数据可视化 hw5

欧阳尔立 14307130293
周之烁 14300180079
杨雨樵 14300180085

一 问题和数据

我们的程序实现将人脸转换为一个与狒狒相似的脸。输入数据包括一个狒狒的图片（ape.jpg）和需要变换的人脸（example1.jpg 或者 example2.jpg）。

二 算法结构

2.1 确定局部仿射变换的对应区域（点）

本组的局部仿射变换区域为 n 个点，n 为特征点的个数。利用手动定位确定猩猩脸特征点集合 $U=\{U_1,U_2,\ldots,U_n\}$，利用旷视 Face++ 的 API 定位人脸特征点集合 $V=\{V_1,V_2,\ldots,V_n\}$。

之所以选择点作为局部区域，原因有二：一是更容易计算仿射变换矩阵（只需要用一个平移向量来表达），二是更容易计算每一个像素点 X 到局部区域 U_i 的距离 $d_i(X)$。

2.2 求出对应局部区域（点）的仿射变换函数

对于每一对对应区域 (U_i,V_i)，都可以求出其仿射变换矩阵。由于本组运用的算法中，局部仿射变换的区域为 n 个点，因此只需求出每一对对应点 (U_i,V_i) 的平移向量 b_i 即可。此时，第 i 对对应点的仿射变换函数为 $V_i=G_i(U_i)=U_i+b_i$。

2.3 进行局部仿射变换

对于狒狒图像的每一个像素点 X，利用局部仿射变换算法，计算该像素点对应到人脸的坐标 $T(X)$。局部仿射变换算法为：

$$T(X)\begin{cases} G_i(X), & X\in U_i, i=1,2,...,n \\ \sum_{i=1}^{n} w_i(X)G_i(X), & X\notin \bigcup_{i=1}^{n} U_i \end{cases}$$

其中，$G_i(X)=X+b_i$，$w_i(X)=\dfrac{1/d_i(X)^e}{\sum_{i=1}^{n}1/d_i(X)^e}$。

对于 e 的选取，这里参考庄吓海老师的 paper[1]，取 $e=2$ 。

2.4 插值求 RGB 值

利用双线性插值法，求出人脸坐标 $T(X)$ 的 RGB 值。假设 $T(X)$ 的坐标为 $(i+u, j+v)$，其中 i 、 j 为整数部分，u 、 v 为小数部分，则点 $T(X)$ 的 RGB 值由坐标为 (i, j) 、 $(i+1, j)$ 、 $(i, j+1)$ 、 $(i+1, j+1)$ 的周围四个点的 RGB 值决定。用 $f(i, j)$ 表示 (i, j) 处的 RGB 值，以此类推，则双线性插值算法为：

$$f(i+u, j+v)=(1-u)(1-v)\cdot f(i, j)+(1-u)v\cdot f(i, j+1)+u(1-v)\cdot f(i+1, j)+uv\cdot f(i+1, j+1)$$

2.5 填色

将狒狒图像所有像素点的 RGB 值进行替换，即为形变图像。

三 代码结构

3.1 仿射变换部分

（1）在 change 函数中读入人脸、狒狒脸的数据，并新建狒狒脸的图像，该图像用于重新设置像素点的颜色

（2）随后生成猩猩脸特征点字典 U（在 get_orangutan_message 函数中实现），人脸特征点字典 V（即进行仿射的点的字典 U 及它们的对应点的字典 V，在 get_face_message 函数中实现）。

（3）对于狒狒图像的每一个像素点，计算该像素点对应到人脸的坐标（locally_affine 函数实现），利用双线性插值（在 interpolation 中实现）得出该人脸坐标的 RGB 值，然后将狒狒图像像素点的 RGB 值进行替换。

3.2 GUI 部分

（1）定义 callback 函数，描述将要显示的图片的大小位置，并对输入的图片调用 change 函数进行变换。

（2）然后对 GUI 的图片布局进行定义

（3）定义一个 Button 类，使用参数 command 为 callback，使之能够完成相应的功能。

[1] Zhuang, X., Rhode, K. S., Razavi, R. S., Hawkes, D. J., & Ourselin, S. (2010). A registration-based propagation framework for automatic whole heart segmentation of cardiac mri. *IEEE Transactions on Medical Imaging, 29*(9), 1612.

