数字图像处理第二次作业

14331074 龚俊宁

1 Exercises

1.1 Linearity

直方图均衡化是线性变换。要证明一个函数是线性变换，则需要证明其加法封闭和数乘封闭。

a. 加法封闭

$$
\begin{aligned}
s_k &= T(r_k) + T(r'_k) \\
&= (\mathrm{L}-1)\sum_{\mathrm{j}=0}^{\mathrm{k}} P(r_j) + (\mathrm{L}-1)\sum_{\mathrm{j}=0}^{\mathrm{k}} P(r'_j) \\
&= \frac{L-1}{MN}\sum_{j=0}^{k} n_j + \frac{L-1}{MN}\sum_{j=0}^{k} n'_j \\
&= \frac{L-1}{MN}\sum_{j=0}^{k} (n_j + n'_j) \\
&= T(r_k + r'_k)
\end{aligned}
$$

b. 数乘封闭

$$
\begin{aligned}
\mathrm{s_k} &= \mathrm{c} * \mathrm{T}(\mathrm{r_k})(\mathrm{c} \neq 0) \\
&= \mathrm{c} * (\mathrm{L}-1)\sum_{\mathrm{j}=0}^{\mathrm{k}} P(r_j) \\
&= \mathrm{c} * \frac{L-1}{MN}\sum_{j=0}^{k} n_j \\
&= \frac{L-1}{MN}\sum_{j=0}^{k} cn_j \\
&= \mathrm{T}(cr_k)
\end{aligned}
$$

综上所述，直方图均衡化满足加法封闭和数乘封闭，所以是线性变换。

1.2 Spatial Filtering

1. 计算卷积

$$w:\begin{bmatrix}1 & 2 & 2 & 1\\ 2 & 3 & 3 & 2\\ 2 & 3 & 3 & 2\\ 1 & 2 & 2 & 1\end{bmatrix} \qquad f:16\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

zero-padding:在 w 的上下左右行列分别插入 3-1 = 2 列

$$w':\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 2 & 2 & 1 & 0 & 0\\ 0 & 0 & 2 & 3 & 3 & 2 & 0 & 0\\ 0 & 0 & 2 & 3 & 3 & 2 & 0 & 0\\ 0 & 0 & 1 & 2 & 2 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

将过滤器旋转后，得到

$$f':16\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

移动算子，利用公式

$$\mathrm{w(x,y)}\vartriangle\mathrm{f(x,y)} = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t)$$

计算后得到卷积

$$result:1/16\begin{bmatrix}15 & 25 & 25 & 15\\ 25 & 40 & 40 & 25\\ 25 & 40 & 40 & 25\\ 15 & 25 & 25 & 15\end{bmatrix}$$

2. 再次利用第 1 问的结果与 *f* 进行卷积计算

$$result':1/256\begin{bmatrix}200 & 325 & 325 & 200\\ 325 & 525 & 525 & 325\\ 325 & 525 & 525 & 325\\ 200 & 325 & 325 & 200\end{bmatrix}$$

从 *result'*中可以看出，像素之间的关系其实并没有改

变，改变的只是灰度值，所以其实重复使用同一滤波器进行滤波，效果并没有越来越好。可以说其实差不多。

3. **相关和卷积的不同之处**

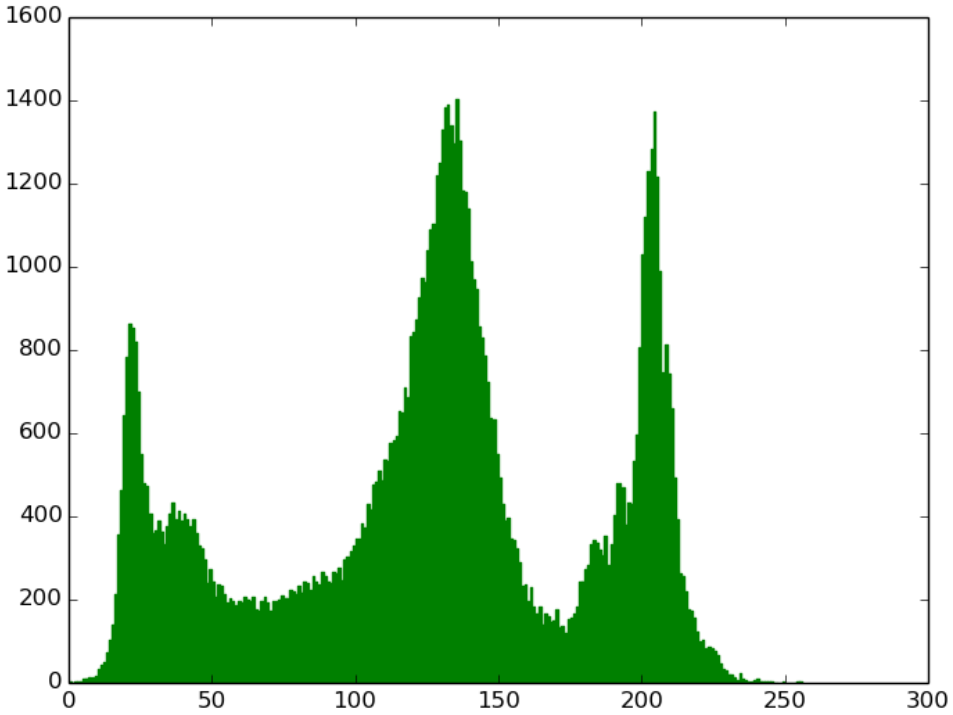
   a. **概念：**相关是滤波器模板移过图像并计算每个位置乘积之和的处理。卷积的机理相似，但滤波器首先要旋转180°。

