Python编程与人工智能实践

应用篇：基于Dlib+局部平移的瘦脸特效（Thin Face）

于泓
鲁东大学
信息与电气工程学院
2022.1.31

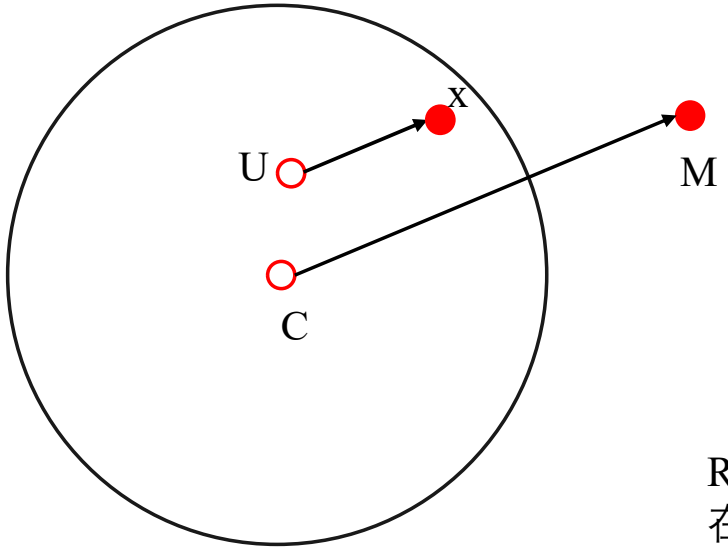
瘦脸特效的基本原理

Interactive Image Warping　局部平移技术

(1) 只对一个圆形区域内的像素点进行修改（**平移**）

(2) 区域内所有点移动方向相同

(3) 距离圆心越近，平移的距离越大
距离圆心越远，平移的距离越小



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：$\frac{\vec{U}-\vec{x}}{\vec{C}-\vec{M}}=\lambda \qquad \lambda<1$

R表示半径
在圆边缘处
的点不平移

$$\lambda=\left(\frac{R^2-\left|\vec{x}-\vec{C}\right|^2}{(R^2-\left|\vec{x}-\vec{C}\right|^2)+\alpha\left|\vec{M}-\vec{C}\right|^2}\right)^2$$

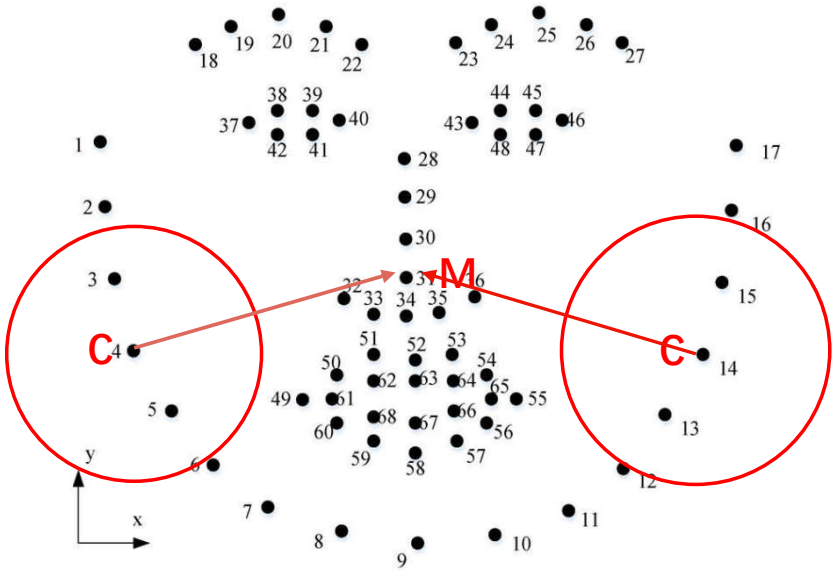
$$\frac{\vec{U}-\vec{x}}{\vec{C}-\vec{M}}=\lambda$$

x处的像素值，由U处的像素值进行替换

$$\lambda=\left(\frac{R^2-\left|\vec{x}-\vec{C}\right|^2}{(R^2-\left|\vec{x}-\vec{C}\right|^2)+\alpha\left|\vec{M}-\vec{C}\right|^2}\right)^2$$

$$\vec{U}=\vec{x}-\left(\frac{R^2-\left|\vec{x}-\vec{C}\right|^2}{(R^2-\left|\vec{x}-\vec{C}\right|^2)+\alpha\left|\vec{M}-\vec{C}\right|^2}\right)^2(\vec{M}-\vec{C})$$

需要确定的参数：
C，主要变化的位置（圆心）
M，主要的变化的方向
R，变化的区域
a，变化的程度



瘦脸任务：

右脸

点4 为圆心，4-6的距离为R，点31（鼻子处）为M

左脸

点14 为圆心，14-12的距离为R，点31（鼻子处）为M

效果:

a=0.6

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
```

```python
def localTranslationWap(img,pt_C,pt_M,r,a):

    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)

    # 计算 CM 之间的距离
    pt_C = np.float32(pt_C)
    pt_M = np.float32(pt_M)
    dis_M_C = np.dot((pt_C-pt_M),(pt_C-pt_M))
```

$$\vec{U}=\vec{x}-\left(\frac{R^2-\left|\vec{x}-\vec{C}\right|^2}{(R^2-\left|\vec{x}-\vec{C}\right|^2)+\alpha\left|\vec{M}-\vec{C}\right|^2}\right)^2(\vec{M}-\vec{C})$$

```python
    # 只对蒙板内大于0的数进行处理
    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

            # 计算 XC之间的距离
            pt_X = np.array([i,j],dtype = np.float32)
            dis_X_C = np.dot((pt_X-pt_C),(pt_X-pt_C))

            # 计算缩放比例
            radio = (r*r-dis_X_C)/(r*r-dis_X_C+a*dis_M_C)
            radio = radio*radio

            # 计算 目标图像（i，j）处由源图像U点替换
            pt_U = pt_X-radio*(pt_M-pt_C)

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

            # 像素替换
            copy_img[j,i] = value

    return copy_img
```

小数

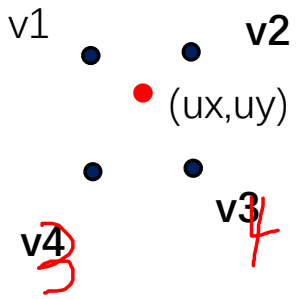
```python
# 双线性差值
def BilinearInsert(src,pt_U):
    ux = pt_U[0]
    uy = pt_U[1]

    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
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("../faceswap/shape_predictor_68_face_landmarks_GTX.dat")

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

    # 瘦脸程度调节
    a = 0.6

    # 右脸缩放
    pt_C_right = landmarks[3]
    pt_M = landmarks[30]
    r_right = np.sqrt(np.dot(landmarks[3]-landmarks[5],landmarks[3]-landmarks[5]))
    img_thin = localTranslationWap(img,pt_C_right,pt_M,r_right,a)

    # 左脸缩放
    pt_C_left = landmarks[13]
    pt_M = landmarks[30]
    r_left = np.sqrt(np.dot(landmarks[13]-landmarks[11],landmarks[13]-landmarks[11]))
    img_thin = localTranslationWap(img_thin,pt_C_left,pt_M,r_left,a)

    # 结果显示
    cv2.imshow('input',img)
    cv2.imshow('output',img_thin)
```