四 开发环境

所有 GUI,算法及接口 API 均由 Python 完成。GUI 使用 Python Tkinter package 完成，API 使用 Face++人脸关键点接口。

五 实现过程

5.1 openCV 安装

我们用 pip install opencv 或 conda install opencv 安装时，都会提示我们没有匹配的 64 位 openCV 库。解决方法是用如下命令安装 openCV 库：

```
conda install -c https://conda.binstar.org/menpo opencv
```

5.2 特征点（局部仿射点）的选取

5.2.1 11 个特征点（眼睛 2x2 + 鼻子 2 + 嘴巴 5）

一开始我们想按照老师提供的思路进行选取：

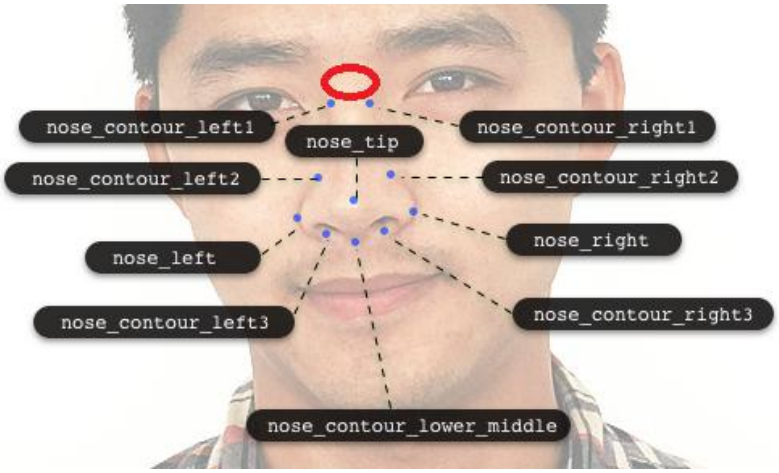
①左右眼各 2 个特征点：两个眼角(2x2=4)

②鼻子 3 个特征点：左右鼻翼及鼻梁(3)

