1设计算法、交互和界面，实现人脸到狒狒的形变。作业的基本算法内容在课堂上讲解，参考课件《9 图像数据基本变换 5 空间变换.pdf》及有关形变的学术资料实现最优效果。

2作业以小组形式，每个小组不能超过3个人。提交作业时只要一个代表提交就可以。记住：不要多人重复提交。

3 提交内容包括：（1）代码，如果有可执行文件同时提交可执行文件；建议使用python，matlab，C/C++等常用开发语言环境；代码要有非常清晰的注释。代码要有非常清晰的注释。（2）数据（如果有用到）。（3）一份报告：在报告中清晰描述问题和数据，数据处理的各个步骤及中间结果，代码结构，开发环境，可执行文件使用手册等细节问题；要求在报告中说明每位同学的贡献和工作内容。

4 要求2017-05-18 前由小组一名代表在elearning上提交小组名单，31日前提交结果。

输入：jpg图像（而不是png。若为png图像，建议先转换成jpg格式）

**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）。

**算法**

**数据处理的各个步骤及中间结果**

# 手动定位的16个狒狒脸特征点

# 包括：

# 1、左右眼各3个特征点：两个眼角及眼球中心(3x2=6)

# 2、鼻子3个特征点：左右鼻翼及鼻尖(3)

# 3、嘴巴7个特征点：将嘴巴从左嘴角至右嘴角六等分，取7个点(7)

# 利用旷视Face++的API定位16个人脸特征点

# 包括：

# 1、左右眼各3个特征点：两个眼角及眼球中心(3x2=6)

# 2、鼻子3个特征点：左右鼻翼及鼻尖(3)

# 3、嘴巴7个特征点：将嘴唇下轮廓从左嘴角至右嘴角六等分，取7个点(7)

**特征点（局部仿射点）的选取**

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：



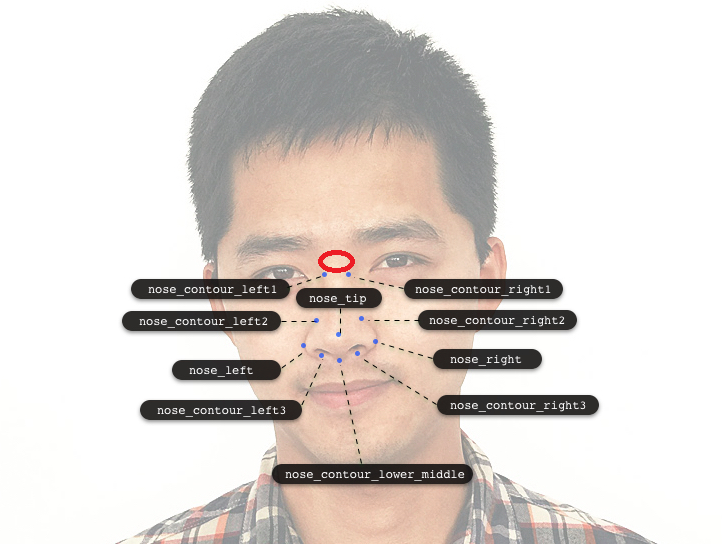
①左右眼各2个特征点：两个眼角(2x2=4)

②鼻子3个特征点：左右鼻翼及鼻梁(3)

