# 人脸美妆大作业最终报告

时间：2018 年 6 月 15 日

组长：常书豪 2015011064

组员：林伯昱 2015011082

组员：马小军 2015011112

# 1 研究背景

## 1.1 人脸关键点检测

随着现代科技的快速发展，生物特征被应用到身份认证上来。现代生物识别技术通过计算机科学技术利用人体固有的生理特征进行身份验证。其中人脸是身份验证的重要途径之一。

人脸包含丰富的信息，是人类识别和验证的主要标志之一，也是图像和视频中感兴趣的部分。与人体其他生物特征相比，人脸识别更容易实现和应用。在视频会议、档案管理、电子相册、身份识别、基于对象的图像等方面有广泛应用，是当前模式识别和人工智能领域的研究热点。人脸识别的研究始于 20 世纪 60 年代，20 世纪 90 年代以来，由于处理器和 GPU 制造技术的发展，一些商业性的人脸识别系统逐渐进入市场，应用于电子商务、人机交互、视频监控、信息安全等领域。

人脸识别技术融合数字图像处理、计算机图形学、计算机视觉、模式识别、神经网络、生理学、心理学等多个学科的理论。人脸关键点检测又称脸部关键点定位。在机器学习的领域里，人脸关键点检测技术有着其独特的应用价值，因为该技术 是众多脸部图片处理的先序步骤，这些应用包括但不限于人脸识别，脸部表情分析，脸部变形动画。

## 1.2 眉毛

眼睛是心灵的窗户，那么我们可以把眉毛看成是窗帘；眼睛是人生的一幅画，那眉毛就是画框。长在眼睛上方的眉毛，在面部占有重要的位置，具有美容和表情作用，能丰富人的面部表情，双眉的舒展、收拢、扬起、下垂可反映出人的喜、怒、哀、乐等复杂的内心活动。在中国文学里，有很多形容眉毛的，如：扬眉剑出鞘、眉飞色舞、剑眉入鬓、蛾眉谈扫、眉头紧锁、喜上眉梢、柳叶弯眉、眉目传情……

眉形即眉毛的形状。大约分为弦月眉、一字眉、三角眉等。眉形从某个角度来说反应了一个人的性格态度。一对漂亮的眉形，就像名家设计的头发一样，对形象有绝对的加分效果，不同眉形给人留下的印象也不同。

方法描述：

## 1.3 文献综述

在图像处理领域，对目标图像的定位是热点和重点课题。在文献[1]中，主要工作在两方面：在人脸检测方面，在 YCbCr 颜色空间建立肤色模型，对常见肤色分析后高斯建模；在五官定位方面，在得到的肤色区域内，分别对似然图和灰度图进行阈值分割处理。两种算法结合起来，对图像质量要求不高，有广泛实用性。

在文献[2]中提出了一种新颖的眉轮廓和下巴轮廓提取方法。我们首先使用空间受限的

子区域 k-均值聚类来划分粗眉区域。然后用蛇的方法提取眉轮廓，有效的图像力。对于下巴轮廓的提取，我们首先估计了几个可能的下巴位置，这些位置被用来建立一系列的曲线作为下巴轮廓的候选人。摘要基于下巴边缘探测器提取的下巴边缘，选择了最大似是实际的下巴轮廓曲线。最后，可信提取的眉轮廓和估计的下巴轮廓被用作面部识别的几何特征。实验结果表明，所提出的算法能够很好地提取出眉轮廓和下巴轮廓，提取的特征对提高人脸识别率是有效的。

在文献[3]中，在求解泊松方程的基础上，采用通用插值机制，对图像区域进行无缝编辑，提出了各种新颖的工具。第一组工具允许将不透明和透明的源图像区域无缝地导入到目标区域。第二组基于相似的数学思想，允许用户在选定的区域内无缝地修改图像的外观。这些变化可以被安排来影响区域内物体的质地、照明和颜色。

本研究通过对输入图形做特征点检测，检测完成之后在原图基础上做区域检测，画出眉毛区域，采用图像处理的方法在原图的基础上对眉毛进行修理，消除边界。

[1]李彦君.人脸五官定位算法研究[D].中北大学，信号与信息处理，硕士
[2]Chen, Q., Cham, W., Lee, K., Extracting eyebrow contour and chin contour for face recognition[J], Elsevier Science Inc. New York, NY, USA, Volume 40 Issue 8: 2292-2300
[3] Patrick Pérez, Gangnet M, Blake A. Poisson image editing[J]. Acm Transactions on Graphics, 2003, 22(3):313-318.