   b. **是否满足交换律：**从计算公式看，卷积满足交换律而相关则并不满足。

2 Programming Task

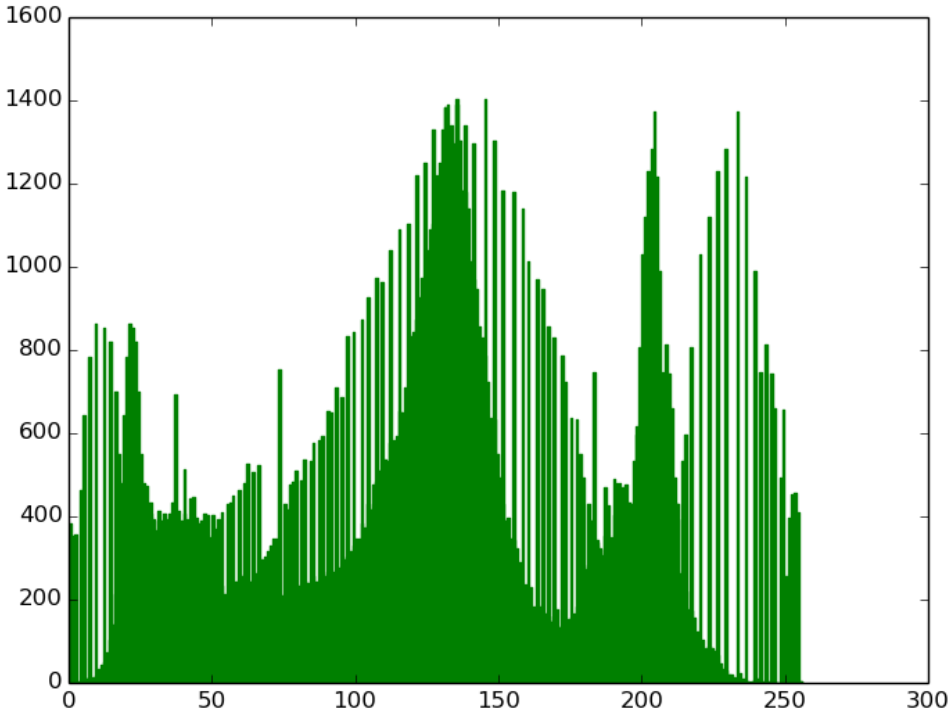
2.1 Histogram Equalization

1. the histogram of the input image



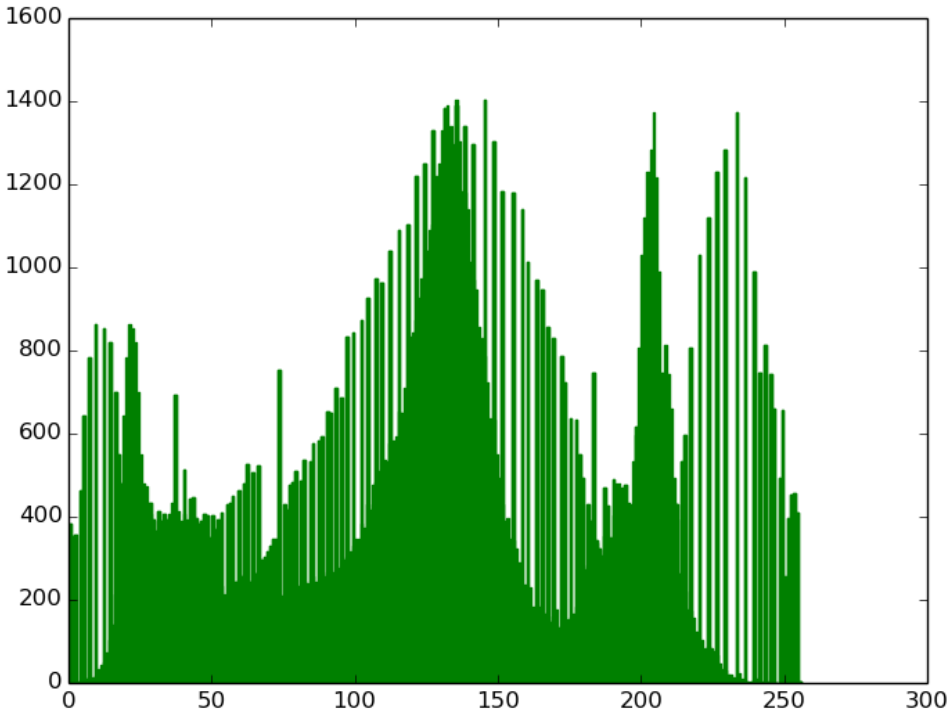
2. the result of equalizing and its histogram
    a. result

    b. its histogram



3. equalize again
    a. result

b. its histogram



4. **设计思路：**

整个 equalize_hist 函数的设计思路其实主要是利用公式

$$s_k = T(r_k) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$$

计算每个像素点的 s_k，其中 L = 256, MN = width * height。之后将该值赋给新图像上对应的点即可。

有趣的现象：

虽然图像是灰度图，可是实际上灰度图同样可以用 rgb 图来计算出直方图及进行均衡化操作。

```
forFunImage = cv2.imread("./forFun.jpg", 1)
```