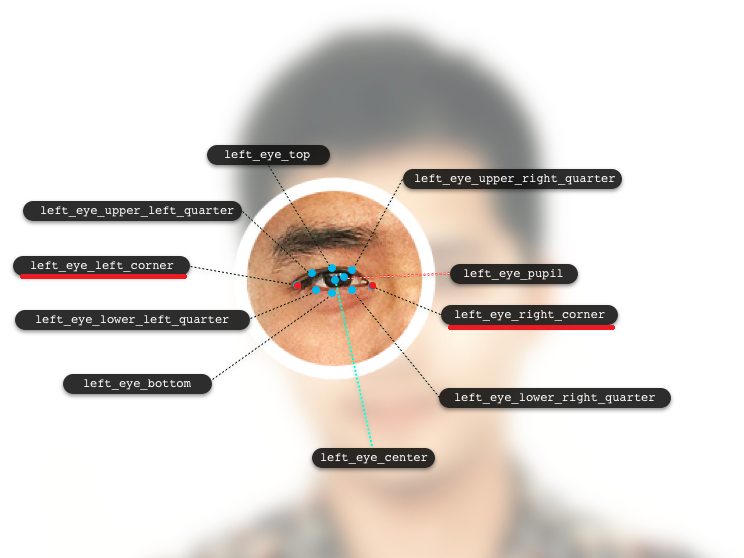
③嘴巴5个特征点：将嘴巴从左嘴角至右嘴角四等分，取5个点(5)

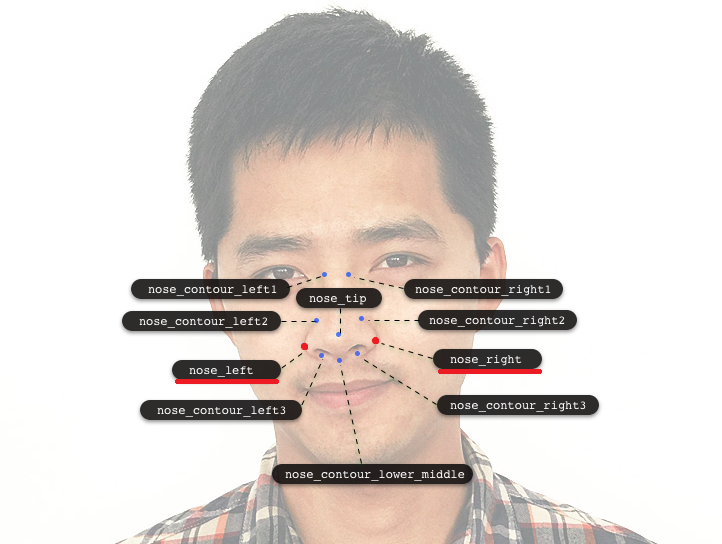
共12个特征点

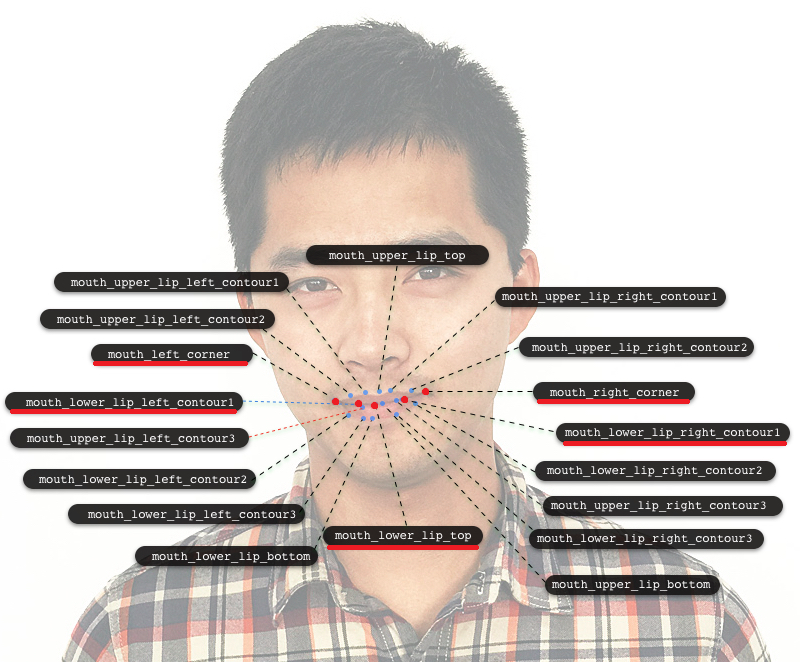
但在Face++识别出来的人脸特征点中，并不包括鼻梁。而且鼻梁与内眼角的距离很近，不考虑鼻梁对结果的影响很小。



