计算机视觉第四次作业之附加题

朱明杰 15331441

一. 实验任务

1. 输入普通 A4 打印纸，上面可能有手写笔记或者打印内容，但是拍照时可能角度不正。输出已经矫正好的标准普通 A4 纸（长宽比为 A4 纸的比例），并裁掉无用的其他内容，只保留完整 A4 纸张。（略，这个报告已经写过了）
2. 输入两张图像，输出一系列渐变的图像。（附加题）

二. 实验工具

Visual studio 2015

CImg Library

三. 算法流程

1. 关键点提取与配准

   事实上这一步是最难的。一开始我想用现成的工具来实现关键点的提取与配准，在 Mathematica 中找到了一个基于 SURF 的 `ImageFeatureTrack` 函数。但是实际效果并不理想：

A =

;

B =

;

```
In[5]:= ImageCorrespondingPoints[A, B, MaxFeatures → 100]
        图像对应点                    最大特征
Out[5]= {{{42.2387, 18.7961}, {135.114, 18.2797}, {266.968, 380.266}, {302.383, 306.436},
    {300.458, 363.869}, {304.475, 202.998}, {262.664, 106.283}, {41.9088, 169.019}},
   {{29.9556, 274.84}, {306.885, 284.377}, {19.0785, 150.387}, {289.215, 98.2733},
    {16.5538, 258.113}, {73.5233, 380.739}, {236.469, 383.708}, {295.77, 125.71}}}
```

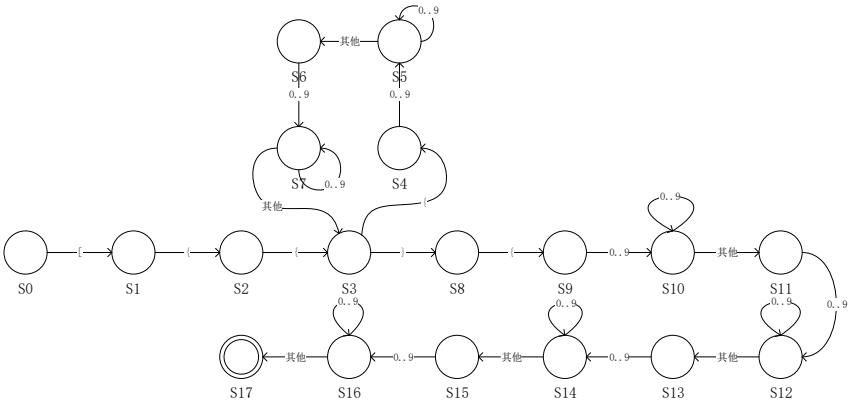
可以看出，两个图的对应点寻找只找出了 8 对点。该函数对于相似的图像能够起到比较好的效果，但是对于相似性不大的图片，效果就不那么好了。

就这样纠结了一周之后，老师提示了一下可以用 Face++的 API 来完成人脸的特征点检测，并且只需要做面部的渐变即可。所以最后我就用了这个。

值得一提的是，在查询解决方法的时候，我在 github 上找到了疑似某位学长的之前的作业。然而发现这位学长用的是 Matlab 打开图片，手动选取对应坐标点，而且这个方法已经被老师 diss 成了“比较 low 的方法”。所以最后我也没采用。

在 API 调用的时候，本来我想把 HTTP 请求与响应整合到整个流程里面的。后来在调用他的 API 的时候，发现虽然文档说免费的人脸特征点检测有 10QPS，但是纵使很慢地发出请求，响应返回的还是很有可能是 `CONCURRENCY_LIMIT_EXCEEDED`，很是恼火，但具体原因便不得而知了。所以在每个范例的文件夹下都有 begin.sh 和 end.sh 脚本来分别得到 begin.bmp 和 end.bmp 的特征点 JSON 文件 begin.json 和 end.json，而且须在 bash 下执行。

很遗憾，原生 C++（即使强如提出了通用线程同步机制的 C++11 标准）并没有直接的 JSON 解析器，而且就此问题而言，JSON 解析的需求比较固定。根据奥卡姆剃刀原则，本人根据编译原理上所学内容，构造了一个有限状态自动机来从 JSON 中得到坐标信息。该有限状态自动机又称土味 JSON 解析器。土味 JSON 解析器的有限状态自动机如下：