# 2 算法原理及实现

## 2.1 眉毛轮廓识别

在对眉毛轮廓的提取这一步我们主要提出两个算法。算法一是使用 opencv 库里面的 Canny 算法做轮廓检测。Canny 边缘检测算法可以分为以下 5 个步骤：

- 应用高斯滤波来平滑图像，去除噪声
- 找寻图像的强度梯度
- 应用非最大抑制技术来消除边误检
- 应用双阈值的方法来决定可能的边界
- 利用滞后技术来跟踪边界

由于关键点的位置在轮廓的上方，所以只需应用 Canny 算法找到下方的轮廓便可。其优点在于精确找出轮廓的边缘，边缘的线条很自然，但是其缺点比较严重，第一是由于曲线不闭合，所以要做复杂的闭合处理，第二是眉毛内部有许多细小的线条，有些可能是眉毛的纹理，所以在找边缘时比较难找，第三是该算法对光线要求比较高，光线略微差一点就检测出错。

算法二是基于关键点的轮廓划分，由于关键点的顺序已知，只需找到眉毛中心的三个点，

向下平移逐个像素查找，得到下方三个 01 灰度变化的点，然后用直线或者贝塞尔曲线将这些点连接起来就成为眉毛的轮廓。该方法的优点是简单，复杂度低，对光线要求低，但是缺点是显然的，每个人的眉毛粗细不同，所以固定的向下平移距离鲁棒性并不强，而且由于用固定的曲线圈边界，有可能出现多圈或少圈的问题，而且自然性也不高。

在实际测试中我们采用第二种算法，即基于 68 点的关键点检测，可以确定眉毛的上边缘 10 个点（各 5 个）。基于亚洲人眉形的规律和后期处理算法，不需要特别精确的边缘定位，可以通过 5 个点的横坐标的距离确定眉毛下边缘 3 个点，然后用直线将关键点连接起来。主要代码如下：

```
if (!shapes.empty() && offset != 0) {
        //左眼区域
        cv::line(temp, cvPoint(shapes[0].part(17).x(), shapes[0].part(17).y()),
cvPoint(shapes[0].part(18).x(), shapes[0].part(18).y()), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(18).x(), shapes[0].part(18).y()),
cvPoint(shapes[0].part(19).x(), shapes[0].part(19).y()), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(17).x(), shapes[0].part(17).y()),
cvPoint(shapes[0].part(18).x(), y[0]), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(18).x(), y[0]),
cvPoint(shapes[0].part(19).x(), y[1]), cv::Scalar(0, 0, 255));
        //右眼区域
        cv::line(temp, cvPoint(shapes[0].part(25).x(), shapes[0].part(25).y()),
cvPoint(shapes[0].part(26).x(), shapes[0].part(26).y()), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(24).x(), shapes[0].part(24).y()),
cvPoint(shapes[0].part(25).x(), shapes[0].part(25).y()), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(26).x(), shapes[0].part(26).y()),
cvPoint(shapes[0].part(25).x(), y[5]), cv::Scalar(0, 0, 255));
        cv::line(temp, cvPoint(shapes[0].part(25).x(), y[5]),
cvPoint(shapes[0].part(24).x(), y[4]), cv::Scalar(0, 0, 255));

    //左眼以37为界，右眼以44为界
    int leftlow = (shapes[0].part(37).y() + shapes[0].part(17).y()) / 2;
    int rightlow = (shapes[0].part(44).y() + shapes[0].part(26).y()) / 2;
    temp.at<cv::Vec3b>(leftlow, shapes[0].part(37).x()) = cv::Vec3b(0, 0, 255);
    temp.at<cv::Vec3b>(rightlow, shapes[0].part(37).x()) = cv::Vec3b(0, 0, 255);
```

