Python编程与人工智能实践

应用篇：基于Dlib+局部缩放的大眼（小眼）特效（Thin Face）

于泓
鲁东大学
信息与电气工程学院
2022.1.31

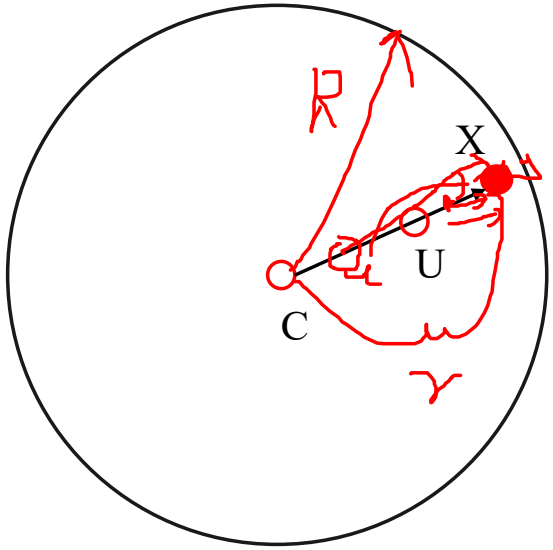
大眼小眼特效的基本原理

local Scale Wapping　局部缩放技术

(1) 只对一个圆形区域内（半径为R）的像素点进行修改（**缩放**）

(2) 区域内所有点沿半径方向移动

(3) 距离圆心越近，移动的距离越大
距离圆心越远，移动的距离越小



若平移后，一个点位于x处，那么平移前这个点位于U?

移动方向相同：$$\frac{\vec{C}-\vec{U}}{\vec{C}-\vec{X}}=\lambda$$

R表示半径
在圆边缘处的点不平移
r 表示 C->X的距离

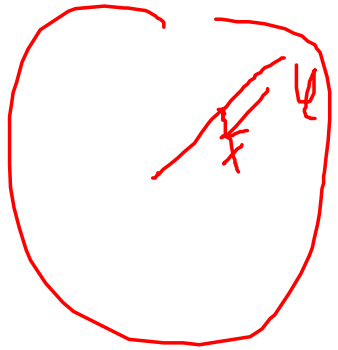
$$\lambda=\left(1-\left(\frac{r}{R}-1\right)^{2}\alpha\right)$$

$$\frac{\vec{C}-\vec{U}}{\vec{C}-\vec{X}}=\lambda$$

$$\vec{U}=\vec{C}+\lambda(\vec{X}-\vec{C})$$

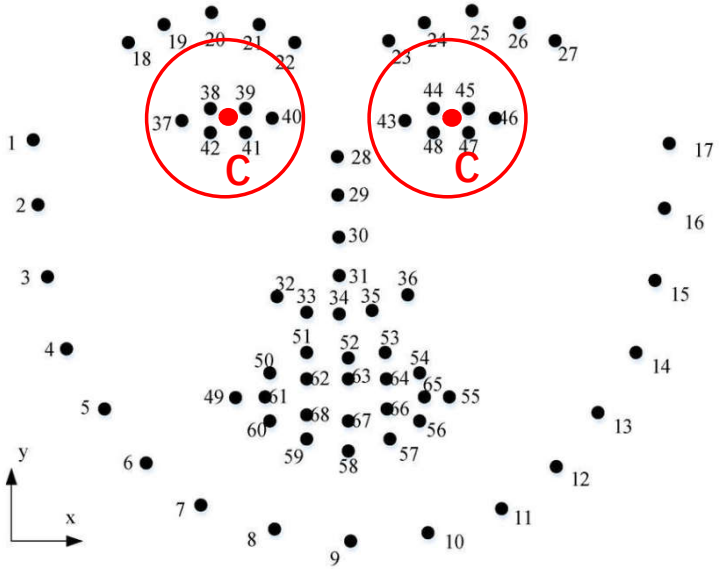
$$\lambda=\left(1-\left(\frac{r}{R}-1\right)^2\alpha\right)$$

$$=\vec{C}+\left(1-\left(\frac{r}{R}-1\right)^2\alpha\right)(\vec{X}-\vec{C})$$



$\alpha>0$ 局部放大 越大，放大的越厉害

$\alpha<0$ 局部缩小 绝对值越大，缩小的越厉害



大眼任务：

右眼

根据38,39,4142，决定圆心C

根据C到37,40的距离决定半径（加适当的放大）

左眼

根据44,45,48,47，决定圆心C

根据C到43,46的距离决定半径（加适当的放大）

效果:

a=1

a=-1

代码实现：

获取68个关键点

```
# 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
```

```
def localScaleWap(img,pt_C,R,scaleRatio):

    h,w,c = img.shape
    # 文件拷贝
    copy_img = np.zeros_like(img)
    copy_img = img.copy()

    # 创建蒙板
    mask = np.zeros((h,w),dtype = np.uint8)
    cv2.circle(mask,pt_C,np.int32(R),255,cv2.FILLED)

    pt_C = np.float32(pt_C)
```

