

计算机视觉附加题1实验报告

实验环境

minGW32 g++

编译命令：

```
g++ main.cpp -lgdi32
```

实验内容

Part1

用CImg编写灰度图像直方图均衡化：

- (a) 用手机拍摄不同光照情况下的图像作为测试样本，不低于5张
- (b) 分别针对灰度图像和彩色图像分别用直方图均衡化方法完成结果
- (c) 对实验结果进行分析，特别是彩色图像直接采用直方图均衡化效果如何，如果要改进从哪些方面进行改进

Part2

根据PPT和参考文献实现颜色转换

- (a) 测试图像集合不低于5对图像（参考图和原图）
- (b) 对实验结果以及存在问题进行分析，给出初步的改进建议和参考文献

实验过程

Part 1

灰度图片的直方图均衡化

定义和实现在 `HistogramEqualizationForGray` 这个类中：

将彩色图像转化为灰度图像

`toGrayImg()` - 根据公式：

```
cimg_forXY(img, x, y) {  
    grayImg(x,y) = img(x,y,0,0) * 0.299 + img(x,y,0,1) * 0.587 + img(x,y,0,2)*0.114;  
}
```

统计灰度图像上每个灰度值出现的频率

`countForProbabilities` - 声明一个长度为256的double类型数组 `probabilities`，遍历图像，将每个像素灰度值做为下表，其对应的数组元素+1。最后每个数组元素除以图像像素点总数（长乘宽）

计算目标数值

对于 $i = 0 \sim 256$ 每个值，将`probabilities`在当前 i 之前的所有元素累加，并乘以255然后取整，存入长度为256的 `targetGrayvalue` 数组。

转化为目标图片

对于原图片中每个像素，将目标图片的该像素赋值为以原图片该点灰度值做为下标在 `targetGrayvalue` 中的元素值。

彩色图片在RGB通道上的直方图均衡化

与灰度图像原理相同，不需要转化为灰度图像的步骤，且其中的每一步都需要在RGB三个通道上做一遍。

Part 2

将RGB色彩空间转化到Lab色彩空间

根据论文中的矩阵乘法公式，得到下面的C++代码：

```
CImg<double> RGBtoLab(const CImg<unsigned char>& inputImg) {
    CImg<double> LMS = CImg<double>(inputImg.width(), inputImg.height(), 1, 3);
    cimg_forXY(inputImg, x, y) {
        LMS(x,y,0,0) = 0.3811 * inputImg(x,y,0,0) + 0.5783 * inputImg(x,y,0,1) + 0.0402
        * inputImg(x,y,0,2);
        LMS(x,y,0,1) = 0.1967 * inputImg(x,y,0,0) + 0.7244 * inputImg(x,y,0,1) + 0.0782
        * inputImg(x,y,0,2);
        LMS(x,y,0,2) = 0.0241 * inputImg(x,y,0,0) + 0.1228 * inputImg(x,y,0,1) + 0.8444
        * inputImg(x,y,0,2);
        if(LMS(x,y,0,0) == 0) LMS(x,y,0,0) = 1;
        if(LMS(x,y,0,1) == 0) LMS(x,y,0,1) = 1;
        if(LMS(x,y,0,2) == 0) LMS(x,y,0,2) = 1;
        LMS(x,y,0,0) = log10(LMS(x,y,0,0));
        LMS(x,y,0,1) = log10(LMS(x,y,0,1));
        LMS(x,y,0,2) = log10(LMS(x,y,0,2));
    }

    CImg<double> lab = CImg<double>(inputImg.width(), inputImg.height(), 1, 3);
    cimg_forXY(LMS, x, y) {
        lab(x,y,0,0) = (double)1/sqrt(3) * (LMS(x,y,0,0) + LMS(x,y,0,1) + LMS(x,y,0,2));
        lab(x,y,0,1) = (double)1/sqrt(6) * (LMS(x,y,0,0) + LMS(x,y,0,1) + (-2) *
        LMS(x,y,0,2));
        lab(x,y,0,2) = (double)1/sqrt(2) * (LMS(x,y,0,0) + (-1)*LMS(x,y,0,1));
    }
    return lab;
}
```

