# 人脸过渡大作业报告

## 1.基本功能介绍

此次大作业实现了从一张人脸照片到另一张人脸照片的过渡。

使用程序流程：

1. 选取控制点（特征点）：使用者可以选择已经存储于文件中的控制点，也可以自己手动选择控制点。在选择的时候需要注意，源图像（SI）中控制点和目标图像（DI）中控制点顺序需要对应（比如SI中先选眼睛再选鼻子，那么DI中也需要先选眼睛再选鼻子），其次，控制点的数量也需要保持一致，如若不然，选取的控制点无效，会被要求重新选取。

2. 显示Delaunay分割效果：完成本次作业的核心算法之一便是Delaunay三角分割。所以会显示Delaunay分割效果。

3. 选择产生多少张过渡图片：过渡图片的数量是使用者选择的。

4. 生成过渡图片并保存：这个过程可能需要一定的时间，程序运行完成之后可以在当前目录下查看产生的图片。

5. 展示过渡图片：每张图片停留1s。

6. 选择是否保存此次选择的特征点：若保存此次选择的特征点，下次运行程序之时可以直接调用保存后的文件。

## 2.设计思路

为了实现程序功能，需要完成以下过程：

2.1 人工选取控制点。控制点取在眉心、眼珠、鼻子两侧等特征比较明显的部位。

2.2 控制点选好之后进行Delaunay三角化。根据选择的特征点将一张图片剖分为若干个三角形。

2.3 根据控制点对应关系，得到SI剖分三角形和DI剖分三角形对应关系。

2.4 根据SI剖分三角形和DI剖分三角形对应关系和过渡系数得到过渡图像中三角形与源/DI剖分三角形对应关系。

2.5 根据三角形对应关系，求出仿射矩阵。

2.6 根据仿射矩阵，求出过渡图像中每一点仿射到源/DI中的点（很有可能是个浮点数）。不同点在不同三角形中，所以对应的仿射矩阵很可能是不一样的。

2.7 根据仿射得到点的坐标，再由双线性插值得到过渡图片中每一点在源/DI中像素值。

2.8 根据双线性插值结果和过度系数，求出过渡图片像素点的像素值。从而得到过渡图片。

2.9 改变过度系数，重新得到新的过渡图片，直至过渡完成。

## 3.原理

基本原理：

人脸过渡过程，可以使看作SI权重不断减小和DI权重不断增大，其本质是多次映射。图像权重体现为坐标和像素值权重。

可以这样来想，从SI直接变化到DI就，没有映射，没有过渡。但是从SI照片经过一张过渡照片，再到DI，就有一次过渡，这个过程中就会发生两次映射，分别是SI映射到过渡图片，DI映射到过渡图片。依此类推，当有两张过渡图片时，会发生四次映射。

在映射的过程中，如果用一个矩阵来代替，必然会导致有很大的误差，为了缩小这个误差，我们应该将图片分为若干个三角形或者是四边形（我选择是分为三角形），这样每个三角形中包含点的映射矩阵就由三角形的三个顶点来确定，而三角形的顶点就是由选取的控制点来确定的。

因为图像可以被剖分为许多三角形，所以我就拿三角形来举个例子：

源图中三角形三个点分别为【（0,0），（100,100）（100,0）】，DI中三个点分别为【（0,0），（200,200），（200,0）】。若只有一张过渡图片，那么与该三角形对应的过渡三角形应该为【（0,0），（150,150）（150,0）】。若有两张过渡图片，那么对应的两个三角形分别为【（0,0），（133,133,0）（133,0）】和【（0,0），（166,166），（166,0）】。依此类推，就可以知道产生n张过渡图片，每张过渡图片的三角形坐标。

得到映射矩阵之后，分别求出SI和DI到过渡图像的映射，最后再附加上像素值权重，便得到过渡图像。

3.1 控制点的选取。

控制点的选取直接影响了过渡图片的质量，根据前面的叙述，三角形映射矩阵是由控制点决定的。那么为什么控制点的选取要选择具有特征的点？

就拿人的眼睛来举例子，如果在SI和DI中，包含眼睛的三角形控制点不是眼睛，那么SI和DI分别向过渡图像映射之后，眼睛的位置可能不是重叠的，这就会导致过渡图像中眼睛数量翻了一倍。

3.2 三角剖分算法

图像三角化要求产生的三角形边不会相交，并且所有的三角形能够刚好完全覆盖整个图像。在实现此目的时，我选用了openCV中的Delaunay剖分，Subdiv2D这个类。

3.3 仿射

仿射的目的是要知道过渡三角形中每一个点对应SI/DI中三角形的点。

若已知，三角形A的三个顶点，分别为（x1，y1），（x2，y2），（x3，y3）对应三角形B的顶点为（a1，b1），（a2，b2），（a3，b3）

可以求出A->B的仿射矩阵。

Aff =

若已知三角形A中某点为(m,n)

可以求出其对应的点(c,d)

