# F. DLIB와 얼굴 랜드마크

1

## DLIB 라이브러리

**Dlib**

- http://dlib.net/
- 복잡한 소프트웨어를 생성하기 위한 기계 학습 알고리즘 및 도구가 포함된 C++로 짜여진 라이브러리
- 안면인식에 관련한 기능이 마련되어 있어서, 해당 분야에서 자주 쓰이는 라이브러리
- Dlib의 오픈 소스 라이선스를 사용하면 모든 애플리케이션에서 무료로 사용할 수 있음
- Dlib은 C++ 기반의 라이브러리라 pip를 이용하여 직접 설치가 안됨 => Cmake 필요


2

# DLIB 라이브러리

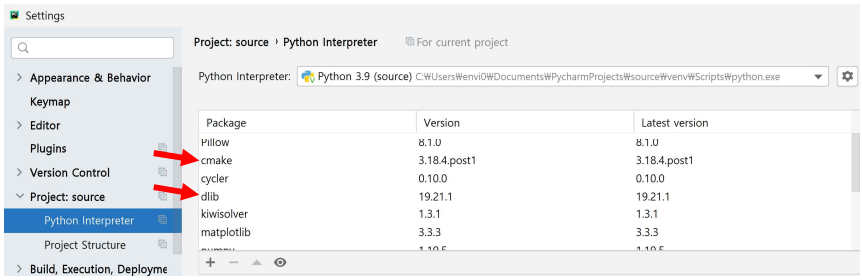
**Dlib 설치**

**in PyCharm**

- 메뉴 File > Settings > Project > Python Interpreter에서

① cmake 패키지 추가

② dlib 패키지 추가

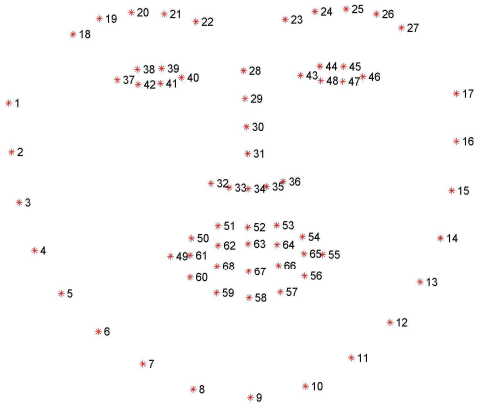



3

# 얼굴 랜드마크 검출

**68개의 랜드마크 위치 (x,y) 검출**

- DLIB 주요 API
  - `dlib.get_frontal_face_detector()` : 얼굴 검출 함수
    - 얼굴 좌표 배열 반환 `(left, top, right, bottom)`
  - `dlib.shape_predictor(파일)` : 랜드마크 검출 함수
    - 68개의 랜드마크 배열 반환 `(part.x, part.y)`



4

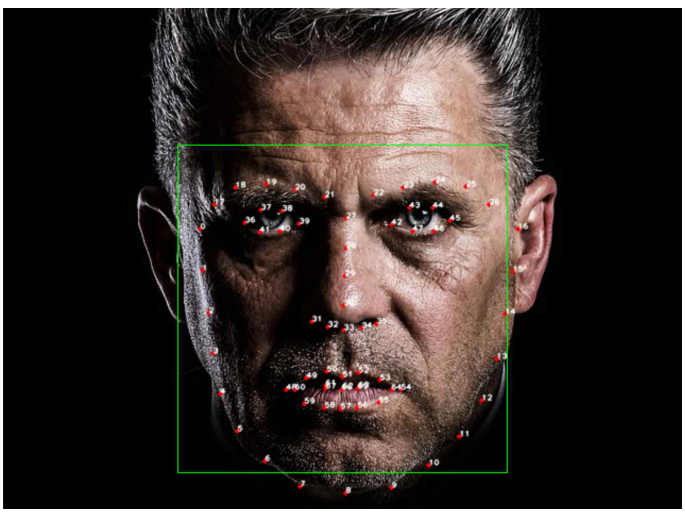
# 얼굴 랜드마크 검출

http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

```
import cv2
import dlib

detector = dlib.get_frontal_face_detector()          # 얼굴 검출기
predictor = dlib.shape_predictor('./shape_predictor_68_face_landmarks.dat') # 랜드마크 검출기

img = cv2.imread("./man_face.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = detector(gray)              # 얼굴 검출기로 얼굴 영역 검출

for rect in faces:
    x,y = rect.left(), rect.top()                              # 얼굴 영역(rect)을 좌표로 변환
    w,h = rect.right()-x, rect.bottom()-y
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 1) # 얼굴 영역 좌표로 사각형 표시
    shape = predictor(gray, rect) # 랜드마크 검출기로 얼굴 랜드마크 검출
    for i in range(68):           # 각 랜드마크 좌표 추출 및 표시
        part = shape.part(i)
        cv2.circle(img, (part.x, part.y), 2, (0, 0, 255), -1)
        cv2.putText(img, str(i), (part.x, part.y), \
             cv2.FONT_HERSHEY_PLAIN, 0.5,(255,255,255), 1, cv2.LINE_AA)
```