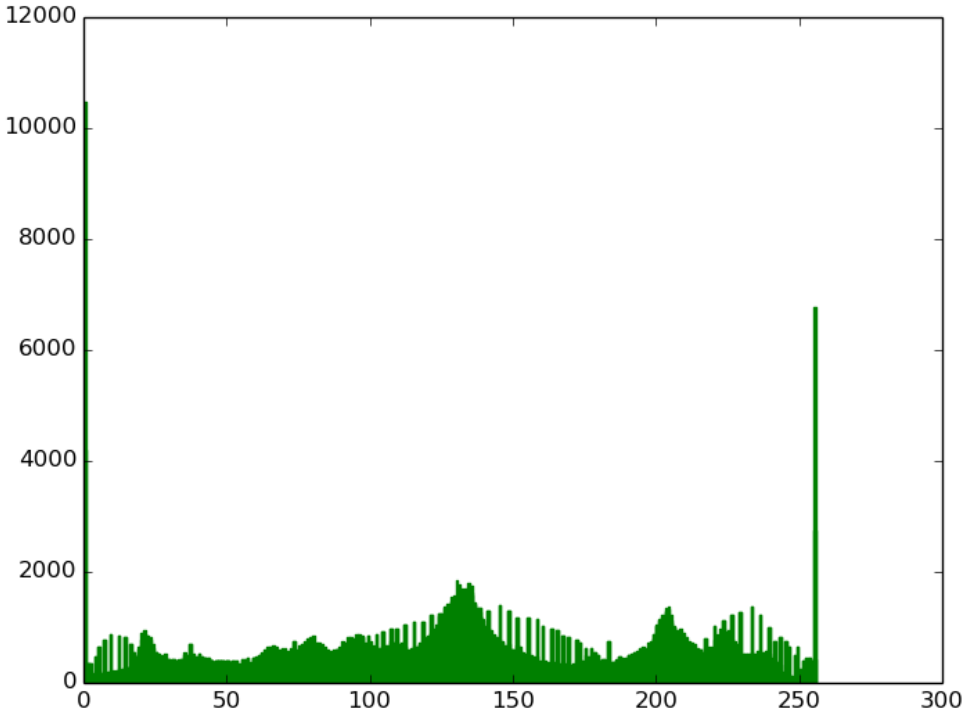
使用 rgb 模式打开一张彩色图后，将之前对灰度图的计算相对应 r, g, b 三色的灰度值及其运用公式，就可以得到相对应的 r, g, b 三色的直方图均衡化的结果，而叠加在一起的话则变成了 rgb 图的直方图均衡化，结果如下：