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：







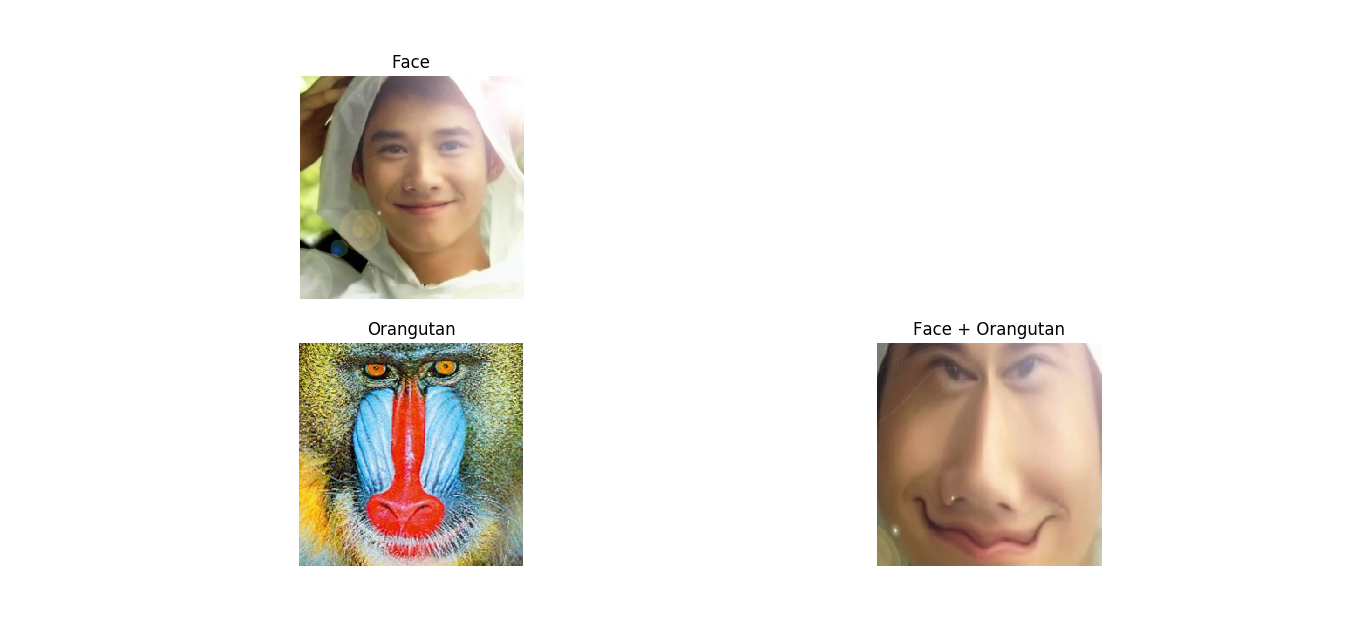
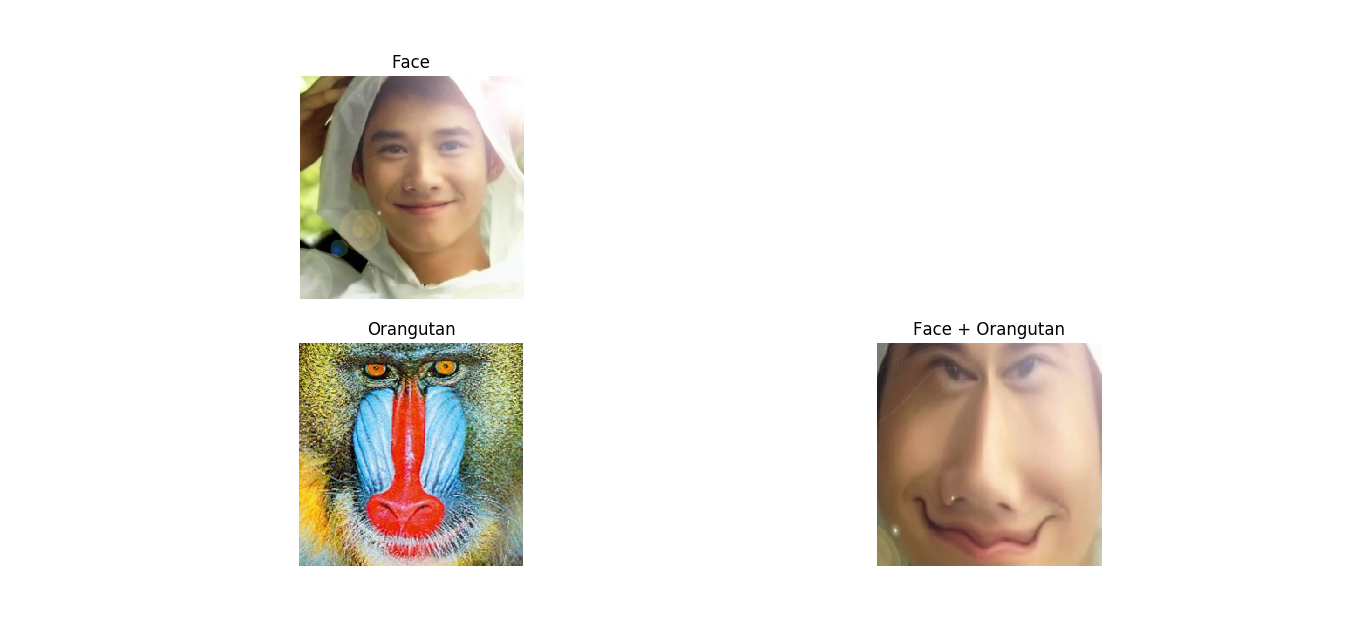
①左右眼各2个特征点：两个眼角(2x2=4)

