

第一次数值分析大作业实验报告

1 需求分析

本次大作业的目标是编写扭曲变形程序，可以对人脸图像进行扭曲变形。

要对图像进行操作，首先要对图片进行读入读出，并且为了方便使用者，也需要基本的应用程序界面。

要对人脸进行数学描述，则需要人脸关键点检测算法或者将已有的关键点数据读入。

由于不同图片的大小不同，其中人脸的大小、位置、角度也各不相同，所以需要对手关键点进行归一化处理。

要进行扭曲，需要选择相应的变形函数，在本次大作业中我选择了 TPS 与 B 样条两种方法。要对 TPS 进行求解还需要求解线性方程组。

在求出 TPS 或 B 样条映射后，由于结果并非整数，所以无法对映到已有像素点上，需要进行插值，在本次大作业中我使用了最近邻插值、双线性插值、双三次插值算法，用户可以对插值算法进行选择。

2 方案基本原理

3.1 变形函数

TPS (Thin plate spline) 变形

寻找一个通过所有控制点的光滑曲面 $f(x, y)$ ，使得能量函数 I 最小的插值模型。

$$I_f = \iint_{R^2} \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx dy$$

具体求解：

给定 n 个控制点 $P_i = (x_i, y_i)$ ，记

$$K = \begin{bmatrix} 0 & U(r_{12}) & \cdots & U(r_{1n}) \\ U(r_{21}) & 0 & \cdots & U(r_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ U(r_{n1}) & U(r_{n2}) & \cdots & 0 \end{bmatrix} \quad P = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \quad L = \begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix}$$

给定 n 个目标点 $P'_i = (x'_i, y'_i)$ ，记

$$V = \begin{bmatrix} x_1' & x_2' & \cdots & x_n' \\ y_1' & y_2' & \cdots & y_n' \end{bmatrix} \quad Y = \begin{bmatrix} V & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \end{bmatrix}^T$$

径向基函数为:

$$f(x, y) = [f_x(x, y), f_y(x, y)]^T = a_1 + a_x x + a_y y + \sum_{i=1}^n w_i U(|P_i - (x, y)|)$$

其中,

$$U(r) = \begin{cases} r^2 \log(r^2), & r \neq 0 \\ 0, & r = 0 \end{cases}$$

系数为线性方程组 $L[w, a_1, a_x, a_y]^T = Y$ 的解。

B 样条变形

给定 $(m+n+1)$ 个平面或空间顶点 $P_i (i=0, 1, \dots, m+n)$, 称 n 次参数曲线段:

$$P_{k,n}(t) = \sum_{i=0}^n P_{i+k} G_{i+n}(t), t \in [0, 1]$$

为第 k 段 n 次 B 样条曲线段 ($k=0, 1, \dots, m$), 这些曲线段的全体称为 n 次 B 样条曲线, 其顶点 $P_i (i=0, 1, \dots, n+m)$ 所组成的多边形称为 B 样条曲线的特征多边形。

其中 $G_{i+n}(t)$ 称为基函数。

设 $C(t)$ 是一段 n 次 B 样条曲线, 控制点 P_k 被移动到了新的位置 $P_k + v$, 新曲线为:

$$D(t) = C(t) + v G_{k,n}(t)$$

曲线仅在 $G_{k,n}(t)$ 不为零的区间内发生了改变。

推广到二维后可以得到:

$$\Delta x = \sum_{l=0}^n \sum_{m=0}^n G_{l,n}(u) G_{m,n}(v) \Delta x_{p_{i+l,j+m}}$$

$$\Delta y = \sum_{l=0}^n \sum_{m=0}^n G_{l,n}(u) G_{m,n}(v) \Delta y_{p_{i+l,j+m}}$$

其中:

$$\begin{aligned}
 u &= \frac{x}{N_x} - \left\lfloor \frac{x}{N_x} \right\rfloor \\
 v &= \frac{y}{N_y} - \left\lfloor \frac{y}{N_y} \right\rfloor \\
 i &= \left\lfloor \frac{x}{N_x} \right\rfloor - 1 \\
 j &= \left\lfloor \frac{y}{N_y} \right\rfloor - 1
 \end{aligned}$$