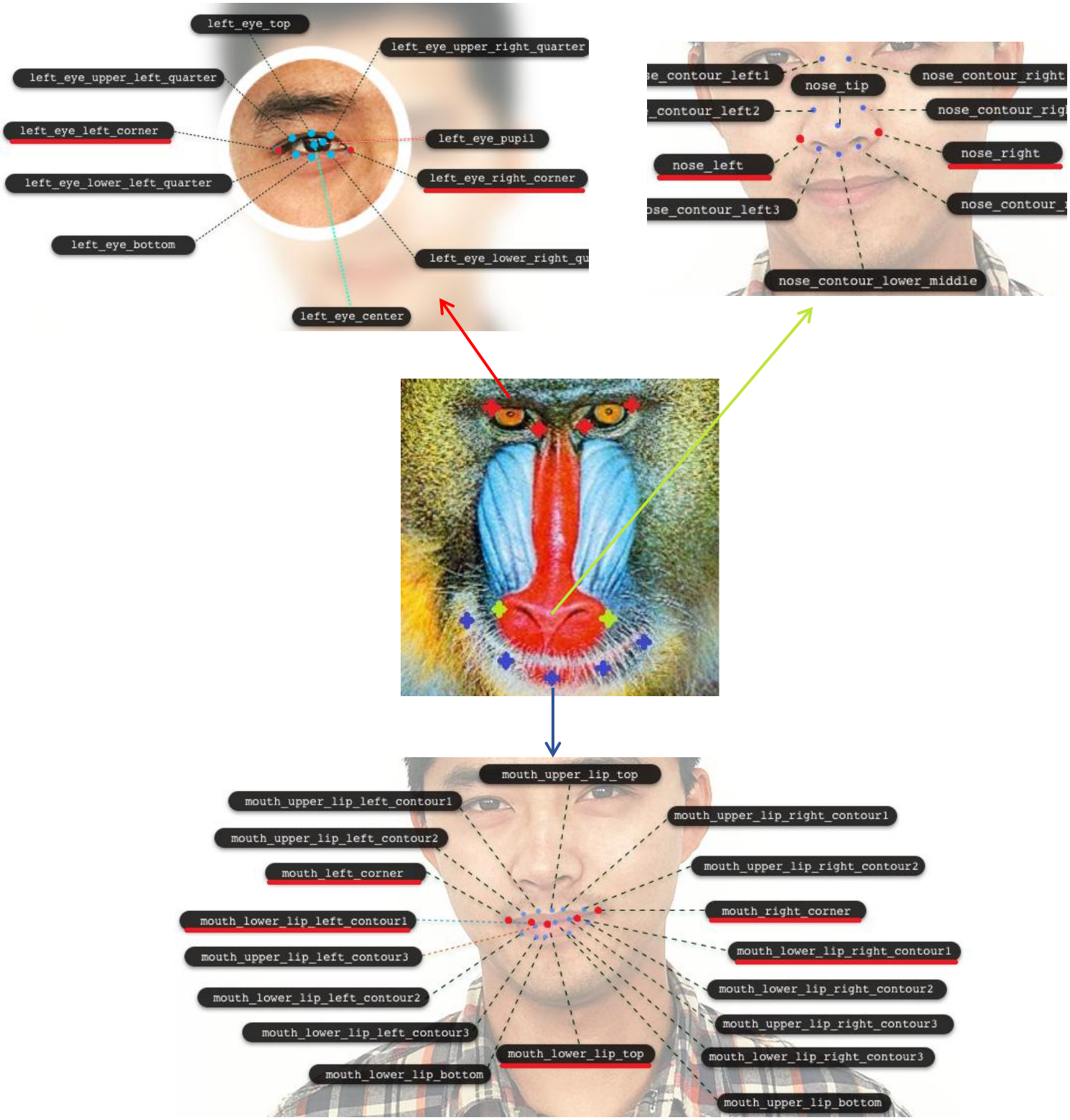
③嘴巴 5 个特征点：将嘴巴从左嘴角至右嘴角四等分，取 5 个点(5)

[共 12 个特征点]