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。

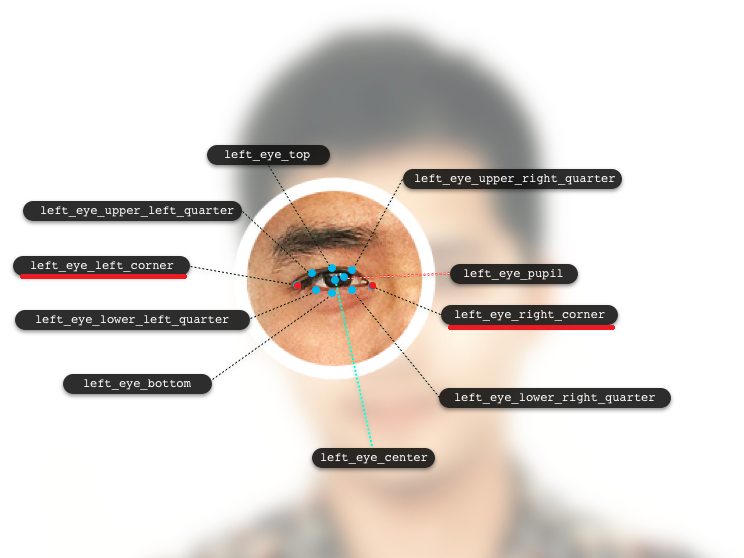
2、选取上下嘴唇交界线上的点作为局部仿射点时，嘴唇上的其他点受鼻子特征点的影响很大，这也是非特征点区域上凸的原因。