## 2.2 眉毛变形

眉的形式有：云纹眉、蝶翅眉、柳叶眉、蝠形眉、螳螂眉、鸳鸯眉、花眉、直眉、环眉、刁眉、方眉、尖眉、点眉、鸭蛋眉、棒槌眉、葫芦眉、火焰眉、寿字眉等。唐明皇令画工画的十眉图名称为：鸳鸯眉、远山眉、五岳眉、三峰眉、垂珠眉、却月眉、分梢眉、涵烟眉、拂云眉、倒晕眉、云锦眉（云形眉）等。

为了修改眉毛的形状，我们采用的是切割下眉毛，再将眉毛修改形状之后贴上去，原本眉毛的位置用眉毛周围的像素贴上去。具体要分以下两步走：

【修改眉毛形状】　　以眉毛的中点为中心，将边缘的眉毛像素线性移位，为了保持连续性，在中点处的位移为 0，边缘处为最大值。主要代码如下：

```
for (int i = 2; i<xlen; i++)
{
    int right_offset = double(i*offset / xlen);
    for (int j = 0; num != 2 && j<50; j++) {
        if (num == 0 && temp.at < cv::Vec3b >(rightlow - j - 1,
shapes[0].part(24).x() + i) == cv::Vec3b(0, 0, 255))
        {
            num = 1;
            imo.at< cv::Vec3b >(rightlow - j - 1 - right_offset,
shapes[0].part(24).x() + i) = im1.at < cv::Vec3b >(rightlow - j - 1,
shapes[0].part(24).x() + i);
            continue;
        }
        if (num == 1 && temp.at < cv::Vec3b >(rightlow - j - 1,
shapes[0].part(24).x() + i) == cv::Vec3b(0, 0, 255))
        {
            num = 2;
        }
        if (num == 1) {
            imo.at< cv::Vec3b >(rightlow - j - 1 - right_offset,
shapes[0].part(24).x() + i) = im1.at < cv::Vec3b >(rightlow - j - 1,
shapes[0].part(24).x() + i);
        }
}
```

【填充原本区域】　　为了避免像素选取的偶然性，取眉毛周围像素的平均值填充。主要代码如下：

```
int xlen = shapes[0].part(19).x() - shapes[0].part(17).x();
    //cout << xlen << endl;
    for (int i = 2; i<xlen; i++) {
        for (int j = 0; num != 2 && j<50; j++) {
```

```
            if (num == 0 && temp.at < cv::Vec3b >(leftlow - j - 1,
shapes[0].part(17).x() + i) == cv::Vec3b(0, 0, 255))
            {
                num = 1;
                imo.at < cv::Vec3b >(leftlow - j - 2, shapes[0].part(17).x() + i) =
im1.at < cv::Vec3b >(leftlow, shapes[0].part(17).x() + i);
                continue;
            }
            if (num == 1 && temp.at < cv::Vec3b >(leftlow - j - 1,
shapes[0].part(17).x() + i) == cv::Vec3b(0, 0, 255))
            {
                num = 2;
            }
            if (num == 1) {
                imo.at < cv::Vec3b >(leftlow - j - 1, shapes[0].part(17).x() + i) =
im1.at < cv::Vec3b >(leftlow, shapes[0].part(17).x() + i);
                imo.at < cv::Vec3b >(leftlow - j - 2, shapes[0].part(17).x() + i) =
im1.at < cv::Vec3b >(leftlow, shapes[0].part(17).x() + i);
                imo.at < cv::Vec3b >(leftlow - j - 3, shapes[0].part(17).x() + i) =
im1.at < cv::Vec3b >(leftlow, shapes[0].part(17).x() + i);
                imo.at < cv::Vec3b >(leftlow - j - 4, shapes[0].part(17).x() + i) =
im1.at < cv::Vec3b >(leftlow, shapes[0].part(17).x() + i);
            }
    }
```

## 2.3 眉毛变色

分为两种方案，使用函数重载的方式实现。

第一种是直接使用 RGB 的调色，输入三个颜色，直接将眉毛区域涂上该颜色。此种方法的优点是方便调色，想要什么色调就直接用什么色调，但缺点是会直接覆盖原本的眉毛纹路，显得比较不自然。所以再进行处理，将原本的纹理和欲改动的颜色迭加，即可达到效果。