但在 Face++识别出来的人脸特征点中，并不包括鼻梁。而且鼻梁与内眼角的距离很近，不考虑鼻梁对结果的影响很小。



因此我们选取除了鼻梁外的 11 个特征点进行局部仿射变换（已标红）：



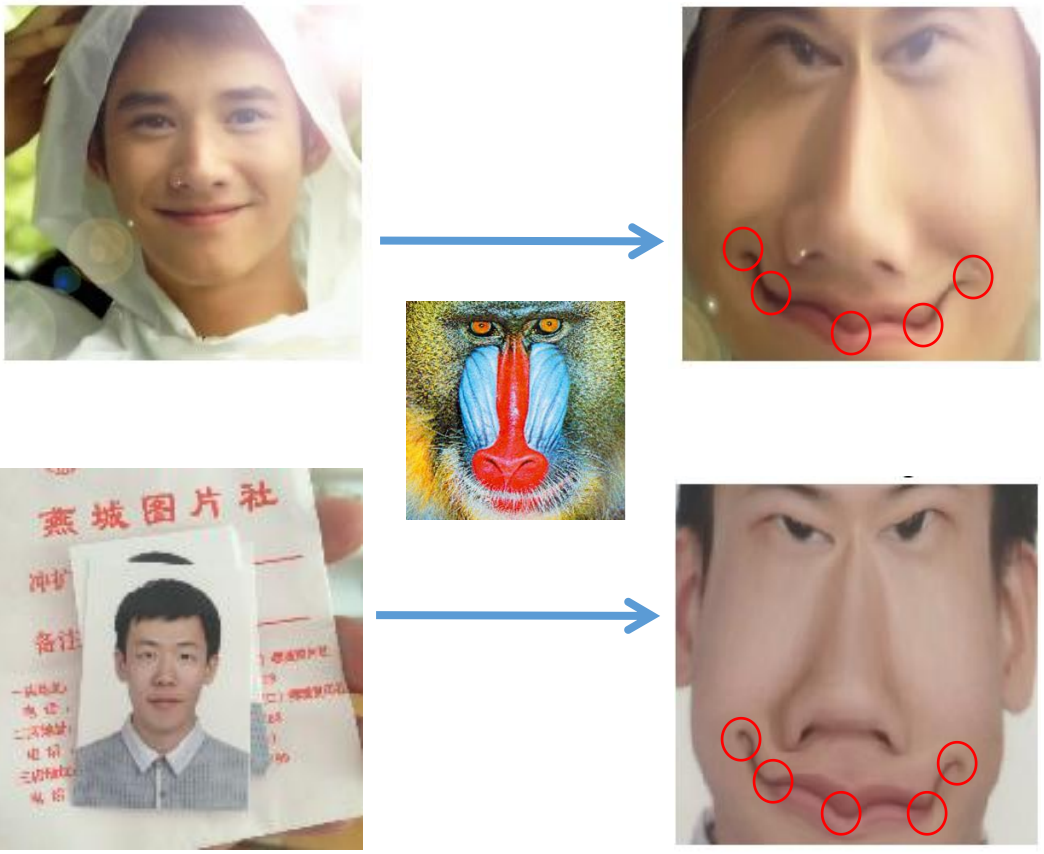
①左右眼各 2 个特征点：两个眼角(2x2=4)

②鼻子 2 个特征点：左右鼻翼(2)

③嘴巴 5 个特征点：将嘴巴从左嘴角至右嘴角四等分，取 5 个点(5)

[共 11 个特征点]

