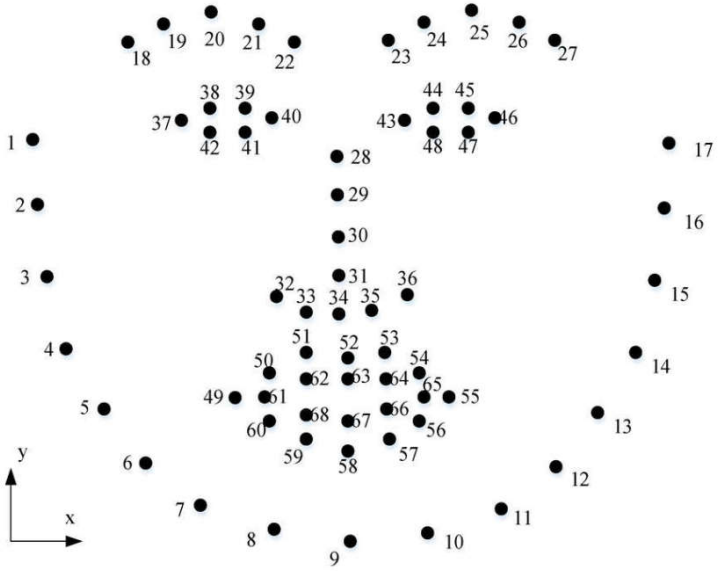
Python编程与人工智能实践

应用篇：OpenCV+Dlib的人脸转换（face swap）

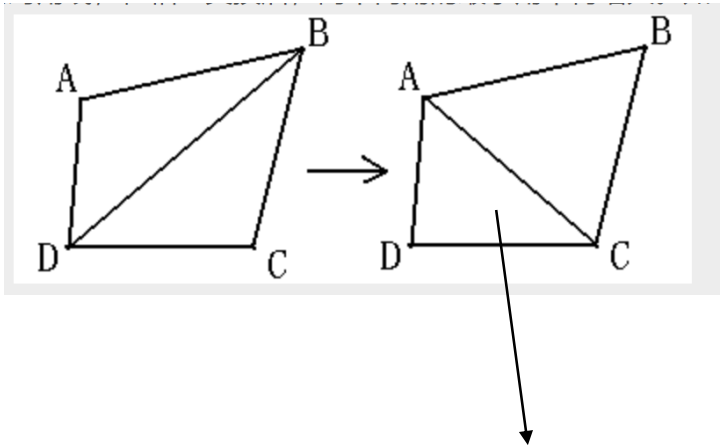
于泓
鲁东大学
信息与电气工程学院
2022.1.29

人脸转换 （face swap）

（1）利用 Dlib 获取68个脸部的关键点

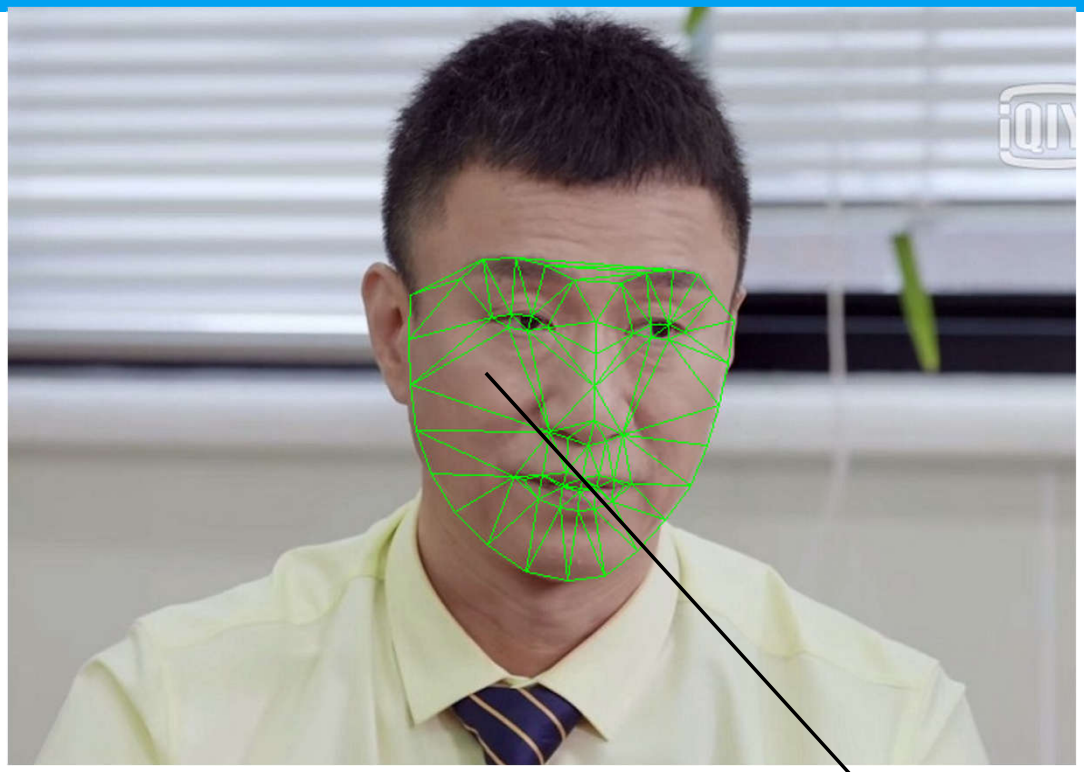


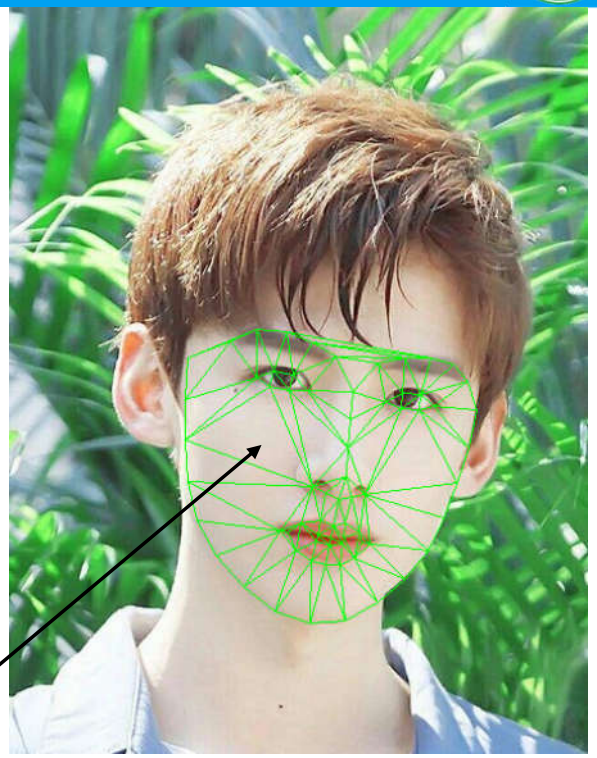
（2） 进行三角剖分（Delaunay）



令最小角最大

利用subdiv实现



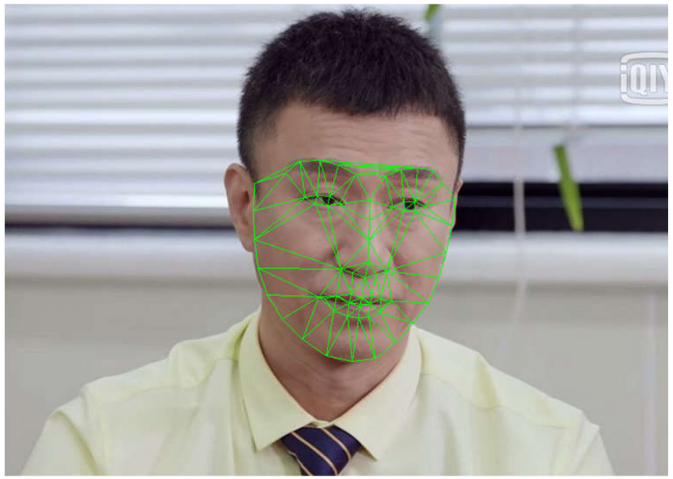


源

目标

(3) **源**三角到**目标**三角的转换

（4）贴图




(5) 色彩均衡

代码实现：

（1）人脸检测，获取68个关键点的坐标

```python
import cv2
import dlib
import numpy as np

# 获取图像中的人脸关键点
# 输入
# img :  图像
# det_face :  人脸检测器
# det_landmarks :  人脸关键点检测器
def get_landmarks_points(img,det_face,det_landmarks):
    # 转换为灰度
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # 检测人脸区域
    face_rects = det_face(gray, 0)

    # 获取68个关键点
    landmarks = det_landmarks(gray, face_rects[0])

    # 获取关键点的坐标
    landmarks_points = []
    parts = landmarks.parts()
    for part in parts:
        landmarks_points.append((part.x,part.y))
    return landmarks_points
```