N_x 、 N_y 分别为 x 、 y 方向上的区间长度。

本次大作业中我将 n 取为 3，三次 B 样条基函数为：

$$\begin{cases}
 G_{0,3}(t) = \frac{1}{6}(-t^3 + 3t^2 - 3t + 1) \\
 G_{1,3}(t) = \frac{1}{6}(3t^3 - 6t^2 + 4) \\
 G_{2,3}(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \\
 G_{3,3}(t) = \frac{1}{6}t^3
 \end{cases}$$

其中，

$$t \in [0,1]$$

3.2 插值方式

最近邻插值

最近邻插值即把距离该点最近的整数点的 RGB 值近似为该点的 RGB 值。在实际实现中，还需要考虑坐标超出图片范围的情况，此时则近似为最近的边界点的 RGB 值。这种插值方法速度快但精度低，不够实用。

双线性插值

在两个方向（ x 轴、 y 轴）分别进行一次线性插值，需要使用该点相邻的四个点的 RGB 信息并做加权运算。用数学公式可表示如下：

$$f(x,y) = \begin{bmatrix} 1-u \\ u \end{bmatrix}^T \begin{bmatrix} f(i,j) & f(i,j+1) \\ f(i+1,j) & f(i+1,j+1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

其中，需要进行插值的非整数点坐标为 (x, y) ， $x=i+u, y=j+v$ ， i, j 为整数， f 为坐标到 RGB 的映射。

双线性内插值算法得到的图像质量较高，基本没有像素值不连续的情况，计算速度也还可以接受。然而此算法具有低通滤波器的性质，使高频分量受损，所以可能会使图像轮廓在一定程度上变得模糊。

双三次插值

与双线性插值类似，但要使用周围 16 个点进行插值。数学表达如下：

$$f(i+u, j+v) = ABC^T$$

$$A = [S(u+1), S(u), S(u-1), S(u-2)]$$

$$C = [S(v+1), S(v), S(v-1), S(v-2)]$$

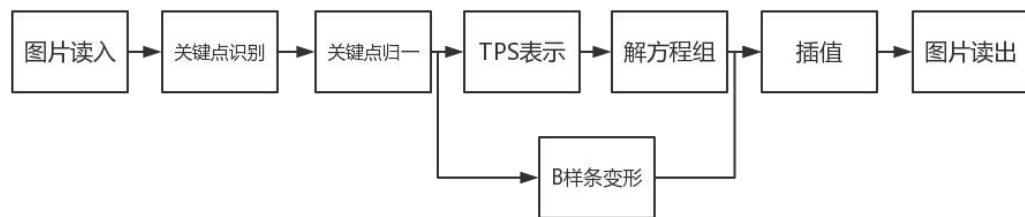
$$B = f(i-1:i+2, j-1:j+2)$$

$$S(x) = \begin{cases} 1 - 2|x|^2 + |x|^3, & |x| \leq 1 \\ 4 - 8|x| + 5|x|^2 - |x|^3, & 1 < |x| < 2 \\ 0, & \text{otherwise} \end{cases}$$

其中各符号含义同上， $S(x)$ 为三次插值基函数。由此算法得到的插值效果很好，但计算时间很长。

3 方案设计

本次大作业涉及的模块及实现的流程如图：



部分模块的具体设计如下：

图片读入读出：

由于本次大作业我使用了 QT 进行开发，为了可以在界面更合适地显示以及便于后续像素点的 RGB 运算，我使用了 QT 中的 QImage 类对图片进行处理，而没有使用 Opencv 进行读写。

关键点识别：

我使用了基于 opencv 的识别算法，借助 Dlib 库中的 shape_predictor_68_face_landmarks 实现了对人脸 68 个关键点的探测，其输出与已有图片的关键点信息相一致，使得新旧图片关键点信息的读入接口可以兼容。