结果如下：



效果并不是很好，最大的问题出在嘴唇上：几乎所有的人脸经过处理后，嘴唇都变成了波浪形。根据形状来看，发生凹陷的几处地方均为我们选取的特征点位置。

我们推测：

1、由于嘴唇较长，5 个特征点仍然不足够用来表示嘴唇信息。

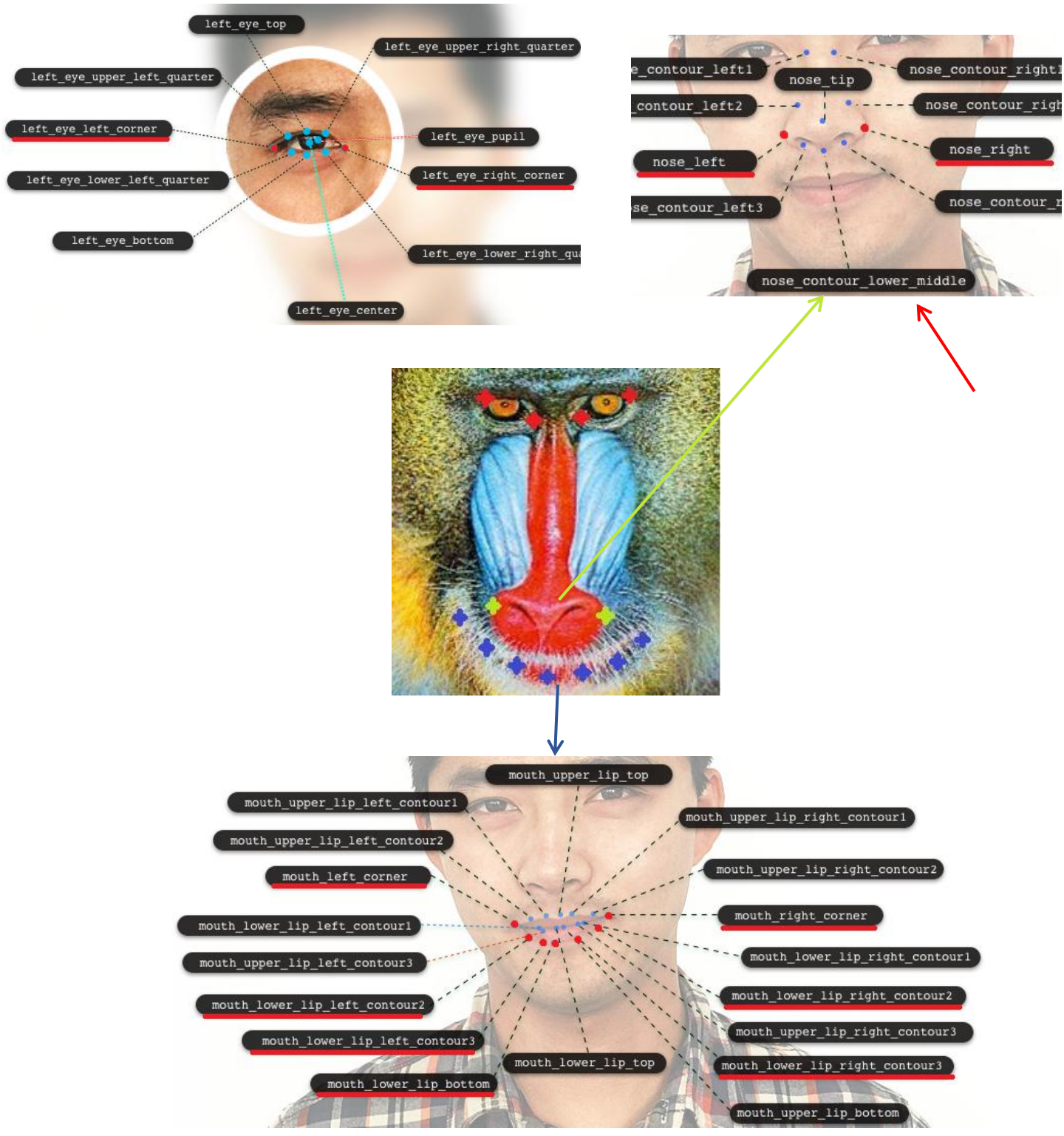
2、选取上下嘴唇交界线上的点作为局部仿射点时，嘴唇上的其他点受鼻子特征点的影响很大，这也是非特征点区域上凸的原因。