a. 原图

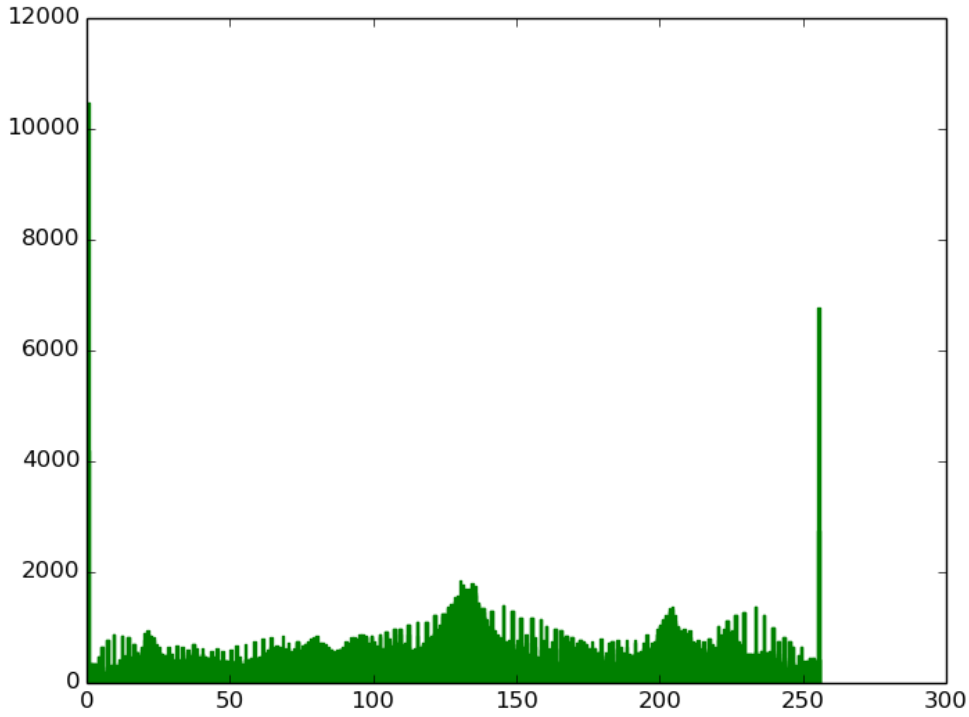
b. 彩色均衡化

c. 红色均衡化

i. 均衡化前直方图

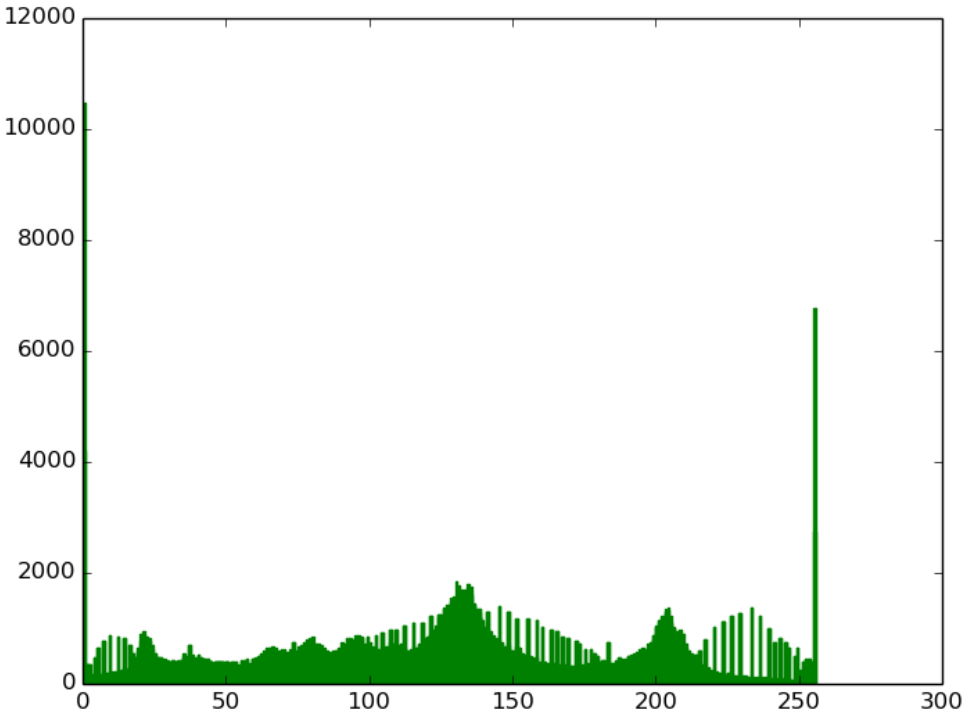

12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

ii. 均衡化后直方图

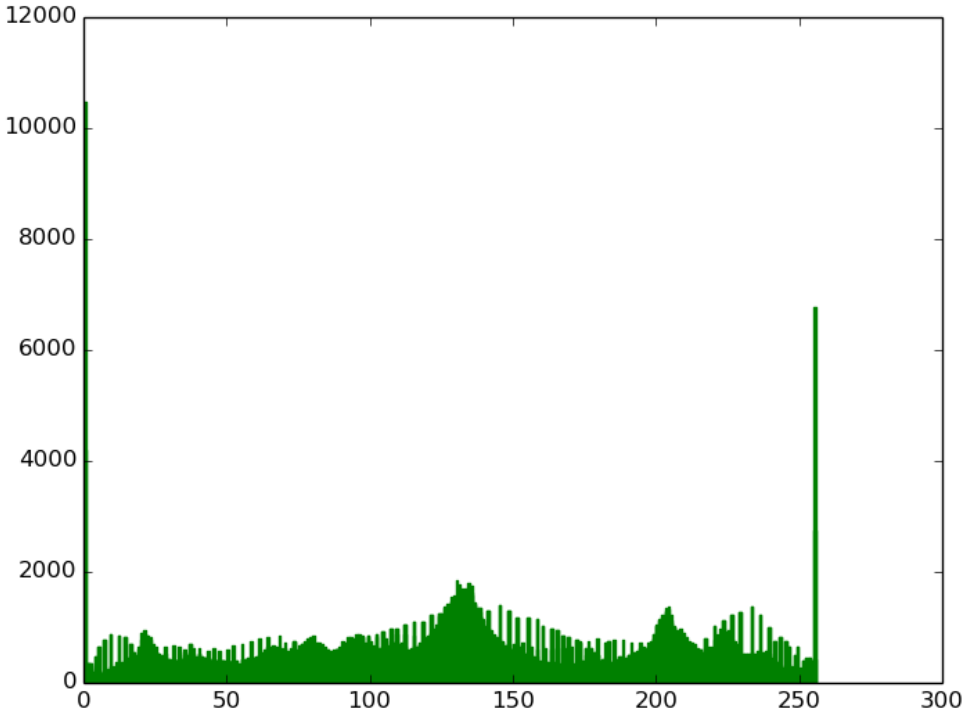

12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

d. 蓝色均衡化

i. 均衡化前直方图

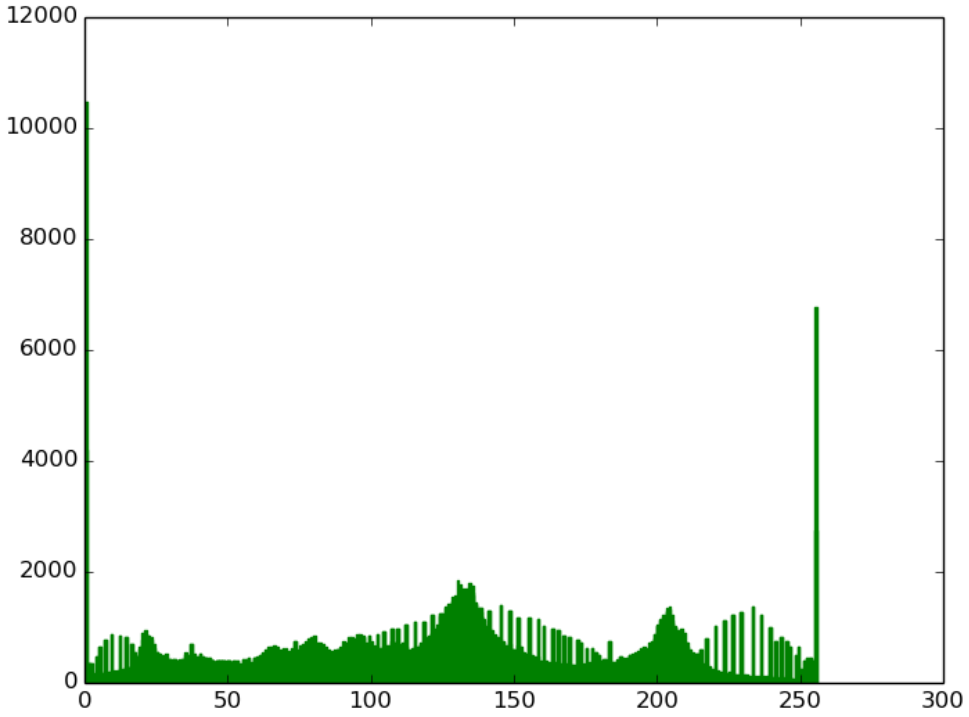


ii. 均衡化后直方图

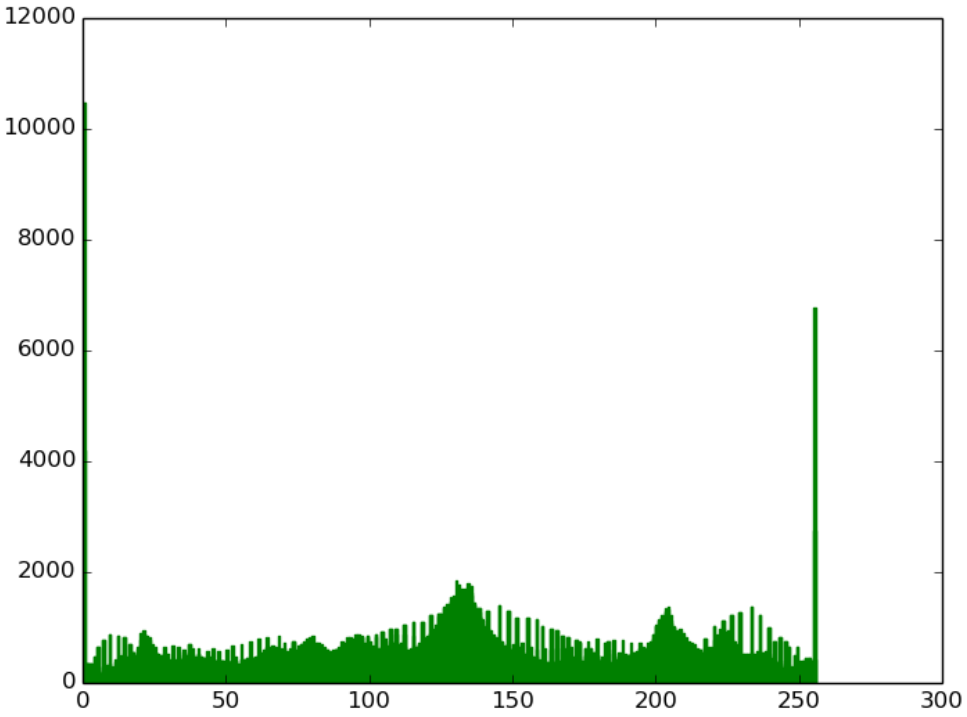


e. 绿色均衡化

i. 均衡化前直方图


12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

ii. 均衡化后直方图


12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
300

这其实是我输出了 cv2.imread 读入的图片后发现的，因为一开始我并没有传入参数 0，缺损意味着默认 rgb 模式打开，那么打印出来的灰度图结果就是 rgb 三色取值一样，后面改变打开模式完成作业后觉得其实是不是彩色图也是一样的处理方法呢？可是可以看到说比起原图，直方图均衡化后的结果都要比原图片尺寸小一点，结果对不对我就不知道了。至于为什么我没有把彩色图均衡化后的直方图打印出来，我是觉得就是三种的值对应相加，然而，我不太会 Python 导致。。运行错误了，于是干脆不打印了。。

而一开始我没有看清楚题目原来可以使用第三方包进行图片保存，所以自己写了一个函数，可是这个函数在输出图片的直方图保存上是存在问题的，因为图片灰度刚好最大值真的可能是 0…而那个算法是最大值作分母，所以这种情况要避开，可是怎么做我好像没有想到特别好的方法。

最后则是对图片的保存和保存直方图上存在问题，就是有的时候如果图片和直方图同时保存(两个函数同时执行)，那么程序会报错。我觉得这样情况可能是因为图片还没有生成保存而直方图就要计算保存了，那么就可能会出现报错。

