<부록 04. face overlay 소스 코드>

```
#!/usr/bin/env python
# coding: utf-8

# cv2 : openCV 이미지 처리 라이브러리
# dlib : 이미지 처리 라이브러리
# numpy : 행렬 연산 처리
#
# 샘플 영상 취득 : videos.pexels.com/search/face
#

# In[1]:


import cv2, dlib, sys
import numpy as np

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# --> 이미지가 너무 커서 문제가 있음


# In[5]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.2

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
```

```
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))

# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[1]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.1

# 얼굴 디텍터 모듈 초기화
detector = dlib.get_frontal_face_detector()
# 얼굴 특징점 모듈 초기화
# shape_predictor_68_face_landmarks.dat 파일은 머신러닝으로 학습된 데이터임
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
#원본 이미지 복사해 놓음.
    ori = img.copy()

    # detect faces 이미지에서 얼굴모양 찾기
    faces = detector(img)
    # 0번 인덱스, 첫번째 인식되는 얼굴 하나만 인식하기
    face = faces[0]



    # visualize # 얼굴영영에 네모 모양 이미지 그리지
    # face.left(), face.top() 좌상단
    # face.right(), face.bottom() 우하단
    # color=(255, 255, 255) 흰색
```

```
    # thickness = 2 2pixel 찍기
    # lineType=cv2.LINE_AA    : 선타입(직선) 지정

    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )
    print(face.left(), face.top(), face.right(), face.bottom())
# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[ ]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.1

# 얼굴 디텍터 모듈 초기화
detector = dlib.get_frontal_face_detector()
# 얼굴 특징점 모듈 초기화
# shape_predictor_68_face_landmarks.dat 파일은 머신러닝으로 학습된 데이터임
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
#원본 이미지 복사해 놓음.
    ori = img.copy()

    # detect faces 이미지에서 얼굴모양 찾기
    faces = detector(img)
    # 0번 인덱스, 첫번째 인식되는 얼굴 하나만 인식하기
    face = faces[0]


    # visualize # 얼굴영영에 네모 모양 이미지 그리지
    # face.left(), face.top() 좌상단
```

```
    # face.right(), face.bottom() 우하단
    # color=(255, 255, 255) 흰색
    # thickness = 2 2pixel 찍기
    # lineType=cv2.LINE_AA   : 선타입(직선) 지정

    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )

# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[ ]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.1

# 얼굴 디텍터 모듈 초기화
detector = dlib.get_frontal_face_detector()
# 얼굴 특징점 모듈 초기화
# shape_predictor_68_face_landmarks.dat 파일은 머신러닝으로 학습된 데이터임
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
#원본 이미지 복사해 놓음.
    ori = img.copy()

    # detect faces 이미지에서 얼굴모양 찾기
    faces = detector(img)
    # 0번 인덱스, 첫번째 인식되는 얼굴 하나만 인식하기
    face = faces[0]

    # 얼굴 영역에서 특징점 찾기
```

```
    dlib_shape = predictor(img, face)
    # dlib 객체를 numpy 객체로 변환
    shape_2d = np.array([[p.x, p.y] for p in dlib_shape.parts()])

    #computer center of face
    #얼굴 영역의 좌상단 좌표 구하기
    top_left = np.min(shape_2d, axis = 0)
    #얼굴 영역의 우하단 좌표 구하기
    bottom_right = np.max(shape_2d, axis = 0)

    # visualize # 얼굴영영에 네모 모양 이미지 그리지
    # face.left(), face.top() 좌상단
    # face.right(), face.bottom() 우하단
    # color=(255, 255, 255) 흰색
    # thickness = 2 2pixel 찍기
    # lineType=cv2.LINE_AA   : 선타입(직선) 지정
    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )

    # 얼국 특징점에 대해 흰색으로 원모양 찍기 (68개)
    for s in shape_2d:
        cv2.circle(img, center=tuple(s), radius=1, color=(255, 255, 255), thickness=2, lineType=cv2.LINE_AA)

    #좌상단 좌표에 원 모양의 좌표 표현하기 (파란색)
    cv2.circle(img, center=tuple(top_left), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
    #우하단 좌표에 원 모양의 좌표 표현하기 (파란색)
    cv2.circle(img, center=tuple(bottom_right), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)

# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[ ]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.1

# 얼굴 디텍터 모듈 초기화
detector = dlib.get_frontal_face_detector()
# 얼굴 특징점 모듈 초기화
# shape_predictor_68_face_landmarks.dat 파일은 머신러닝으로 학습된 데이터임
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