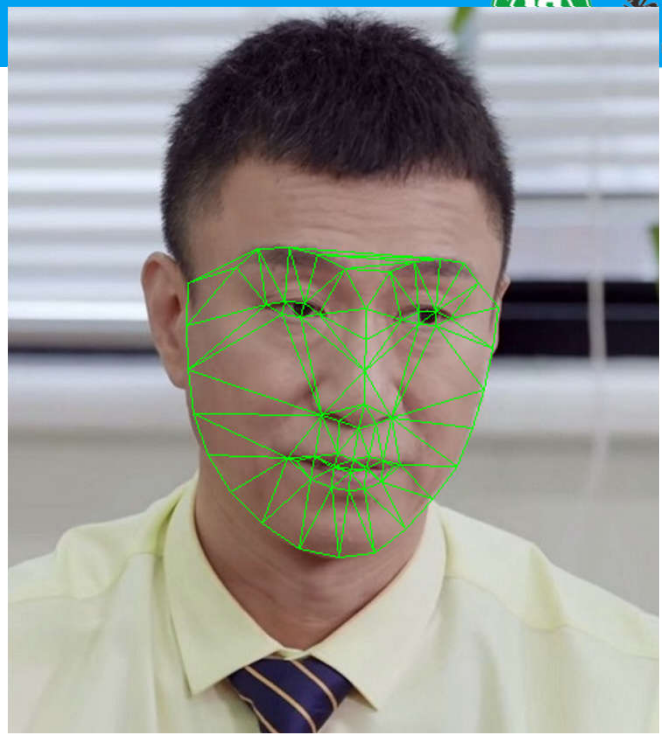
人脸检测器

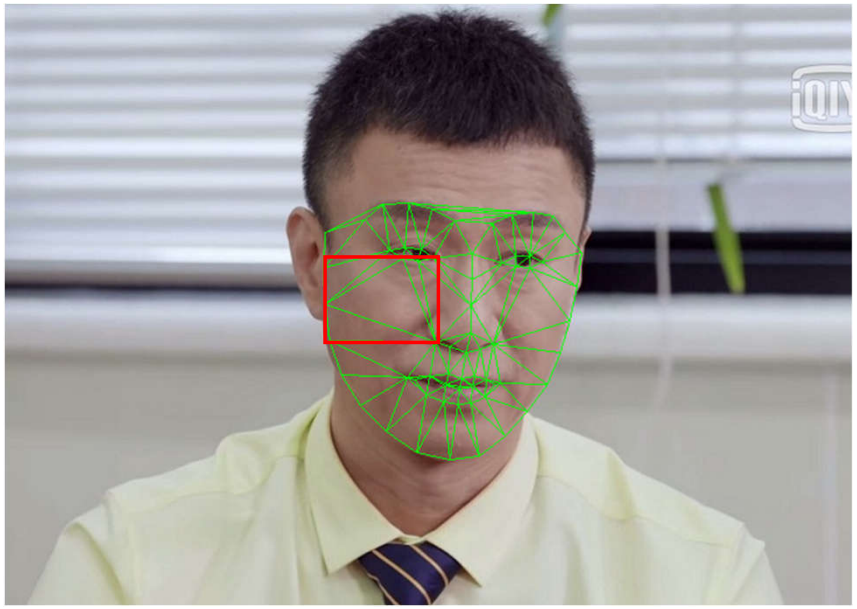
关键点检测器

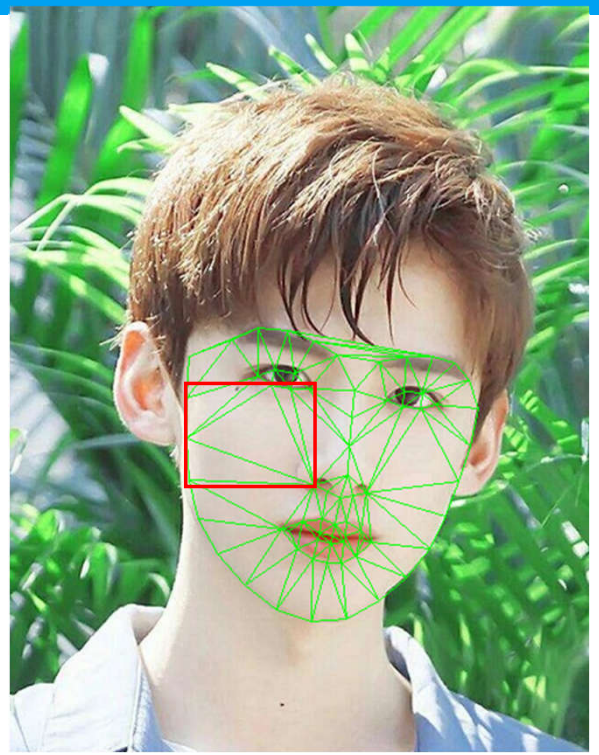
```python
# 获取内解三角形的顶点list
# landmarks_points :  68个关键点的坐标
def get_tri_pt_index_list(landmarks_points):

    points = np.array(landmarks_points,np.int32)
    # print(points)
    # 得到人脸区域的凸包
    convexhull = cv2.convexHull(points)

    # 获取凸包的外截矩形
    rect = cv2.boundingRect(convexhull)

    # 定义subdiv 用来进行三角剖分
    subdiv = cv2.Subdiv2D(rect)
    subdiv.insert(landmarks_points)
    triangles = subdiv.getTriangleList()

    triangles = np.array(triangles,dtype = np.int32)

    # 获取每个三角形的坐标位置
    list_index_tris = []
    for t in triangles:
        pt1 = (t[0],t[1])
        pt2 = (t[2],t[3])
        pt3 = (t[4],t[5])

        index_pt1 = np.where((points == pt1).all(axis=1))[0]
        index_pt2 = np.where((points == pt2).all(axis=1))[0]