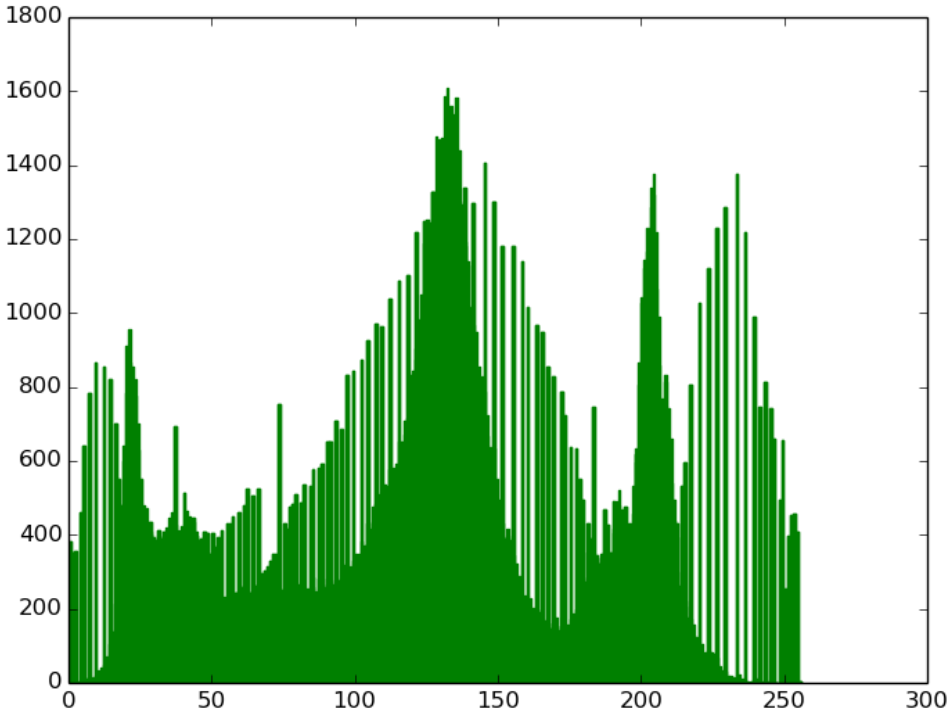
(这个自己有兴趣搞搞的部分应该不算算法部分吧，因为好像已经超过两页了，2333，不是这样计算彩色图的话，嗯，也没有关系，就玩玩而已)

2.2 Spatial Filtering

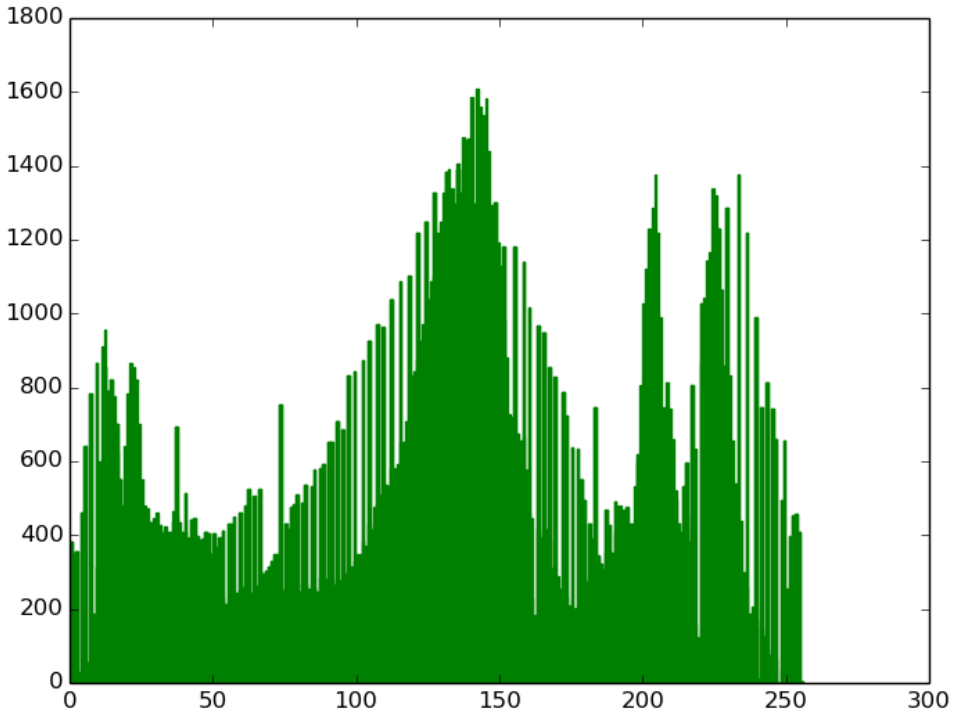
1. Smooth input image with averaging filters