各状态区间说明：

S0-S3：过滤掉前面的信息。

S3-S5：读特征点 y 坐标。

S5-S7：读特征点 x 坐标

S9-S10：读取人脸宽度。

S11-S12：读取人脸左上角 y 坐标。

S13-S14：读取人脸左上角 x 坐标。

S15-S16：读取人脸高度。

2. Delaunay 三角剖分与 Boyer-Watson 算法

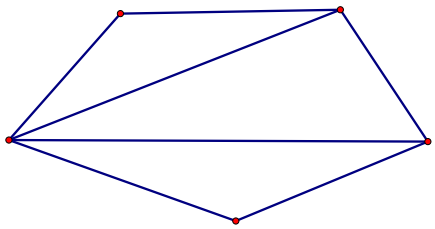
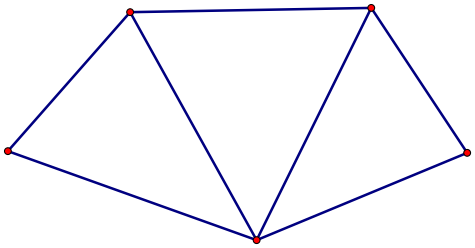
得到图像的特征点集之后，我们要对点集进行三角剖分。这里我们采用老师建议的 Delaunay 三角剖分。对于每一个三角形ΔABC，Delaunay 三角剖分不允许其它的点D在这三角形的外接圆内。点D在ΔABC外接圆内的充要条件为

$$\begin{vmatrix} A_x & A_y & A_x^2+A_y^2 & 1 \\ B_x & B_y & B_x^2+B_y^2 & 1 \\ C_x & C_y & C_x^2+C_y^2 & 1 \\ D_x & D_y & D_x^2+D_y^2 & 1 \end{vmatrix} > 0$$

其中要求，A、B、C三点**逆时针**排列。一开始我没有注意到这个要求，于是就被坑

了 QAQ。Delaunay 三角剖分较一般三角剖分而言，三角形构成比较匀称，不会出现特别。

Delaunay 三角剖分的优点如下：

 上图是一个非 Delaunay 三角剖分。我们可以看到，剖分出来的三角形长宽比比较夸张。在之后的形变中，可能会由于三角形坐标变化过大而让变化的效果看上去不自然。	 上图是一个 Delaunay 三角剖分。我们可以看到，剖分出来的三角形长宽比比较协调。

具体实现对一个点集的 Delaunay 三角剖分可以使用 Boyer-Watson 算法，这个维基百科上直接有伪代码，对着实现就行，其本质便是枚举调整。

3. 线性插值

   把动画拆成 RGB 三通道分别完成插值。

   假设我们的动画是一个时变矩阵$\mathbf{A}(t)$，其中$0 \le t \le 1$，而且我们已知起始矩阵$\mathbf{A}(0)$与终止矩阵$\mathbf{A}(1)$，那么，对于一个时刻t的矩阵$\mathbf{A}(t)$中的点P，先找到三角剖分的点集在时刻t应该在的位置（这个就对两两对应的点坐标进行线性插值即可，可以先预处理），然后通过算出点P对于每个三角形的重心坐标来判断点P是在哪个三角形中。对于一个三角形ΔABC来说，P的重心坐标$[\lambda_1 \quad \lambda_2 \quad \lambda_3]^{\mathrm{T}}$计算方法通过解如下线性方程组来获得：

$$\begin{bmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

   然后我们就可以得到点P在$t = 0$时的坐标点P_0与$t = 1$时的坐标点P_1，这只需要把上式左边的矩阵中的三角形ΔABC参数换成$t = 0$时与$t = 1$时的对应参数即可。那么我们对灰度的线性插值也应运而生：

$$\mathbf{A}_P(t) = (1-t)\mathbf{A}_{P_0}(0) + t\mathbf{A}_{P_1}(1)$$

   把t在$[0,1]$区间中均匀采样，即可得到一系列连续变化的图像。

4. 得到 GIF

   这里可以用许多工具，如 ffmpeg、ImageMagick 等。本人用 ffmpeg 来实现。在 Morphing 文件夹里面，有 togif.bat 批处理来将一系列的图片转换成 gif 动画。

四. 算法代码（CImg）实现

见压缩包下代码。

五. 实验结果

见工程文件下的 Dataset/编号/Morphing 中的结果。

六. 分析与评价

需要指出来的是，这一套模型对标点的正确性与完备性要求是很高的。例如下面这一对

图像：

		
开始图像	结束图像	$t = 0.5$时的图像

我们可以看到，由于 Face++的标关键点只对人的五官进行标注，眼镜并未标注，所以中间图像看上来就有两副眼镜；同理，脖子的形变也不自然，究其原因也是关键点的标注不到位。

另外，Release 选项真的比 Debug 选项快了许多许多。