape, frame, MOUTH_INNER_POINTS, True)
```

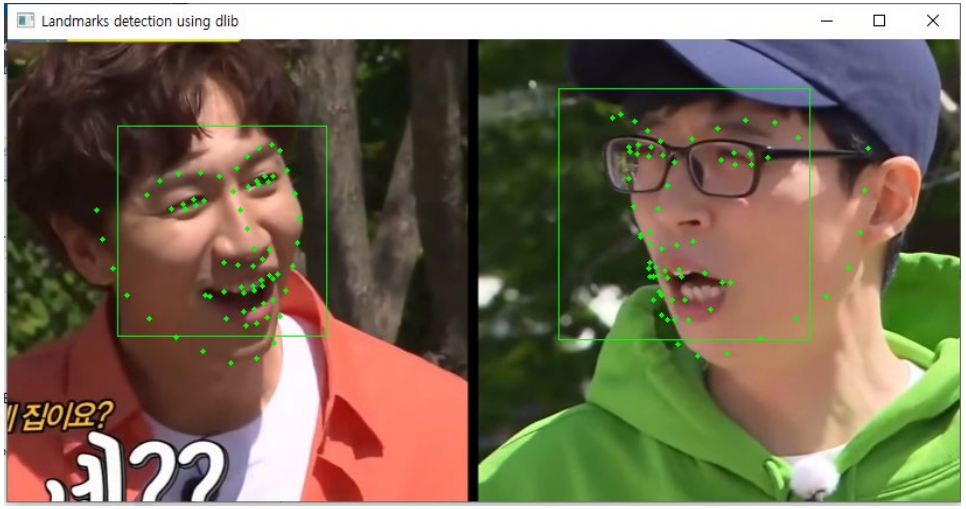
```
JAWLINE_POINTS = list(range(0, 17))
RIGHT_EYEBROW_POINTS = list(range(17, 22))
LEFT_EYEBROW_POINTS = list(range(22, 27))
NOSE_BRIDGE_POINTS = list(range(27, 31))
LOWER_NOSE_POINTS = list(range(31, 36))
RIGHT_EYE_POINTS = list(range(36, 42))
LEFT_EYE_POINTS = list(range(42, 48))
MOUTH_OUTLINE_POINTS = list(range(48, 61))
MOUTH_INNER_POINTS = list(range(61, 68))
ALL_POINTS = list(range(0, 68))
```


```
def draw_shape_lines_range(np_shape, image, range_points, is_closed=False):
    """Draws the shape using lines to connect the different points"""
    np_shape_display = np_shape[range_points]
    points = np.array(np_shape_display, dtype=np.int32)
    cv2.polylines(image, [points], is_closed, (255, 255, 0), thickness=1, lineType=cv2.LINE_8)
```

동영상 기반 실시간 검출

2_landmarks_detection_dlib_from_videofile.py

- 동영상에서 취득한 영상을 화면에 출력하면서 검출한 얼굴 영역과 랜드마크를 계속 표시한다.
  - dlib 함수로 얼굴을 검출하고, dlib 함수로 얼굴의 주요 랜드마크를 검출한다.
- Esc 를 입력하면 프로그램을 종료한다.
- 스페이스 바를 누르면 다시 키를 누를 때까지 잠시 정지한다.




```
path = '../../data/'                # 동영상이 존재하는 곳
file = '러닝맨 오프닝.mp4'
full_file_name = path + file

# Create VideoCapture object to get images from the webcam:
#video_capture = cv2.VideoCapture(0)              # USB CAM camera
video_capture = cv2.VideoCapture(full_file_name)    # video file
```