第二种是使用 HSL 色系的调色，此方法直接调整色调(hue)和饱和度(saturation)即可，可以保留原本眉毛的纹路。写了一个函数，实现从 RGB 变换成 HSL 色系，修改色调和饱和度之后，再将其变回 RGB 的过程。

尔后，只需要将左右眉毛的八个点，分别进入 draw_eyebrow 函数中，就可以将眉毛覆盖成指定的颜色了。由于眉毛八个点近似于眉毛的轮廓，所以就分别将点联机围成的区域视为眉毛内部，直接将其调色即可。由于对于整张图进行循环会太冗余，所以找到眉毛八个点的极值作为 for 循环的边界，在该范围内进行判断即可，我使用了八个判别式来确定点是否是眉毛区域。然后调用上述的 RGB 着色或是 HSL 着色。主要代码如下：

```
void draw_eyebrow(cv::Mat temp, int eb[2][8], double hue, double saturation) {
    for (int y = *std::min_element(eb[1], eb[1] + 8); y < *std::max_element(eb[1],
eb[1] + 8); y++) {
        for (int x = eb[0][0]; x < eb[0][4]; x++) {
            // if the pixel in the region, change its color
            if (double(x - eb[0][0]) * (eb[1][1] - eb[1][0]) <= double(y -
eb[1][0]) * (eb[0][1] - eb[0][0]) &&
                double(x - eb[0][1]) * (eb[1][2] - eb[1][1]) <= double(y -
eb[1][1]) * (eb[0][2] - eb[0][1]) &&
                double(x - eb[0][2]) * (eb[1][3] - eb[1][2]) <= double(y -
eb[1][2]) * (eb[0][3] - eb[0][2]) &&
                double(x - eb[0][3]) * (eb[1][4] - eb[1][3]) <= double(y -
eb[1][3]) * (eb[0][4] - eb[0][3]) && !(
                    double(x - eb[0][5]) * (eb[1][6] - eb[1][5]) <= double(y -
eb[1][5]) * (eb[0][6] - eb[0][5]) &&
                    double(x - eb[0][6]) * (eb[1][7] - eb[1][6]) <= double(y -
eb[1][6]) * (eb[0][7] - eb[0][6])) && !(
                        double(x - eb[0][0]) * (eb[1][5] - eb[1][0]) <= double(y -
eb[1][0]) * (eb[0][5] - eb[0][0]) &&
                        double(x - eb[0][7]) * (eb[1][4] - eb[1][7]) <= double(y -
eb[1][7]) * (eb[0][4] - eb[0][7]))
                ) {
                cv::Vec3b color = temp.at<cv::Vec3b>(y, x); //original color
                double my[3];
                GBR2HSL2GBR(color[0], color[1], color[2], hue, saturation, my);
//change its hue
                temp.at<cv::Vec3b>(y, x) = cv::Vec3b(int(my[0]), int(my[1]),
int(my[2])); //put back the pixel
            }
        }
    }
}
```

## 2.4 图形界面

本实验的 GUI 图形界面是基于内嵌在 vs2017 内的 Qt 平台来完成的，总体的思路是对每个界面上的按钮进行信号槽设置，在 mainwindow.h 文件中声明各个函数体，在 mainwindow.cpp 文件中书写具体的函数实现。同时，对于需要调用的函数，都集中放在 tool.cpp 文件中实现，并在 tool.h 头文件中声明。具体代码可直接到工程文件中查找，这里不再赘述。

# 3 成果展示

## 3.1 眉毛变形

本眉毛变形算法可以满足眉毛的上扬和上弯的要求，变形的尺度通过像素定义，为了看起来自然，将像素设定在正负 20 之间，但是更大一些的也可以。正的为上扬，负的为下弯。从图中可以看到初步满足眉毛变形的要求。