   a. 3 X 3

   标定前：

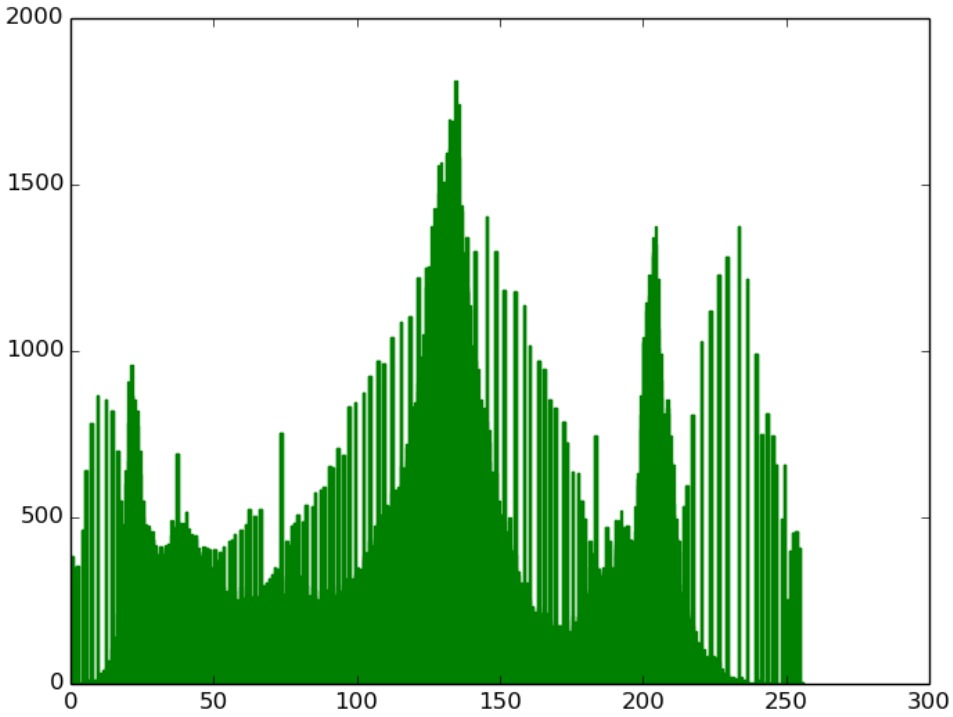


标定后：

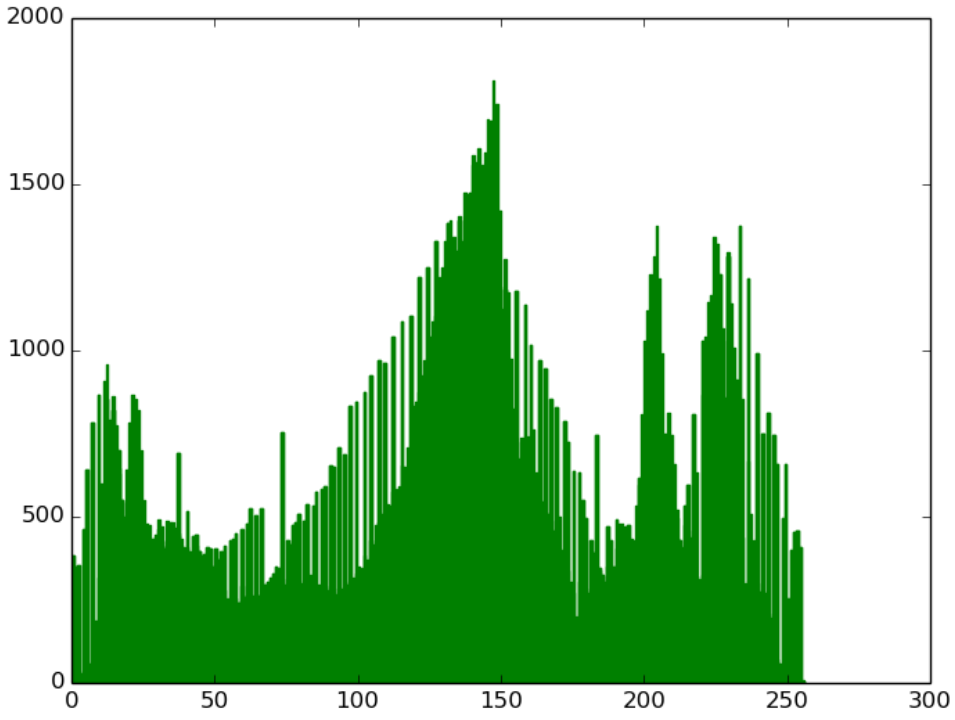

1800
1600
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300

b. 7 X 7

标定前：

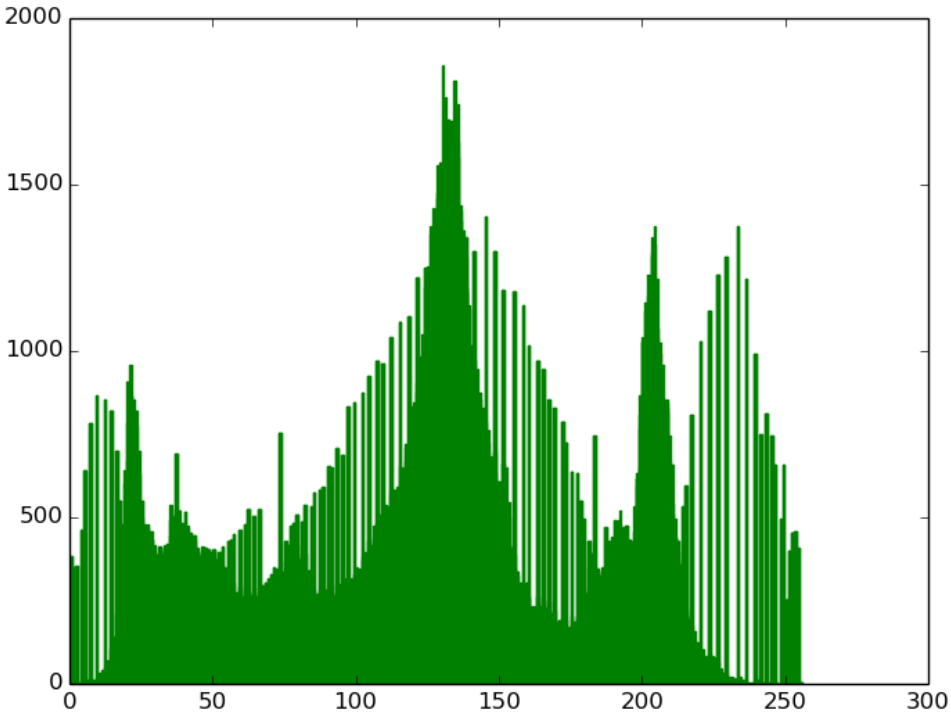