需要注意取对数要避免真数为0的情况，否则会在第一步色彩空间转化就不能得到正确的图片。

计算待处理图片和参考图片中每个像素的l, a, b三个通道的均值和标准差

遍历两幅图片，在每幅图片的每个通道加上该通道的数值，遍历完毕后每个通道的数值除以该图片上像素点的总数作为两幅图片共六个通道的均值。

标准差的计算方法类似。

处理待处理图片

将待处理图片均值变为0

通过每个像素的三个通道都减去该通道的平均值实现。

将待处理的图片的方差变为与参考图片相同

通过将待处理图片的每个像素点的数值先除以待处理图片的标准差再乘以参考图片的标准差决定。

将待处理图片的均值变为原图片的均值

通过每个像素的三个通道都加上参考图片中对应通道的平均值实现。

将Lab色彩空间转换回RGB色彩空间

同样使用论文中的矩阵乘法。

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

LMS矩阵得到后需要将矩阵的值作为指数，10作为底数进行一步转化获得LMS。

这里需要注意在赋值给RGB图像时，如果色彩范围超出0~255需要归为0~255。

实验结果

Part 1

测试样例1



原图



RGB 空间直方图均衡化

- 这张照片中由于是夜晚拍摄本身颜色并不丰富，在RGB进行均衡化之后丰富了色彩信息。

测试样例2



原图



RGB 空间直方图均衡化

- 这幅图片在RGB通道进行直方图均衡化之后效果正常的原因是晚霞的亮部变亮而安布变得更暗，照片中其他部分则因为饱和度较低，所以没有过于明显的颜色变化。

测试样例3



原图



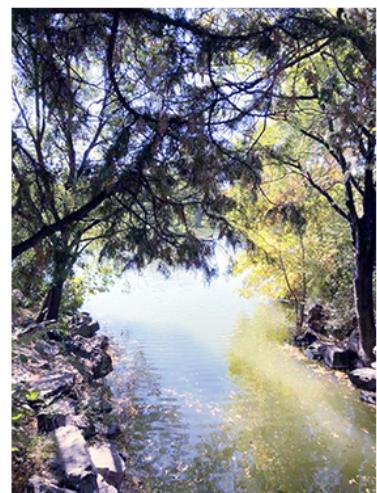
RGB 空间直方图均衡化

- RGB直方图均衡化之后使得原来子在图片中较为明显的黄色调 (R+G) 变淡，明显看出RGB空间直方图均衡化之后画面偏蓝。

测试样例4



原图



RGB 空间直方图均衡化

- 这幅图RGB空间的直方图均衡化问题在于由于G通道被均衡化，使得整体图片中原本浓郁的绿色系发白。

测试样例5



原图



RGB 空间直方图均衡化

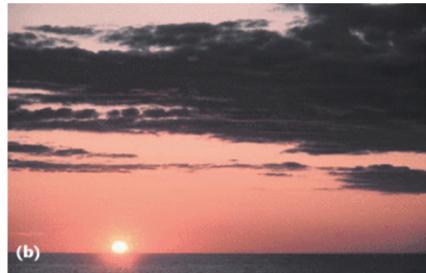
- 这个光线下直方图均衡化会使白的地方更白，通过查看文件夹的大图可以发现，原图的人脸五官是清晰的，但由于人脸是图片中颜色较亮的地方，在RGB通道上进行直方图均衡化后会出现五官过度曝光导致模糊的情况。

Part 2

测试样例1



待处理图片



参考图片



合成图片

- 该样例为论文中样例，效果较好。

测试样例2



待处理图片



参考图片



合成图片

- 在测试这一样例的时候发现了出现LMS空间取log是遇到真数为0时会导致lab空间发生错误的Bug。总体效果正常

测试样例3



待处理图片



参考图片



合成图片

- 使用该样例的目的在于测试在图像色彩不够丰富时该算法能否正常使用。

测试样例4



待处理图片



参考图片



合成图片

- 与样例3目的一致。由于待处理图片明亮部分的亮度没有参考图片高，而且面积较小，所以处理过后图片变暗。

测试样例5



待处理图片

参考图片

合成图片

- 测试3和4用于测试光线较暗的情况下，即黑色占据面积较大的情况下。样例5主要用于当参考图片和待处理图片色彩较为单一时的情况。会发现花茎的细节地方由于原本的绿色必须被参考图片中的接近晚霞的亮黄色取代，导致了颜色过于跳跃的情况。

测试样例6



待处理图片

参考图片

合成图片

- 选取这组图片的原因在于测试完全黑白的图片作为待处理图片时加以颜色丰富的参考图片时的效果。但最终结果显示只是大体色调偏向参考图片中面积最大的颜色。

实验改进

Step1

针对Part1中彩色图像在RGB通道中分别和在各个颜色通道中做相同直方图均衡化，再将三个通道叠加在一起并不能取得很好的效果。由于RGB通道相互影响，会使得图像的主色调发生较大改变，发生整体泛白的现象。比如图像偏黄，但均衡化之后图像变蓝（参考测试样例3）。