关键点归一化：

本步骤是决定扭曲效果的重要一步，只有将目标人脸的关键点合适地映射到原有人脸上，才可以进行下一步扭曲。

1、位置匹配：

我使用了控制关键点均值一致的方法，对目标关键点进行整体平移，使中心与原关键点中心重合。

2、大小匹配：

我采用了控制关键点到中心点的欧氏距离均值一致（即方差）的方法，对目标关键点根据其到中心的距离进行加权放缩，权重为该目标点到中点的距离与目标点的平均距离之比，距离中心近的点放缩较小，距离中心远的点反之，则可以较好地使两组点大小适配。

3、角度匹配：

通过计算出人脸太阳穴处两关键点(第 0 号与第 16 号)连线与 x 轴的夹角便可确定该脸的方向，再通过三角运算解出旋转到相同角度后的坐标，旋转时保持到中心距离恒定。但该方法只能实现纸面内旋转，由于缺乏立体信息，无法对关键点进行进一步匹配，因而在处理部分侧脸的照片时依旧会有失真，但保证了图片不会因为目标人脸的角度而产生旋转。

解方程组：

本次大作业使用 QR 分解的方法对线性方程组进行求解，具体数学步骤不再赘述。

B 样条变形：

我采用了迭代逼近的方法，不断根据网格点的位移计算出关键点的位置，再根据原关键点到目标关键点的距离 d 调整网格点的位移，直到所有关键点对应的 d 之和小于精度要求，便可退出循环。

在生成新人脸时我采用了正向映射、逆向映射、局部逐个变形三种方法。

1、正向映射：

在求出网格点位移后，遍历待扭曲的人脸图片上的像素点，通过 B 样条公式映射到新的人脸上，如果映射到的点已经被映射过，则取均值；如果在遍历后仍有点没有被映射到，则取为最近 16 点 RGB 的均值。

2、逆向映射：

在求网格点位移的时，通过目标关键点到原关键点的距离 d' 而不是 d 来调整网格点的位移，这样就得到了原求取目标的逆映射，也就是我们所要求的新人脸上每个点在待变形人脸上的位置的映射。遍历新人脸，对于每一个点通过这个映射找到它在待变形人脸上的位置，再进行插值得到 RGB 值。

3、局部逐个变形：

这个方法不需要迭代逼近，而是对关键点逐一进行 B 样条逆向映射。为了尽可能充分变形且变形区域又不会过度重叠，所以需要控制点网格大小进行动态调节，使得相邻的关键点变形时可以刚好重合接近四分之一变形面积，从而变形后的人脸尽可能光滑。

4 误差分析

4.1 舍入误差

本次大作业计算过程中的变量均为 `double` 类型，可以存储 15 位有效数字，又因为大部分点的坐标及 RGB 值的计算都是 100~1000 之间的数字，所以相对误差限为 $\frac{1}{2} \times 10^{-12}$ ；但在最后输出图片像素点的 RGB 表示中只能保留到整数，所以

总的相对误差限为 $\frac{1}{2}$ 。

4.2 截断误差

最近邻插值

已知：

$$\varepsilon(A^*) \approx \sum_{k=1}^n \max \left| \frac{\partial f}{\partial x_k} \right| \varepsilon(x_k^*)$$

本插值方法会把非整点近似为最近的整点，所以点的坐标的误差限为：

$$\varepsilon(x^*) = \varepsilon(y^*) = \frac{1}{2}$$

所以截断误差限为：

$$\varepsilon(A^*) \approx \frac{1}{2} \left(\max \left| \frac{\partial f}{\partial x} \right| + \max \left| \frac{\partial f}{\partial y} \right| \right)$$

其中 f 为点的坐标到 RGB 值的映射。

双线性插值

已知双线性插值公式及插值余项公式如下：

$$f(x, y) = \begin{bmatrix} 1-u \\ u \end{bmatrix}^T \begin{bmatrix} f(i, j) & f(i, j+1) \\ f(i+1, j) & f(i+1, j+1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

首先考虑前两个矩阵，即在 x 轴方向的插值，可得当 $y=j$ 时，在 x 方向的截断误差为：

$$|R_x(x, j)| \leq \frac{1}{2!} \max_{i \leq x < i+1} \left| \frac{\partial^2 f}{\partial x^2} \right| \max_{0 \leq u < 1} |u(1-u)| = \frac{1}{8} \max_{i \leq x < i+1} \left| \frac{\partial^2 f}{\partial x^2} \right|$$