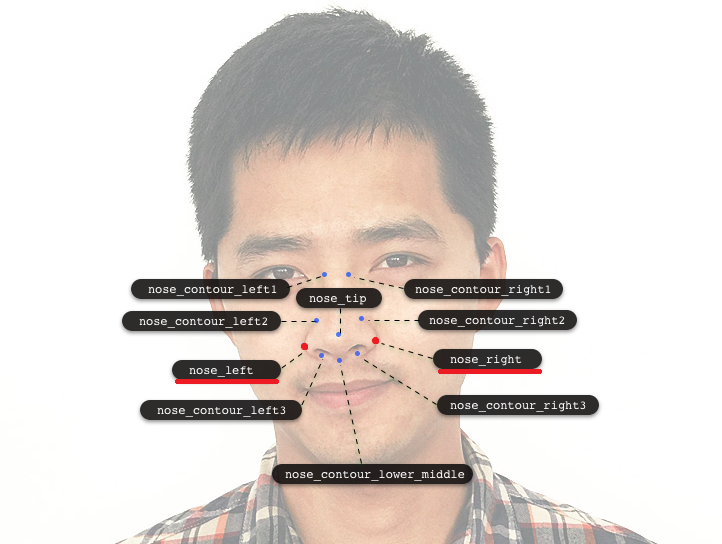
因此，我们接下来对嘴唇的特征点选取进行了改进。

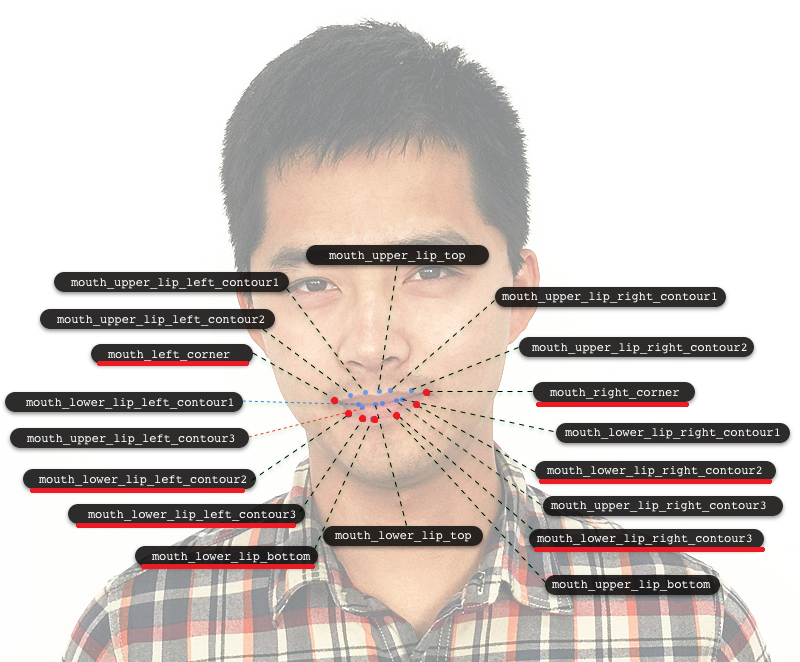
2、13个特征点（眼睛2x2 + 鼻子2 + 嘴巴7）