原图



RGB 空间直方图均衡化

考虑到RGB通道只是作为三个颜色通道，都进行均衡化必然会使图像色调发生转变，因而需要一个不影响色调的方法。我尝试了先将图像从RGB空间转换到HSL空间，因而其中只有H通道代表色调，而S通道代表饱和度，L通道代表亮度，尝试在后两个通道上分别或者同时做均衡化然后分析结果。但由于HSL的取值范围并不在0~255范围之内且如果强行映射到0~255范围，会导致由于取整的精度丢失问题（这部分在代码中进行了实现，但最终效果不尽人意）。

Step2

后来尝试了另一个将亮度和色调分离开来的色彩空间YCbCr，发现对其亮度通道Y进行直方图均衡化之后可以达到较好效果。

这是因为Cb和Cr通道代表的是色彩，而Y通道作为亮度是和两个色彩通道正交（即Y的改变不会使图像本身的色相发生很大的变化）。

改进结果如下：

测试样例1



原图



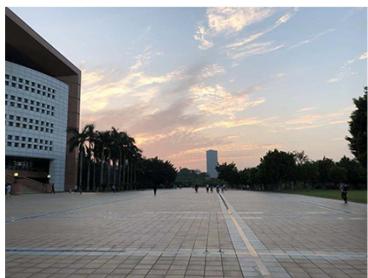
RGB 通道



YCbCr 通道

- YCbCr通道上的均衡化保留了原图色彩较为单一的特征，不想RGB通道中过度丰富了颜色。

测试样例2



原图



RGB 通道



YCbCr 通道

- 主观上说，这幅图片YCbCr通道的表现并没有RGB优秀，但是相比于原图的确增加了图片的对比度，但是在色相上没有像RGB那样丰富。（原因是这张图的颜色的丰富并不会产生失真的效果，所以还可以接受）

测试样例3



原图



RGB 通道



YCbCr 通道

- 这张图就是在RGB空间明显色调发生变化的图片，在YCbCr通道上明显可以发现其整体的黄色调保留了下来（可能由于原图在拍摄的时候用了滤镜，并没有很明显的变化）。

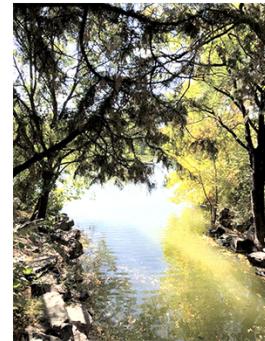
测试样例4



原图



RGB 通道



YCbCr 通道

- 这张图由于原图是在树荫下拍摄，光线较弱，因此无论是RGB还是YCbCr通道或多或少都增加了整体亮度，但是仍然是后者对于颜色的整体信息更符合原图的特征。

测试样例5



原图

RGB 通道

YCbCr 通道

- 由于建筑在拍摄时灯光为红色，整体图片色调中R和B通道数值较高，而G通道数值较小，但进行均衡化之后，让像素点在G上也大范围分布，即出现一些G值较高的点，集中在亮度高的部分，比如建筑外墙的装饰钢筋上（RGB通道上的均衡化发绿）。而YCbCr却保留原色，可是上文提到的亮部的细节变少的问题仍然没有得到很好的解决。

实验思考

在原图和参考图色彩均较为鲜明的情况下，色彩迁移效果较好，即使遇到光线较暗或者色彩相对单一的地方也仍然能够保持较好效果。但是当图像颜色变为黑白时，该算法不再适用。

针对与Part2中将彩色图像映射到几乎为灰色的图像上效果不好的问题，查找到了这篇论文在RGB空间中将彩色图片的颜色映射到黑白图片上的方法。

论文连接：<https://ieeexplore.ieee.org/abstract/document/4579871>

这里使用人工为黑白图片找寻参考色并生成调色板，为图像灰度分块，形成灰度图像每一块和调色板上一部分的映射关系，为灰度图像染色。

此外由于这样直接改变每个像素的值，如果两者的标准差相差过多，会导致原本连续平滑的待处理图像由于颜色上的巨大变化产生跳跃，出现锯齿色块。这个问题可以在映射的过程中加入插值进行修改，或者使用高斯模糊将跳跃弱化。