同理：

$$|R_x(x, j+1)| \leq \frac{1}{8} \max_{i \leq x < i+1} \left| \frac{\partial^2 f}{\partial x^2} \right|$$

再考虑最后一个矩阵，并使用公式：

$$\varepsilon(A^*) \approx \sum_{k=1}^n \max \left| \frac{\partial f}{\partial x_k} \right| \varepsilon(x_k^*)$$

可得在 x 方向的总截断误差为：

$$|R_x(x, y)| \leq (1-\nu)|R_x(x, j)| + \nu|R_x(x, j+1)| \leq \frac{1}{8} \max_{i \leq x < i+1} \left| \frac{\partial^2 f}{\partial x^2} \right|$$

同理在 y 方向：

$$|R_y(x, y)| \leq \frac{1}{8} \max_{j \leq y < j+1} \left| \frac{\partial^2 f}{\partial y^2} \right|$$

总误差限为：

$$|R(x, y)| \leq |R_x(x, y)| + |R_y(x, y)| = \frac{1}{8} \left(\max_{i \leq x < i+1} \left| \frac{\partial^2 f}{\partial x^2} \right| + \max_{j \leq y < j+1} \left| \frac{\partial^2 f}{\partial y^2} \right| \right)$$

双三次插值

可以近似认为双三次插值在 x 、 y 两个方向上满足三次样条插值条件，具有埃尔米特余项：

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)^{n+1}$$

且每一段分别在两个边界点上各满足两个插值条件

$$R_3(x) = \frac{f^{(4)}(\xi)}{4!} (x - x_i)^2 (x - x_{i+1})^2$$

与双线性的分析类似，可以得到：

$$|R_x(x, y)| \leq \frac{5}{384} \max_{i-1 \leq x < i+2} \left| \frac{\partial^4 f}{\partial x^4} \right|$$

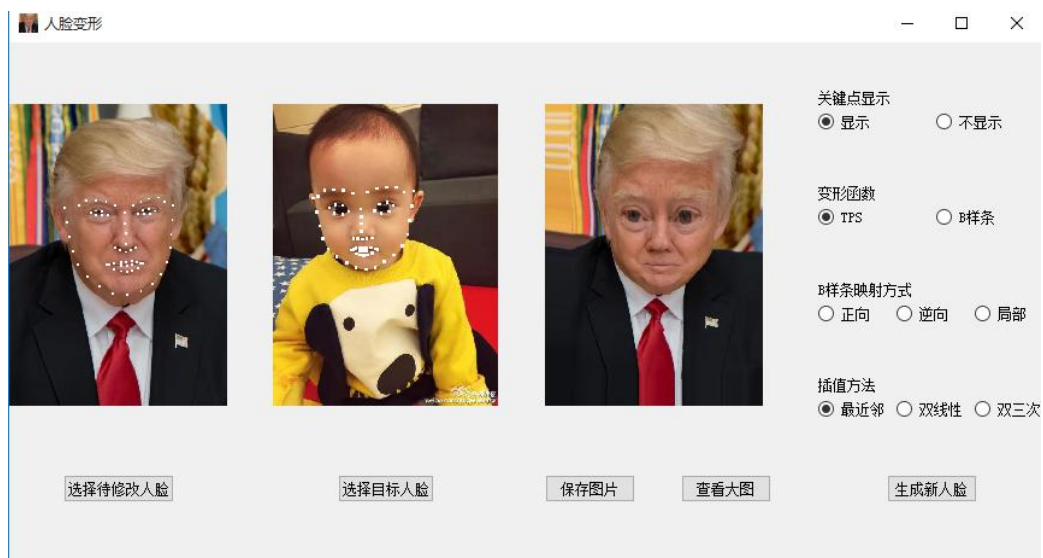
$$|R_y(x, y)| \leq \frac{5}{384} \max_{j-1 \leq y < j+2} \left| \frac{\partial^4 f}{\partial y^4} \right|$$