```
#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
#원본 이미지 복사해 놓음.
    ori = img.copy()

    # detect faces 이미지에서 얼굴모양 찾기
    faces = detector(img)
    # 0번 인덱스, 첫번째 인식되는 얼굴 하나만 인식하기
    face = faces[0]



    # 얼굴 영역에서 특징점 찾기
    dlib_shape = predictor(img, face)
    # dlib 객체를 numpy 객체로 변환
    shape_2d = np.array([[p.x, p.y] for p in dlib_shape.parts()])

    #computer center of face
    #얼굴 영역의 좌상단 좌표 구하기
    top_left = np.min(shape_2d, axis = 0)
    #얼굴 영역의 우하단 좌표 구하기
    bottom_right = np.max(shape_2d, axis = 0)

    # 얼굴의 중심점 찾기
    # .astype(np.int) -> 중심점이 소수점으로 나올수 있기 때문에 정수형으로 변환
    center_x, center_y = np.mean(shape_2d, axis=0).astype(np.int)

    # visualize # 얼굴영영에 네모 모양 이미지 그리지
    # face.left(), face.top() 좌상단
    # face.right(), face.bottom() 우하단
    # color=(255, 255, 255) 흰색
    # thickness = 2 2pixel 찍기
    # lineType=cv2.LINE_AA    : 선타입(직선) 지정
    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )

    # 얼국 특징점에 대해 흰색으로 원모양 찍기 (68개)
    for s in shape_2d:
        cv2.circle(img, center=tuple(s), radius=1, color=(255, 255, 255), thickness=2, lineType=cv2.LINE_AA)

    #좌상단 좌표에 원 모양의 좌표 표현하기 (파란색)
    cv2.circle(img, center=tuple(top_left), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
    #우하단 좌표에 원 모양의 좌표 표현하기 (파란색)
```

```
    cv2.circle(img, center=tuple(bottom_right), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)

    # 얼굴 중심점 표현하기 (빨강색)
    cv2.circle(img, center=tuple((center_x, center_y)), radius=1,  color=(0, 0, 255), thickness=2, lineType=cv2.LINE_AA)
# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[1]:


import cv2, dlib, sys
import numpy as np

# 이미지 크기 비율 조절
scaler = 0.1

# 얼굴 디텍터 모듈 초기화
detector = dlib.get_frontal_face_detector()
# 얼굴 특징점 모듈 초기화
# shape_predictor_68_face_landmarks.dat 파일은 머신러닝으로 학습된 데이터임
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

#동영상 파일 로드
cap = cv2.VideoCapture('Image.mp4')

# 이미지 프레임 단위로 읽기
while True:
    ret, img = cap.read()

    if not ret:
        break
# 이미지 크기 지정된 비율 크기만큼 재조정
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
#원본 이미지 복사해 놓음.
    ori = img.copy()

    # detect faces 이미지에서 얼굴모양 찾기
    faces = detector(img)
    # 0번 인덱스, 첫번째 인식되는 얼굴 하나만 인식하기
    face = faces[0]


    # 얼굴 영역에서 특징점 찾기
    dlib_shape = predictor(img, face)
    # dlib 객체를 numpy 객체로 변환
    shape_2d = np.array([[p.x, p.y] for p in dlib_shape.parts()])
```

```
    #computer center of face
    #얼굴 영역의 좌상단 좌표 구하기
    top_left = np.min(shape_2d, axis = 0)
    #얼굴 영역의 우하단 좌표 구하기
    bottom_right = np.max(shape_2d, axis = 0)

    # 얼굴의 중심점 찾기
    # .astype(np.int) -> 중심점이 소수점으로 나올수 있기 때문에 정수형으로 변환
    center_x, center_y = np.mean(shape_2d, axis=0).astype(np.int)

    # visualize # 얼굴영영에 네모 모양 이미지 그리지
    # face.left(), face.top() 좌상단
    # face.right(), face.bottom() 우하단
    # color=(255, 255, 255) 흰색
    # thickness = 2 2pixel 찍기
    # lineType=cv2.LINE_AA   : 선타입(직선) 지정
    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )

    # 얼국 특징점에 대해 흰색으로 원모양 찍기 (68개)
    for s in shape_2d:
        cv2.circle(img, center=tuple(s), radius=1, color=(255, 255, 255), thickness=2, lineType=cv2.LINE_AA)