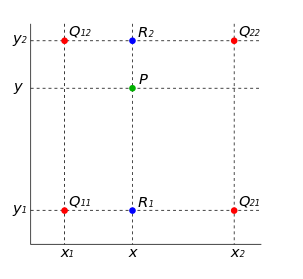
3.4 双线性插值

根据仿射变换求出对应点坐标是为了得到像素值。

得到的坐标多数情况下不是整数，而如何求取非整数（非网格点）处像素值呢？

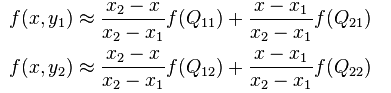
我采用的是双线性插值法：

如下图：假设P为我们要求的插值节点，Q11，Q12，Q21，Q22为网格点。



双线性插值方法是先在一个方向上插值（x方向），再在另一个方向上插值（y方向）。

举个例子，先在x上插值：



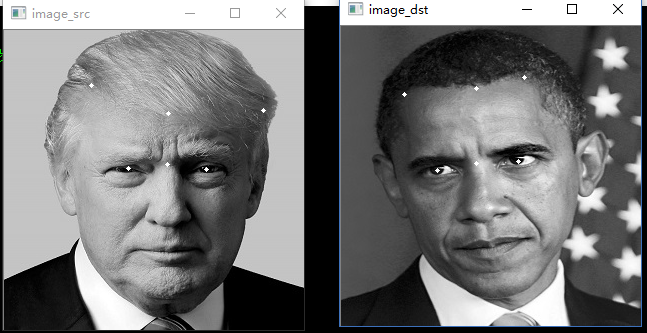
再在y上插值：

## 4.程序实现以及结果

4.1 控制点选取

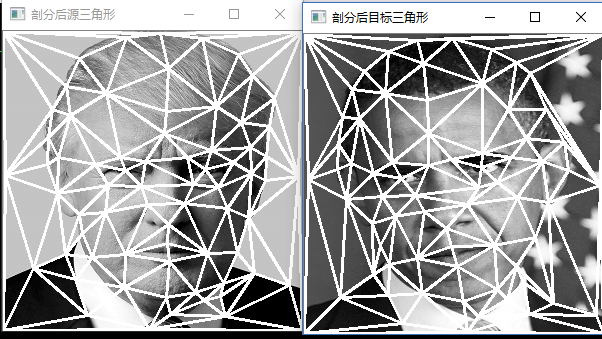
主要调用setMouseCallback()函数，得到鼠标左键点击处坐标。

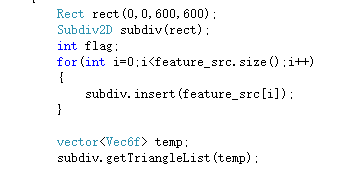
并且调用circle()函数将鼠标左键左键点击处点亮。

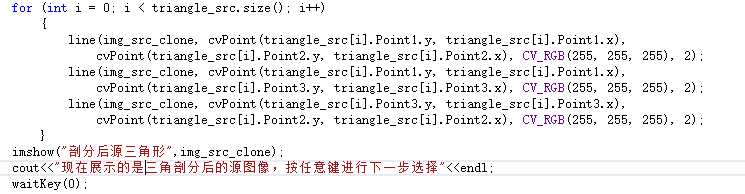


4.2 Delaunay三角化

利用openCV中的Subdiv2D类，将控制点插入后，就可以得到三角化后，三角形的坐标，然后我再将三角形的边画出来，得到以下结果。

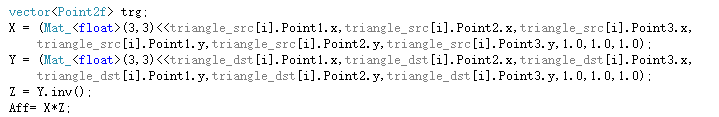






4.3 仿射矩阵计算以及坐标仿射

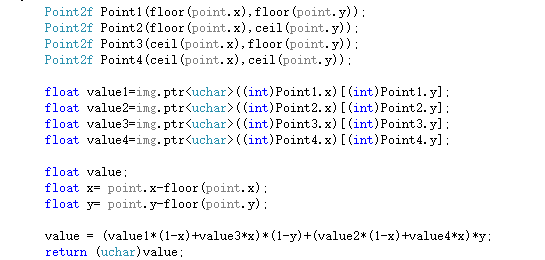
仿射矩阵计算：



坐标仿射：



4.4 双线性插值



## 5.实验结果以及对结果的讨论

由于产生的图片比较多，不放便一张一张的贴出来，所以找了一张有特点的。



以上图片出现了一个问题就是，某些区域会变得模糊，在这张图片中是下巴部分（原因肯可能是我在选取控制点时没选取好，下巴和下巴没有对应起来，也怪Trump是个双下巴，不好找。）

## 6.实验中遇到的问题

1. 特征点的选取必须要有对应顺序，如果顺序不对应，那么产生的过渡照片质量会很糟糕。

而顺序检验这一点程序无法做到，能做到的只有特征点数量检测。