因而总的误差限为：

$$|R(x, y)| \leq |R_x(x, y)| + |R_y(x, y)| = \frac{5}{384} \left(\max_{i-1 \leq x < i+2} \left| \frac{\partial^4 f}{\partial x^4} \right| + \max_{j-1 \leq y < j+2} \left| \frac{\partial^4 f}{\partial y^4} \right| \right)$$

5 运行效果

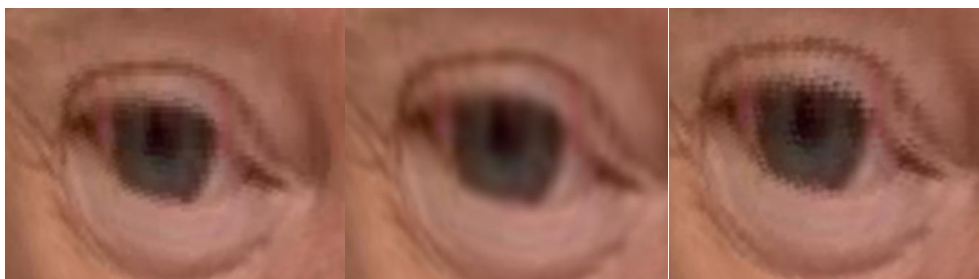
5.1 TPS 变形效果



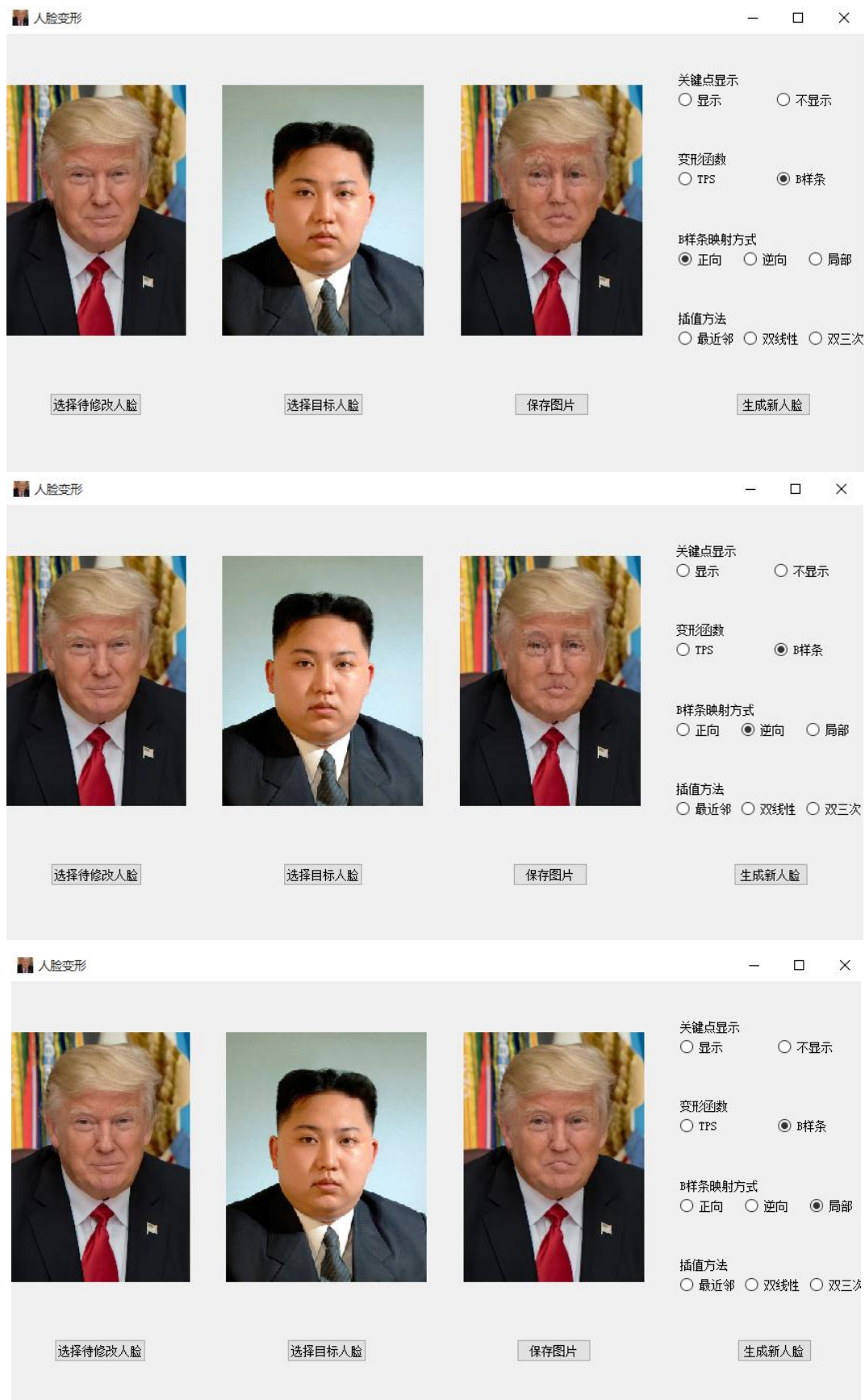
程序界面如图，可对关键点是否显示、变形函数种类、B样条的映射方式、插值方法进行选择。生成新人脸后还可以保存图片或查看大图。