    #좌상단 좌표에 원 모양의 좌표 표현하기 (파란색)
    cv2.circle(img, center=tuple(top_left), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
    #우하단 좌표에 원 모양의 좌표 표현하기 (파란색)
    cv2.circle(img, center=tuple(bottom_right), radius=1,  color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)

    # 얼굴 중심점 표현하기 (빨강색)
    cv2.circle(img, center=tuple((center_x, center_y)), radius=1,  color=(0, 0, 255), thickness=2, lineType=cv2.LINE_AA)
# 이미지 출력
    cv2.imshow('img', img)
# 지연시간(1ms)을 두어서 출력이 용이하게
    cv2. waitKey(1)

# 이미지 크기가 scaler 크기만큼 조절되어서 출력


# In[1]:


import cv2, dlib, sys
import numpy as np

scaler = 0.1
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

cap = cv2.VideoCapture('Image.mp4')
#cap = cv2.VideoCapture(0)
```

```
#오버레이(얼굴에 씌울 이미지)할 이미지 지정
overlay = cv2.imread('ryan_transparent2.png', cv2.IMREAD_UNCHANGED)
# overlay = cv2.imread('AIEE.png', cv2.IMREAD_UNCHANGED)

# 오버레이 변환 함수 (구글링에서 찾음)
def overlay_transparent(background_img, img_to_overlay_t, x, y, overlay_size=None):
    try :
        bg_img = background_img.copy()
        # convert 3 channels to 4 channels
        if bg_img.shape[2] == 3:
            bg_img = cv2.cvtColor(bg_img, cv2.COLOR_BGR2BGRA)

        if overlay_size is not None:
            img_to_overlay_t = cv2.resize(img_to_overlay_t.copy(), overlay_size)

        b, g, r, a = cv2.split(img_to_overlay_t)

        mask = cv2.medianBlur(a, 5)

        h, w, _ = img_to_overlay_t.shape
        roi = bg_img[int(y-h/2):int(y+h/2), int(x-w/2):int(x+w/2)]

        img1_bg = cv2.bitwise_and(roi.copy(), roi.copy(), mask=cv2.bitwise_not(mask))
        img2_fg = cv2.bitwise_and(img_to_overlay_t, img_to_overlay_t, mask=mask)

        bg_img[int(y-h/2):int(y+h/2), int(x-w/2):int(x+w/2)] = cv2.add(img1_bg, img2_fg)

        # convert 4 channels to 4 channels
        bg_img = cv2.cvtColor(bg_img, cv2.COLOR_BGRA2BGR)
        return bg_img
    except Exception : return background_img

#비디오 파일 프레임 단위로 읽기 #
while True:
    ret, img = cap.read()
    if not ret:
        break
    img = cv2.resize(img, (int(img.shape[1]*scaler), int(img.shape[0]*scaler)))
    ori = img.copy()

    # detect faces
    faces = detector(img)
    # 0번 인덱스, 얼굴 하나만 인식하기
    face = faces[0]

    dlib_shape = predictor(img, face)
    shape_2d = np.array([[p.x, p.y] for p in dlib_shape.parts()])

    #computer center of face
    top_left = np.min(shape_2d, axis = 0)
```

```
    bottom_right = np.max(shape_2d, axis = 0)

    #얼굴 크기만큼 리사이즈
    # 1.8은 1.8배 키움
    # 정수형 변환을 위해 앞에 int를 지정
    face_size = int(max(bottom_right-top_left) * 1.8)

    center_x, center_y = np.mean(shape_2d, axis=0).astype(np.int)

    #오버레이할 이미지를 찾은 얼굴의 중심에 두고 오버레이할 크기만큼 지정해줌
    result = overlay_transparent(ori, overlay, center_x, center_y, overlay_size=(face_size, face_size) )

    # visualize #
    img = cv2. rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
                         thickness = 2, lineType=cv2.LINE_AA )

    for s in shape_2d:
        cv2.circle(img, center=tuple(s), radius=1, color=(255, 255, 255), thickness=2, lineType=cv2.LINE_AA)

    cv2.circle(img, center=tuple(top_left), radius=1,   color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
    cv2.circle(img, center=tuple(bottom_right), radius=1,   color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)

    cv2.circle(img, center=tuple((center_x, center_y)), radius=1,   color=(0, 0, 255), thickness=2, lineType=cv2.LINE_AA)

    cv2.imshow('img', img)
    cv2.imshow('result', result)
    cv2. waitKey(1)


# In[ ]:
```