因此，我们接下来对嘴唇的特征点选取进行了改进。

5.2.2 13 个特征点（眼睛 2x2 + 鼻子 2 + 嘴巴 7）

首先，我们将嘴唇的特征点数量增至 7 个，即用六等分替代原来的四等分。其次，我们选取了离鼻子较远的嘴唇下轮廓进行特征点定位。

这样一来，我们选取了 13 个特征点进行局部仿射变换：



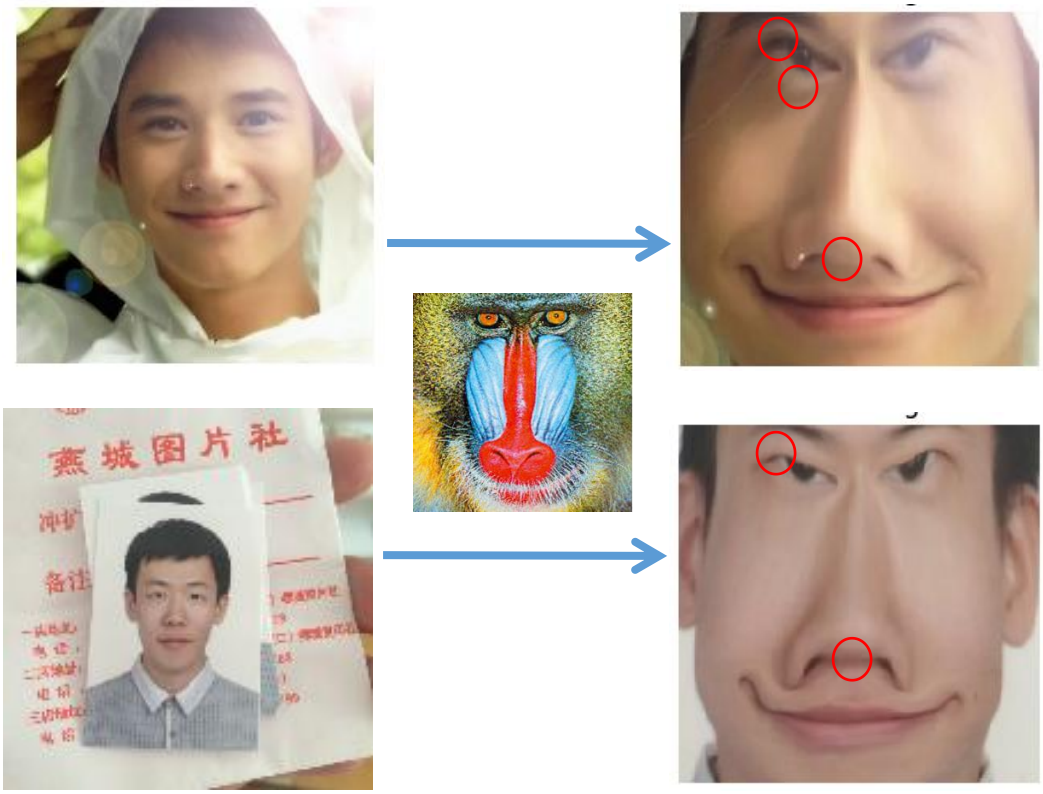
①左右眼各 2 个特征点：两个眼角(2x2=4)

②鼻子 2 个特征点：左右鼻翼(2)

③嘴巴 7 个特征点：将嘴唇下轮廓从左至右六等分，取 7 个点(7)

[共 13 个特征点]