        index_pt3 = np.where((points == pt3).all(axis=1))[0]

        if index_pt1.size !=0 and index_pt2.size !=0 and index_pt3.size !=0:
            list_index_tris.append((index_pt1[0],index_pt2[0],index_pt3[0]))
    return list_index_tris
```

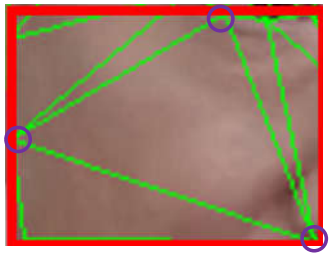
三角剖分

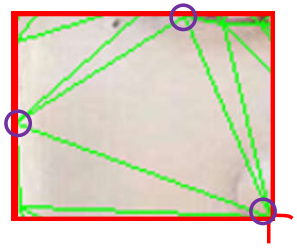
获取三个顶点的index











(1) 利用三个顶点，做外接矩形
实现矩形->矩形的变换
(2) 只保留三角区域

```python
def get_one_rect_from_tri(img,tri,landmarks):

    # 获取三个顶点的坐标
    pt1 = landmarks[tri[0]]
    pt2 = landmarks[tri[1]]
    pt3 = landmarks[tri[2]]
    points = np.array([pt1,pt2,pt3],dtype = np.int32)

    # 做一个外接矩形
    crop_rect = cv2.boundingRect(points)
    (x,y,w,h) = crop_rect

    # 计算三个顶点在外接矩形上的坐标
    points_in_rect = points -np.array([(x,y)])

    # 进行图像截取
    crop_img = img[y:y+h,x:x+w]

    return crop_img,crop_rect,points_in_rect
```