经测试发现，三种插值方式实际上对整体图片的变形效果影响不大，只有在将某一局部区域放大后差别才显现出来。

以下三图从左到右依次为最近邻插值、双线性插值、双三次插值后的效果。可见双三次插值后的精度最高，效果最好；双线性插值会造成模糊；最近邻插值的效果也不错，考虑到其效率较高，所以更为实用。



5.2 B样条变形效果



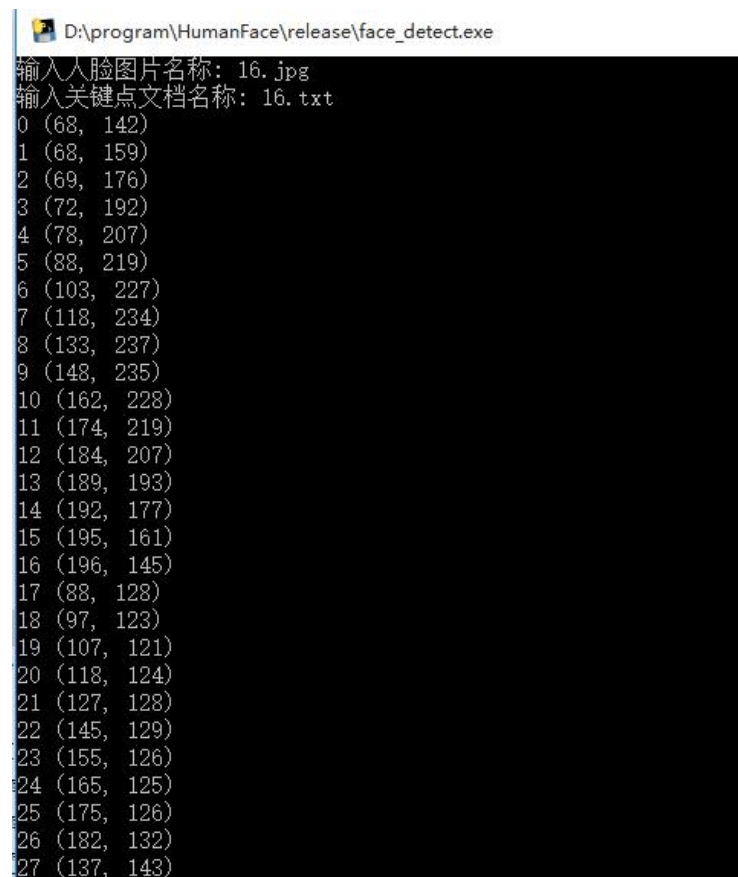
以上三图从上到下分别为：正向、逆向、局部逐点的 B 样条变形。

对比以上三图可以发现，正向映射对人脸的匹配程度最高，但由于对没有被映射到的点用周围点 RGB 的均值进行估计，所以图像比较不连续，会有锯齿状条纹产生；逆向映射的人脸匹配度和图像质量都居中，而且由于是通过网格逆向映射，生成的图片在网格边界处也会有不连续的现象；局部逐点变形的图像质量最高，可与 TPS 媲美，但匹配程度很差，变形不充分。