5

# 얼굴 랜드마크 검출

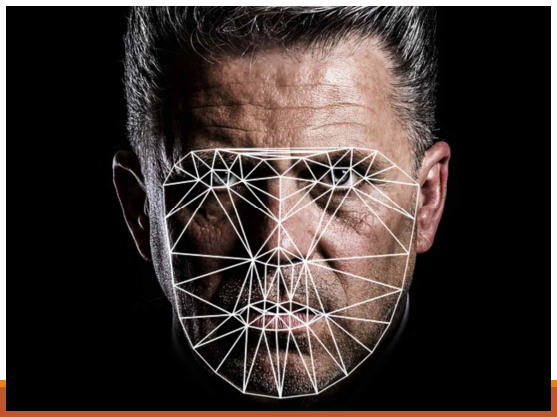


6

# 얼굴 랜드마크 응용

**들로네 삼각분할(Delaunay Triangulation)**

- 2차원 평면에 분포하는 점들을 대상으로 삼각형들을 만드는데 각각의 삼각형들은 본인들 세 점을 제외한 다른 점을 포함하지 않게끔 삼각형을 만들어 분할하는 것
- OpenCV 주요 API
  - cv2.Subdiv2D(x,y,w,h) : 주어진 범위 내에서 들로네 삼각형 분할 객체 생성



7

# 얼굴 랜드마크 응용

**들로네 삼각분할(Delaunay Triangulation)**

```
points = []
for i in range(68):
    part = shape.part(i)        # 부위별 좌표 추출
    points.append((part.x, part.y))

x, y, w, h = cv2.boundingRect(np.float32(points))
subdiv = cv2.Subdiv2D((x, y, x + w, y + h))  # 들로네 삼각 분할 객체 생성
subdiv.insert(points)                        # 랜드마크 좌표 추가
triangleList = subdiv.getTriangleList()      # 들로네 삼각형 좌표 계산

# 들로네 삼각형 그리기
h, w = img.shape[:2]
cnt = 0
for t in triangleList:
    pts = t.reshape(-1, 2).astype(np.int32)
    # 좌표 중에 이미지 영역을 벗어나는 것을 제외(음수 등)
    if (pts < 0).sum() or (pts[:, 0] > w).sum() or (pts[:, 1] > h).sum():
        print(pts)
        continue
    cv2.polylines(img, [pts], True, (255, 255, 255), 1, cv2.LINE_AA)
    cnt += 1
```

얼굴영상 내 검출한 삼각형의 정보 : x0, y0, x1,y1, x2, y2

```
[[227. 203. 218. 170. 242. 191.]
 [218. 170. 227. 203. 198. 186.]
 [266. 361. 204. 365. 197. 327.] ...]
```

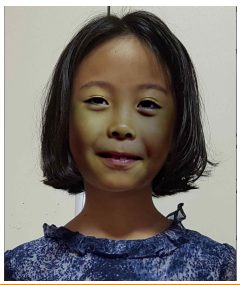
8

# 얼굴 랜드마크 응용

**얼굴 스와핑(face swaping)**

- 얼굴 들로네 삼각형을 이용하여 두사람의 얼굴을 뒤바꿈
  - A&B 얼굴 랜드마크 검출
  - A&B 얼굴 들로네 삼각형
  - A 얼굴 들로네 삼각형을 B 삼각형으로 와핑 변환
    - A 얼굴의 삼각형이 B 얼굴의 삼각형과 동일해 짐
  - A 얼굴의 와핑된 삼각형들을 B 삼각형에 합성
    - B의 랜드마크 영역만 A의 랜드마크 영역으로 바뀜





9

# 얼굴 랜드마크 응용

**얼굴 스와핑(face swaping)**

```
def warpTriangle(img1, img2, pts1, pts2):
    x1, y1, w1, h1 = cv2.boundingRect(np.float32([pts1]))
    x2, y2, w2, h2 = cv2.boundingRect(np.float32([pts2]))

    roi1 = img1[y1:y1 + h1, x1:x1 + w1]
    roi2 = img2[y2:y2 + h2, x2:x2 + w2]

    offset1 = np.zeros((3, 2), dtype=np.float32)
    offset2 = np.zeros((3, 2), dtype=np.float32)
    for i in range(3):
        offset1[i][0], offset1[i][1] = pts1[i][0] - x1, pts1[i][1] - y1
        offset2[i][0], offset2[i][1] = pts2[i][0] - x2, pts2[i][1] - y2

    mtrx = cv2.getAffineTransform(offset1, offset2)
    warped = cv2.warpAffine(roi1, mtrx, (w2, h2), None, cv2.INTER_LINEAR, cv2.BORDER_REFLECT_101)

    mask = np.zeros((h2, w2), dtype=np.uint8)
    cv2.fillConvexPoly(mask, np.int32(offset2), (255))

    warped_masked = cv2.bitwise_and(warped, warped, mask=mask)
    roi2_masked = cv2.bitwise_and(roi2, roi2, mask=cv2.bitwise_not(mask))
    roi2_masked = roi2_masked + warped_masked
    img2[y2:y2 + h2, x2:x2 + w2] = roi2_masked
```

10