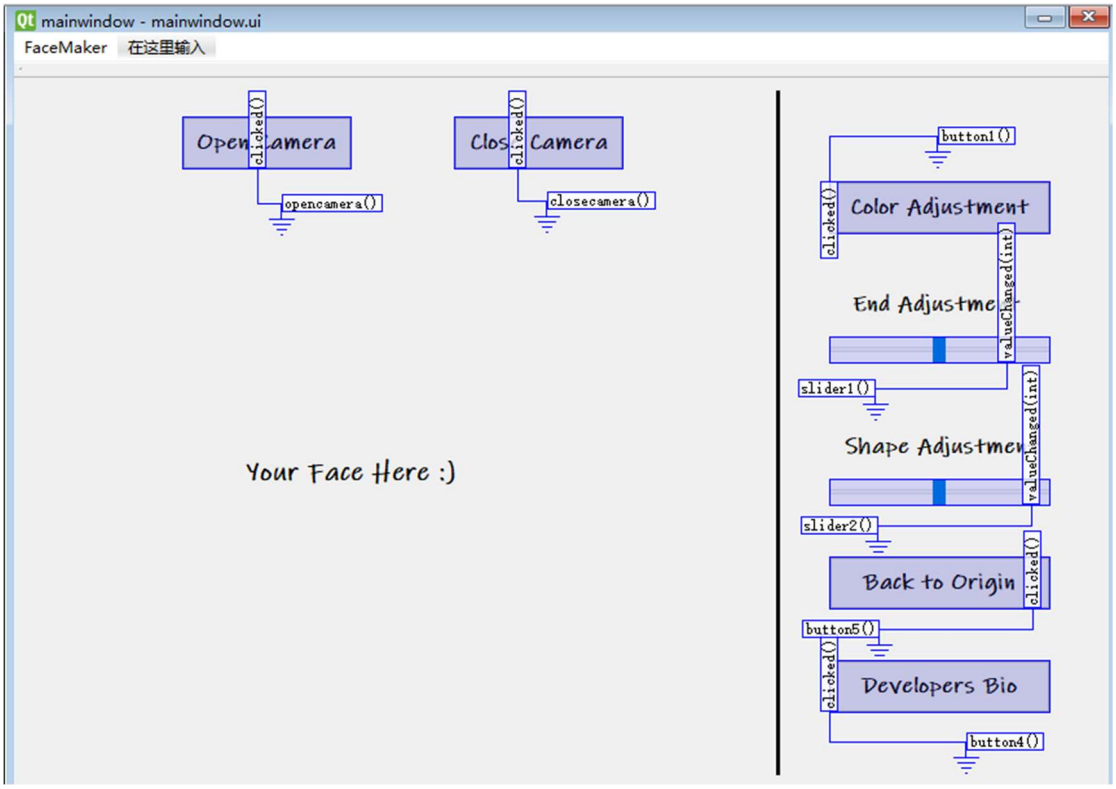
offset=10　　offset=-10

## 3.2 眉毛变色

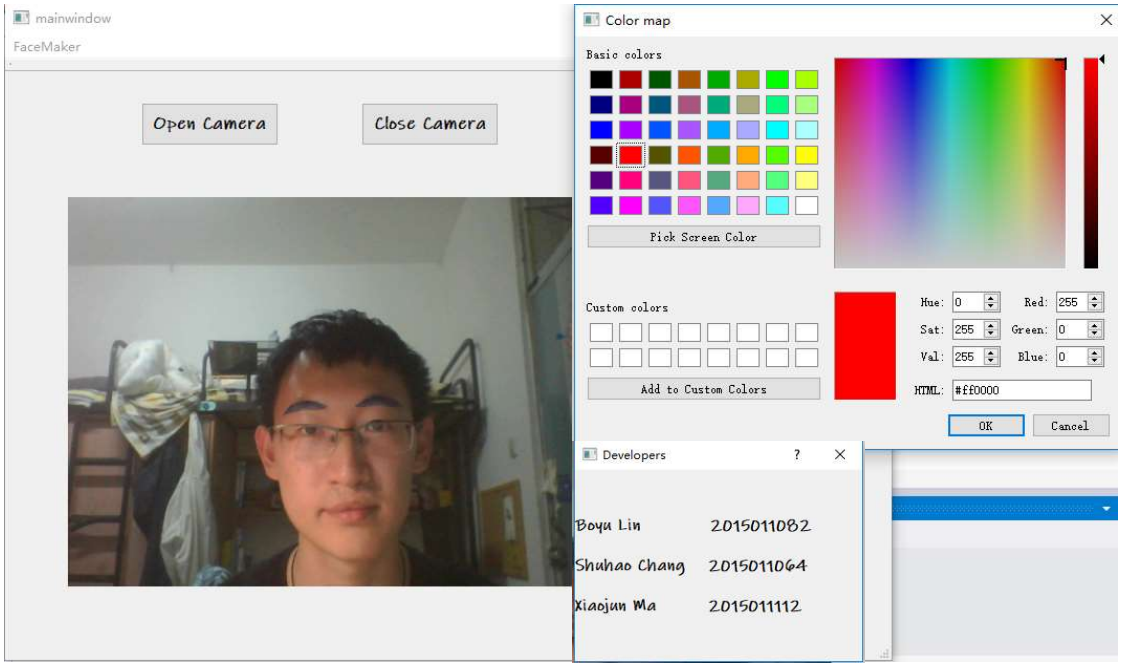
本眉毛变色算法能够实现实时自定义变色，并且对头部摆动具有一定的鲁棒性，具体效果如下图所示:

## 3.3 图形界面

图形界面底层信号槽接口如下。从图中可见，该图形界面支持调用调色板进行颜色选择，支持使用滚动条调节眉毛末梢偏置量，支持打开、关闭摄像头操作，支持开发者简介。




图形界面实际应用窗口样式如下：



# 4 思考讨论

（1）在眉形改变方面，本次实验对人脸美妆算法有了初步的学习，修改的眉形也满足了初步的设想，但是依然存在一些局限性和不足。首先，由于算法是从中点向两边线性移动，所以在位移较大的时候不够自然，斜率很大。其次，由于填充像素只填充了眉毛原本的区域，边缘临近的黑像素会使边缘不连续。

（2）在眉毛颜色改变方面，RGB 调色的部分应该可以再进行处理的，但因为时间关系并没有再深入研究，并且用 HSL 调色函数就可以实现调色同时保留眉毛纹路的功能，所以就只使用 HSL 函数。在 OpenCV 中好像有 cvtColor 的函数可以将 RGB 改成 HSL 调整数值再改回 RGB 的功能，但由于不明原因一直报错，所以只好自己写一个转换函数，还好并不复杂。但发现在暖色调的时候，如橘色、黄色等，和皮肤颜色相近的时候，变色会很不明显，和周围皮肤相近，可以再调整效果。

（3）曾经有思考用 Bezier 曲线作为眉毛的边界点，但发觉计算量较大而且不好写判别式，加上直接只用八个点似乎没有太大的差别，所以就直接使用直线取的区域了。原本直线判别式是使用两点式，$(y-y_2)/(y_1-y_2) <= (x-x_2)/(x_1-x_2)$，但会因为头部摆动导致斜率有时正有时负，此时判别式的大于小于就会相反，比较麻烦。并且在分母为 0 的时候，无法正常绘制眉毛。后来将其调整成$(y-y_2)*(x_1-x_2) <= (x-x_2)*(y_1-y_2)$，不但分母为 0 的状况解决了，判别式乘-1 的问题也不用考虑了，一次解决两个问题。

（4）比较可惜的是，后来并没有完美解决眉毛变形的问题，眉毛剪切还有些不完美，而且跟助教提到的眉毛粗细调整的部分，也没调出来。原本是想用眉毛附近包含皮肤的区域，透过图像处理的手段，来达到粗细的变化。若要眉毛变粗的话，就将皮肤部分降采样，而眉毛的部分使用线性插值增加像素点，使得总像素点不变就可以成功使得眉毛变粗。而眉毛变细也是同理，将眉毛部分的像素点降采样，而皮肤的部分用线性插值增加像素点，就可以达到眉毛变细的效果。

# 5 组内分工

常书豪：眉毛轮廓定位算法、制作图形界面、共同 Debug

马小军：眉毛形状改变算法、共同 Debug

林伯昱：眉毛颜色改变算法、共同 Debug