$$\vec{U}=\vec{C}+\lambda(\vec{X}-\vec{C})$$

$$=\vec{C}+\left(1-\left(\frac{r}{R}-1\right)^{2}\alpha\right)(\vec{X}-\vec{C})$$

```
for i in range(w):
    for j in range(h):

        # 只计算半径内的像素
        if mask[j,i] ==0:
            continue

        pt_X = np.array([i,j],dtype = np.float32)


        dis_C_X = np.sqrt(np.dot((pt_X-pt_C),(pt_X-pt_C)))

        alpha = 1.0 - scaleRatio * pow(dis_C_X / R - 1.0, 2.0)

        pt_U = pt_C + alpha*(pt_X-pt_C)

        # 利用双线性差值法，计算U点处的像素值
        value = BilinearInsert(img,pt_U)
        copy_img[j,i] = value

return copy_img
```

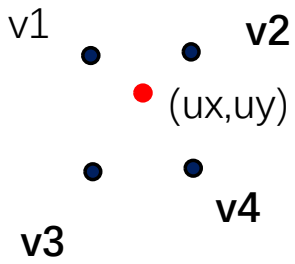
```python
# 双线性差值
def BilinearInsert(src,pt_U):
    ux = pt_U[0]
    uy = pt_U[1]

    x1=np.float32(int(ux))
    x2=x1+1
    y1=np.float32(int(uy))
    y2=y1+1

    v1 = np.float32(src[int(y1),int(x1)])
    v2 = np.float32(src[int(y1),int(x2)])
    v3 = np.float32(src[int(y2),int(x1)])
    v4 = np.float32(src[int(y2),int(x2)])

    part1 = v1 * (x2 - ux) * (y2 - uy)
    part2 = v2 * (ux - x1) * (y2 - uy)
    part3 = v3 * (x2 - ux) * (uy - y1)
    part4 = v4 * (ux - x1) * (uy - y1)

    insertValue=part1+part2+part3+part4
    return insertValue.astype(np.uint8)
```



```python
# 滑块的响应函数
def empty(a):
    pass


if __name__ == "__main__":

    # 创建滑块
    cv2.namedWindow("TrackBars")
    cv2.resizeWindow("TrackBars",640,30)
    cv2.createTrackbar("big eye","TrackBars",100,200,empty)

    # 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("../faceswap/shape_predictor_68_face_landmarks_GTX.dat")  # 68点

    # 打开图片
    img = cv2.imread('sunhonglei.jpg')

    # 获取源图像的68个关键点的坐标
    landmarks = get_landmarks_points(img,det_face,det_landmarks)
    landmarks = np.array(landmarks)

    # 大眼调节参数
    scaleRatio =1

    # 小眼调节参数
    # scaleRatio =-1
```

```
# 右眼
index = [37,38,40,41]
pts_right_eyes = landmarks[index]
crop_rect = cv2.boundingRect(pts_right_eyes)
(x,y,w,h) = crop_rect
pt_C_right = np.array([x+w/2,y+h/2],dtype = np.int32)

r1 = np.sqrt(np.dot(pt_C_right-landmarks[36],pt_C_right-landmarks[36]))
r2 = np.sqrt(np.dot(pt_C_right-landmarks[39],pt_C_right-landmarks[39]))
R_right  =  1.5*np.max([r1,r2])

# 左眼
index = [43,44,45,47]
pts_left_eyes = landmarks[index]
crop rect = cv2.boundingRect(pts left eyes)
(x,y,w,h) = crop_rect
pt_C_left = np.array([x+w/2,y+h/2],dtype = np.int32)
r1 = np.sqrt(np.dot(pt_C_left-landmarks[42],pt_C_left-landmarks[42]))
r2 = np.sqrt(np.dot(pt_C_left-landmarks[46],pt_C_left-landmarks[46]))
R_left  =  1.5*np.max([r1,r2])

# 大右眼
img_bigeye = localScaleWap(img,pt_C_right,R_right,scaleRatio)

# 大左眼
img_bigeye = localScaleWap(img_bigeye,pt_C_left,R_left,scaleRatio)

# 显示
cv2.imshow('input',img)
cv2.imshow('output',img_bigeye)
```

```
while True:
    scaleRatio_new = cv2.getTrackbarPos("big eye","TrackBars")
    # 负数 小眼特效
    # scaleRatio_new = -scaleRatio_new/100
    # 正数 大眼特效
    scaleRatio_new = scaleRatio_new/100

    if scaleRatio != scaleRatio_new:
        scaleRatio = scaleRatio_new
        print("processing  scaleRatio= %.2f"%(scaleRatio))

        # 大左眼
        img_bigeye = localScaleWap(img,pt_C_right,R_right,scaleRatio)

        # 大右眼
        img_bigeye = localScaleWap(img_bigeye,pt_C_left,R_left,scaleRatio)
        cv2.imshow('output',img_bigeye)
        print("done")


    key=cv2.waitKey(1000) & 0xFF

    if key == ord('q'):
        break
```