根据三个顶点做外接矩形
并计算三个顶点在新的外接矩形上的坐标

```python
def get_face_cover(img_src,img_dst,landmarks_src,landmarks_dst,list_index_tris):

    img_cover = np.zeros_like(img_dst,np.uint8)

    for tri in list_index_tris:

        # 源图片的一个三角上截取一个矩形
        crop_img_src,crop_rect_src,points_in_rect_src = get_one_rect_from_tri(img_src,tri,landmarks_src)

        # 从目标图片的相同位置的三角上也截取一个矩形
        crop_img_dst,crop_rect_dst,points_in_rect_dst = get_one_rect_from_tri(img_dst,tri,landmarks_dst)

        # 计算变换矩阵
        pts_src = np.float32(points_in_rect_src)
        pts_dst = np.float32(points_in_rect_dst)
        M = cv2.getAffineTransform(pts_src,pts_dst)

        # 实现源 到目标的转换
        (x,y,w,h) = crop_rect_dst
        warped_src = cv2.warpAffine(crop_img_src, M, (w, h))
```

```
    # 由于只替换目标图像中三角区域内的部分
    # 因此需要做一个mask,对三角区域内填充255，其他部分填充0
    mask_dst = np.zeros((h,w),np.uint8)
    cv2.fillConvexPoly(mask_dst,points_in_rect_dst,255)
    warped_tri = cv2.bitwise_and(warped_src,warped_src,mask=mask_dst)

    # 直接叠加三角连接处会重复相加，因此在叠加时，只对区域内非零的部分进行叠加
    # img_mask[y:y+h,x:x+w] = img_mask[y:y+h,x:x+w] + warped_tri
    img_area = img_cover[y:y+h,x:x+w]
    img_area_gray = cv2.cvtColor(img_area,cv2.COLOR_BGR2GRAY)
    _,mask_area = cv2.threshold(img_area_gray,1,255,cv2.THRESH_BINARY_INV)
    warped_tri = cv2.bitwise_and(warped_tri,warped_tri,mask=mask_area)
    img_area = cv2.add(img_area,warped_tri)
    img_cover[y:y+h,x:x+w] = img_area

return img_cover
```

只保留三角部分

对接缝的部分进行处理

人脸融合

```
def face_swap(img_dst,img_cover,landmarks_dst):

    # 获取人脸部分的凸包
    img_dst_gray = cv2.cvtColor(img_dst,cv2.COLOR_BGR2GRAY)
    points = np.array(landmarks_dst,np.int32)
    convexhull = cv2.convexHull(points)

    # 凸包填充，得到掩模，获取非人脸部分
    face_mask = np.zeros_like(img_dst_gray)
    face_mask_255 = cv2.fillConvexPoly(face_mask, convexhull, 255)
    face_mask_0 = cv2.bitwise_not(face_mask_255)
    img_noface = cv2.bitwise_and(img_dst, img_dst, mask=face_mask_0)

    # 将非人力脸 和人脸部分 叠加
    result = cv2.add(img_noface, img_cover)
    cv2.imshow("Image_result",result)

    # 颜色调整
    (x, y, w, h) = cv2.boundingRect(convexhull)
    center_face = (int((x + x + w) / 2), int((y + y + h) / 2))

    seamlessclone = cv2.seamlessClone(result, img_dst, face_mask_255, center_face, cv2.NORMAL_CLONE)

    return seamlessclone
```

目标图像中去除人脸部分

替换成从源图像转换来的人脸

色彩调整

主函数:

```
if __name__ == "__main__":
    # 创建人脸检测器
    det_face = dlib.get_frontal_face_detector()

    # 加载标志点检测器
    det_landmarks = dlib.shape_predictor("shape_predictor_68_face_landmarks_GTX.dat")  # 68点

    # 打开图片
    img_src = cv2.imread('sunhonglei.jpg')
    img_dst = cv2.imread('baijingting.jpg')

    # 获取源图像的68个关键点的坐标
    landmarks_src = get_landmarks_points(img_src,det_face,det_landmarks)

    # 获取目标图像的68个关键点的坐标
    landmarks_dst = get_landmarks_points(img_dst,det_face,det_landmarks)

    # 获取用来进行三角剖分的关键点的index—list
    list_index_tris = get_tri_pt_index_list(landmarks_src)

    # 获取目标图像中 需要替换的部分
    img_cover = get_face_cover(img_src,img_dst,landmarks_src,landmarks_dst,list_index_tris)

    # 进行人脸替换
    result = face_swap(img_dst,img_cover,landmarks_dst)

    cv2.imshow("img src",img_src)
    cv2.imshow("img dst",img_dst)
    cv2.imshow("img cover",img_cover)
    cv2.imshow("result",result)

    cv2.waitKey(0)
```