综上，三种方法各有优劣，但都不是最优，因此 B 样条方法实现人脸变形还可以进一步改善。

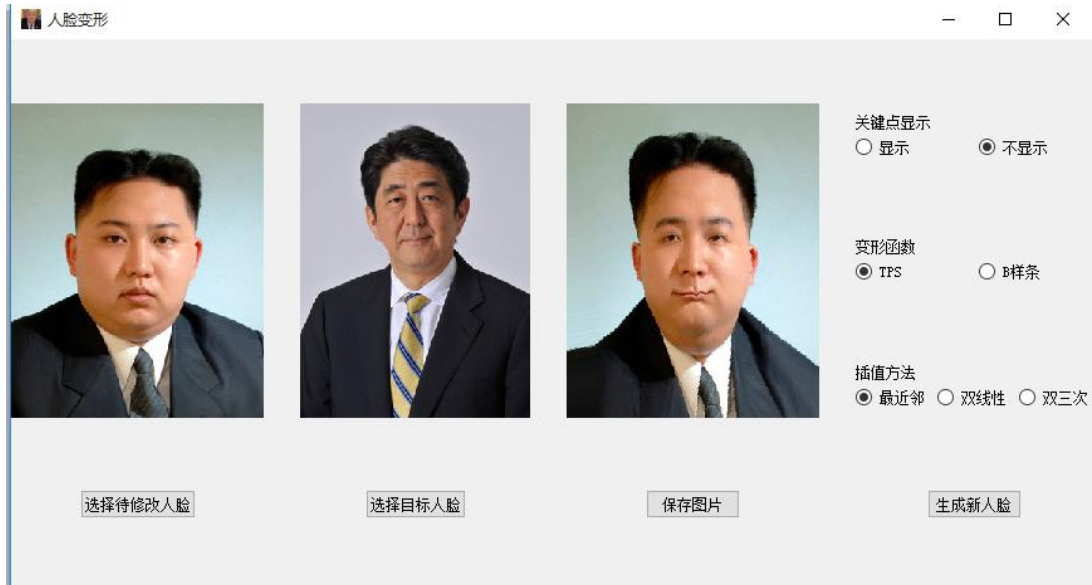
5.3 自主实现人脸关键点检测

运行已经打包好的 python 脚本，对相同目录下的新图片进行人脸关键点检测，并输出。



```
D:\program\HumanFace\release\face_detect.exe
输入人脸图片名称: 16.jpg
输入关键点文档名称: 16.txt
0 (68, 142)
1 (68, 159)
2 (69, 176)
3 (72, 192)
4 (78, 207)
5 (88, 219)
6 (103, 227)
7 (118, 234)
8 (133, 237)
9 (148, 235)
10 (162, 228)
11 (174, 219)
12 (184, 207)
13 (189, 193)
14 (192, 177)
15 (195, 161)
16 (196, 145)
17 (88, 128)
18 (97, 123)
19 (107, 121)
20 (118, 124)
21 (127, 128)
22 (145, 129)
23 (155, 126)
24 (165, 125)
25 (175, 126)
26 (182, 132)
27 (137, 143)
```

使用生成的数据进行人脸变形，可见效果良好：



6 小结

通过这次大作业的实践，让我再一次温习了程序设计和界面开发，也对课上所学的数值分析的知识及其实际应用有了更多认识。在学习 TPS、B 样条以及关键点探测的过程中，进一步提高了自己查阅资料的能力，也使我可以从全新的角度去认识图像处理。在与曾哲妮同学（学号：2016013303）的讨论中我获得了很多关于 B 样条变形的启发，在此对她表示感谢。

7 参考文献

- [1] Dlib 提取人脸特征点（68 点，opencv 画图），
<https://blog.csdn.net/zmdsjtu/article/details/53454071>
- [2] Gregory Sharp, Marta Peroni, Rui Li, James Shackelford, and Nagarajan Kandasani. Evaluation of plastimatch b-spline registration on the empire10 data set. Medical Image Analysis for the Clinic A Grand Challenge, 2010.