结果如下：



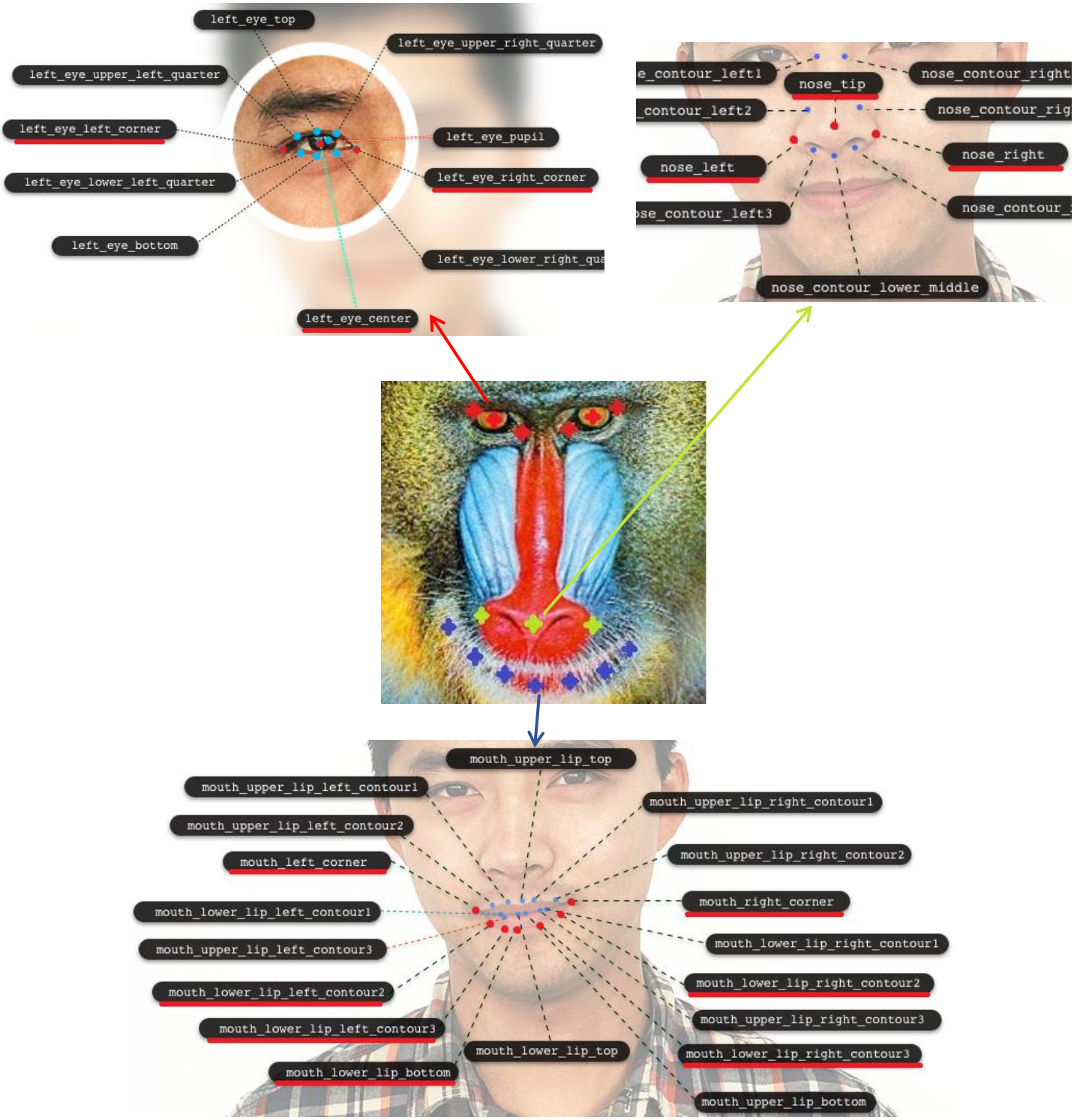
可以看出，嘴唇不平滑的问题解决了。

接下来需要优化的是眼睛与鼻子。眼睛有点呈“z”字状，而鼻子下端呈“n”字形。按理来说，理想的眼睛形状不应该有弯折，而鼻子的形状应为“m”字形。因此，我们接下来对眼睛和鼻子的特征点选取进行了改进。