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：







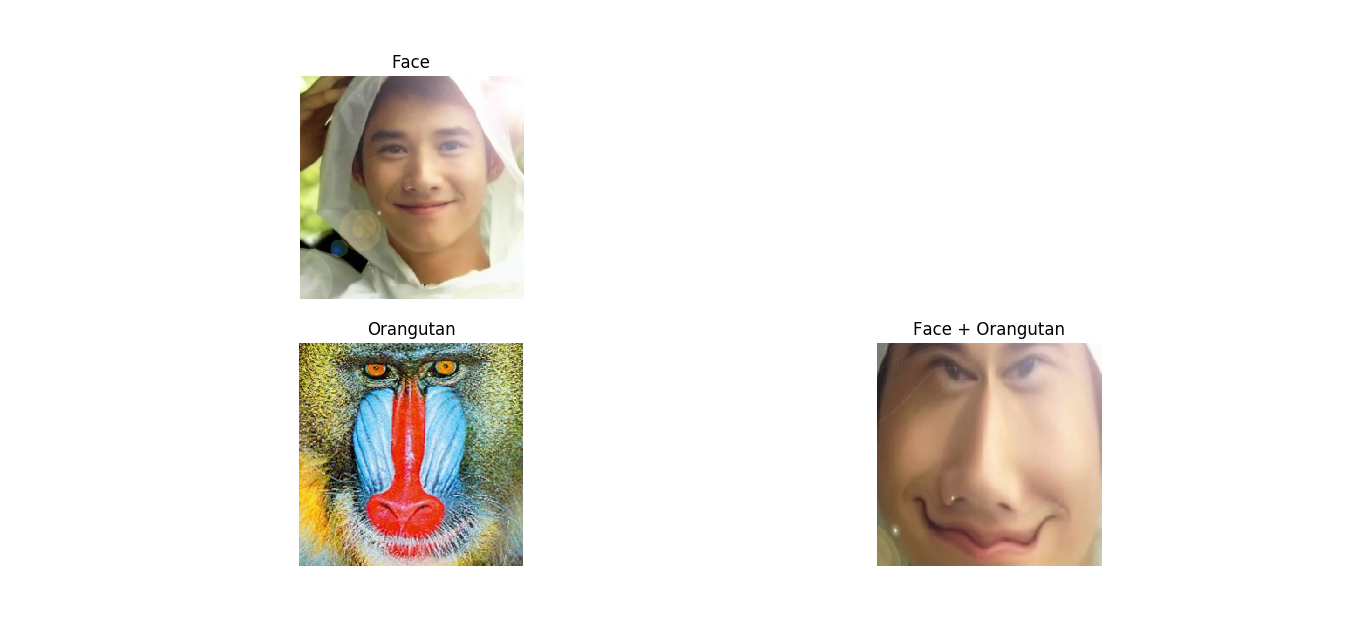
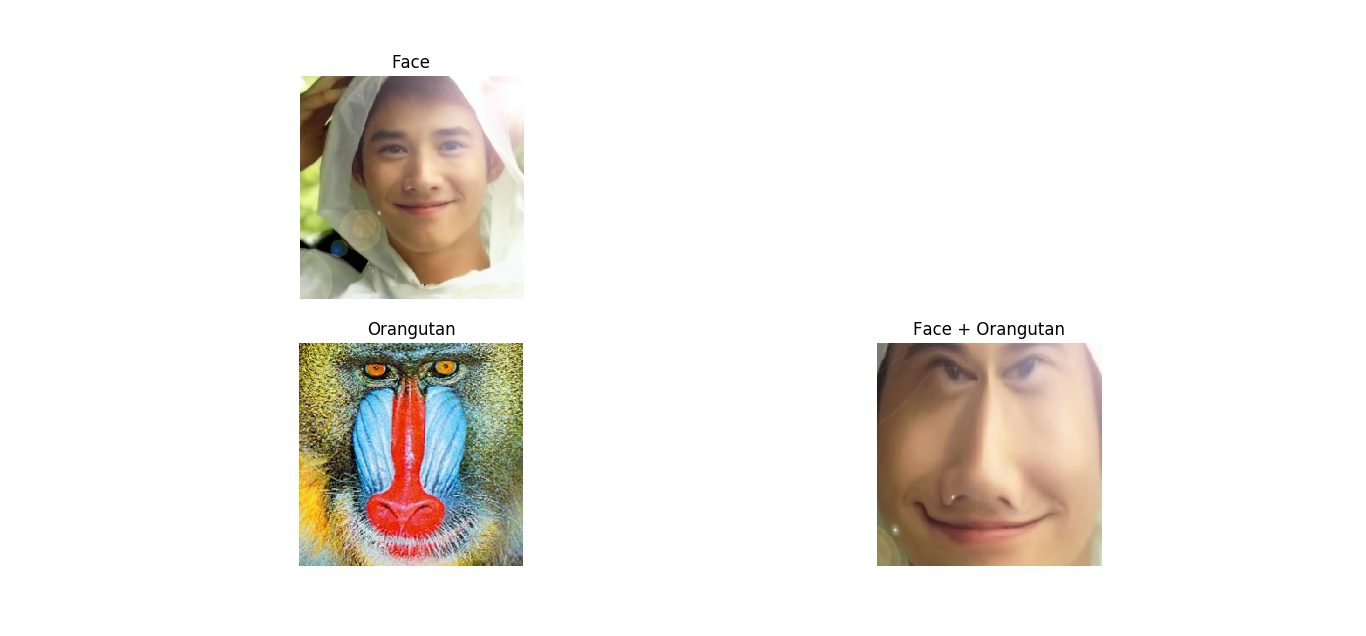
①左右眼各2个特征点：两个眼角(2x2=4)