5.2.3 16 个特征点（眼睛 3x2 + 鼻子 3 + 嘴巴 7）

首先，我们加入了眼睛中心作为特征点，试图解决眼睛形状的扭曲问题。其次，要使鼻子形状为“m”字形，我们加入了鼻头作为特征点。

这样一来，我们选取了 16 个特征点进行局部仿射变换：



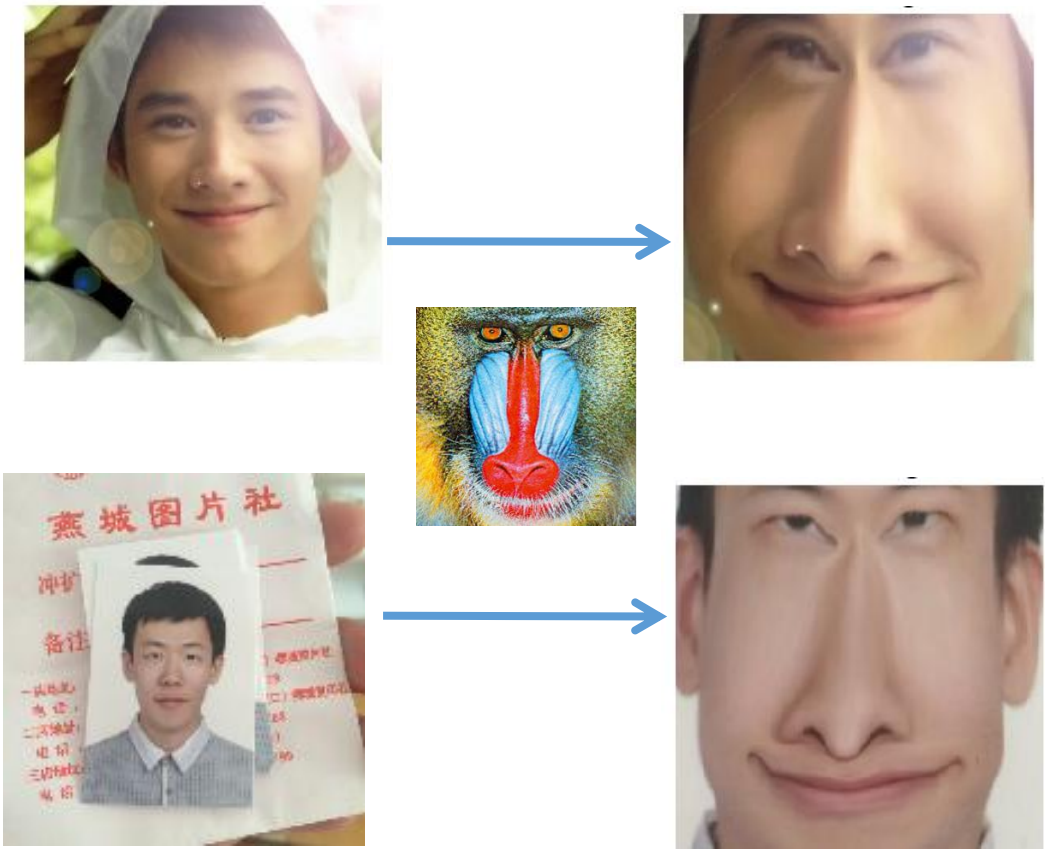
①左右眼各 3 个特征点：两个眼角及眼睛中心(3x2=6)

②鼻子 3 个特征点：左右鼻翼及鼻头(3)

③嘴巴 7 个特征点：将嘴唇下轮廓从左至右六等分，取 7 个点(7)

[共 16 个特征点]

结果如下：



已经能达到较好的效果。

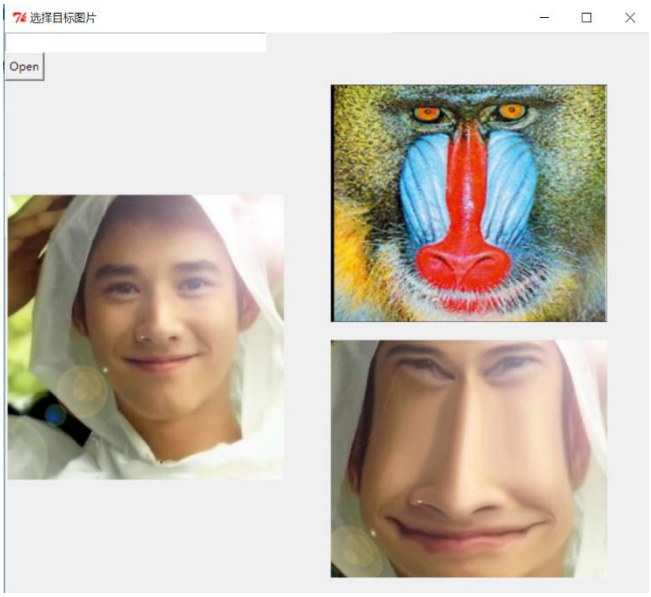
六 程序使用方法

1. 运行命令：

>> python gui.py

在弹出的窗口中点击“open”按钮选择需要变换的 jpg 图片文件（example1.jpg 或者 example2.jpg）。若为其他图片格式，建议先自行转化为 jpg 格式。

2. 等待几秒钟，界面上会显示原图像与变换后的图像，并与猩猩进行对比。注：变换过程需要联网，变换速度与网络状况有关。



七 小组各成员工作内容

杨雨樵：GUI 的实现；

周之烁：局部仿射变换算法的实现；

欧阳尔立：API 调用以及报告的撰写。