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。

3、16个特征点（眼睛3x2 + 鼻子3 + 嘴巴7）

这样一来，我们选取了13个特征点进行局部仿射变换：

**代码结构**

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。

GUI部分：

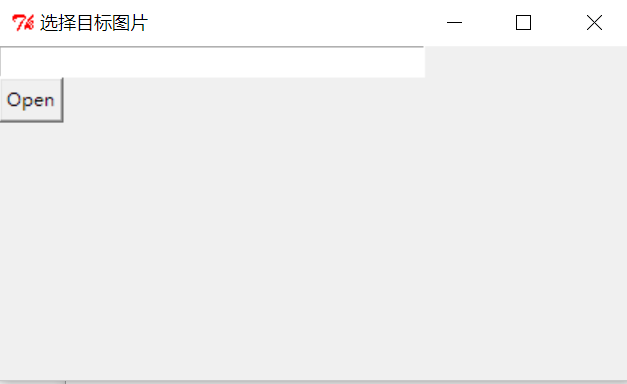
1.定义callback函数，描述将要显示的图片的大小位置，并对输入的图片调用change函数进行变换。

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。

**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。



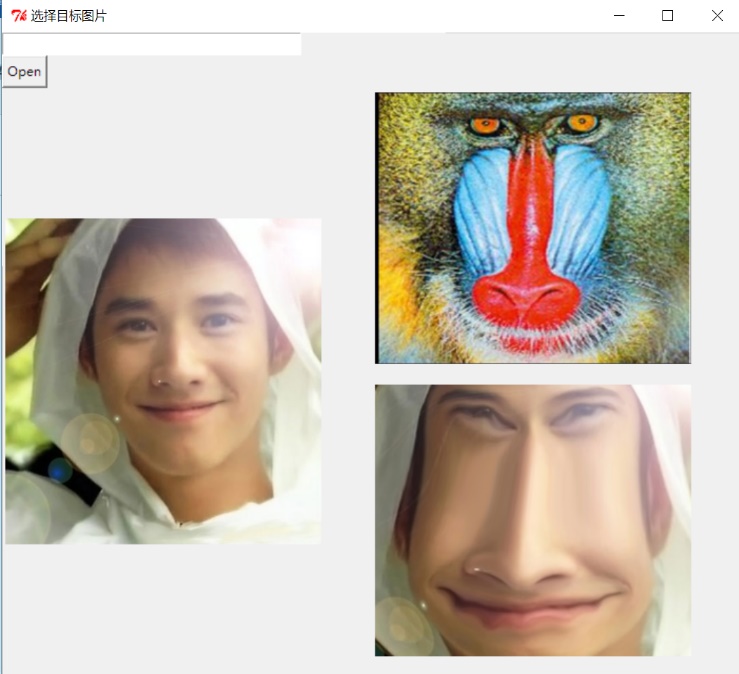
**程序使用方法**

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图片文件（example1.jpg或者example2.jpg）。

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。



**每位同学的贡献和工作内容**

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。