标定后：

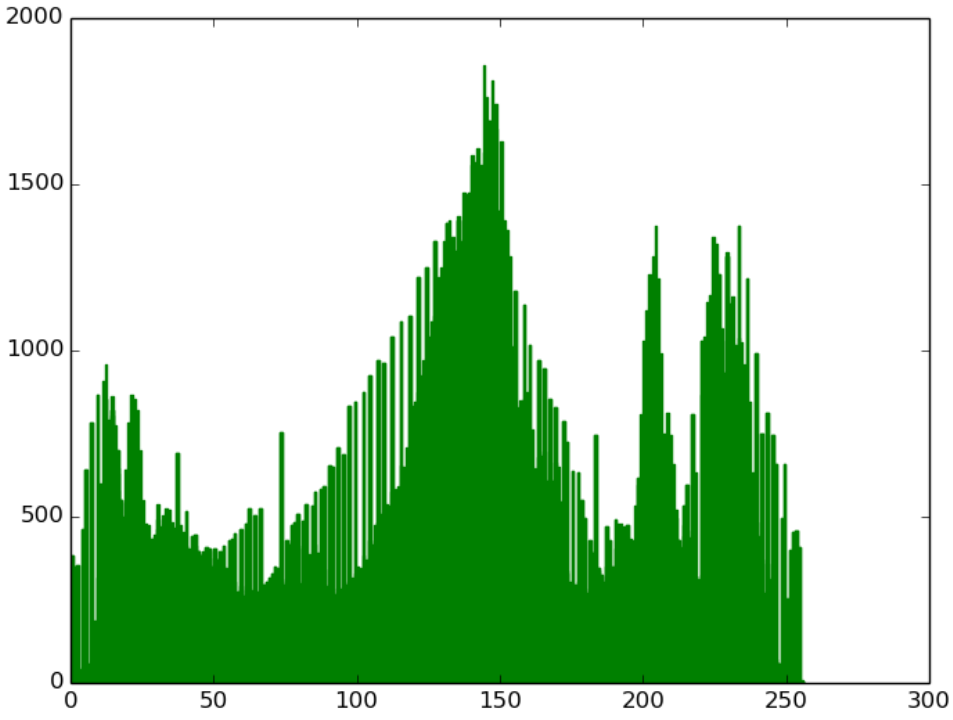


c. 11 X 11

标定前：



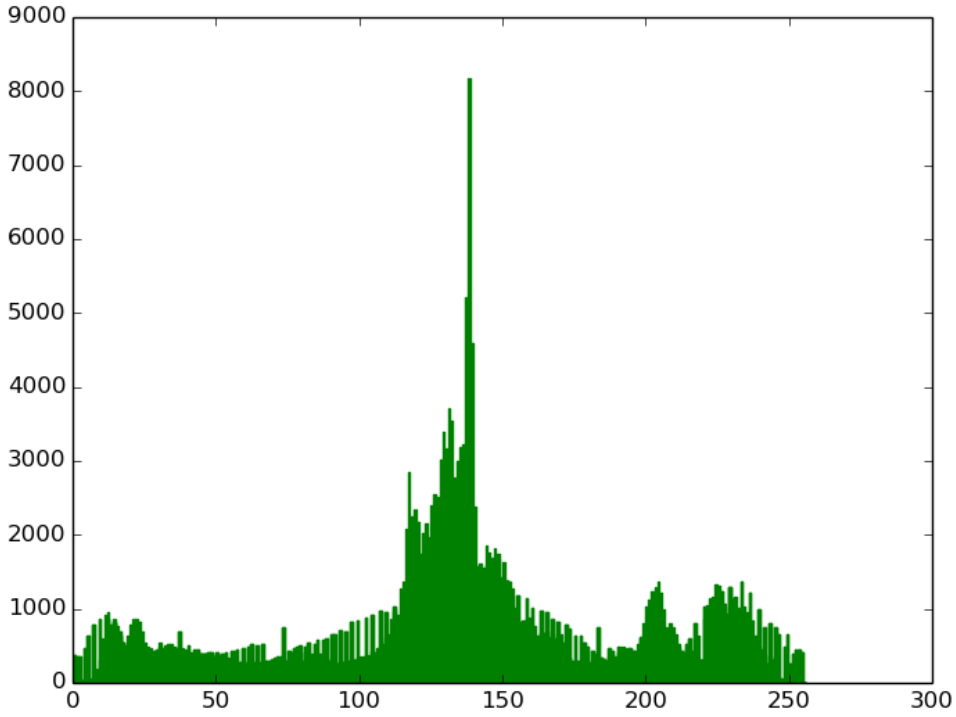
标定后：



2. Laplacian filter

标定前：

标定后：



标定前加上原图后锐化的图像：

标定后加上原图后锐化的图像：

拉普拉斯算子可以用作图像锐化的原因：

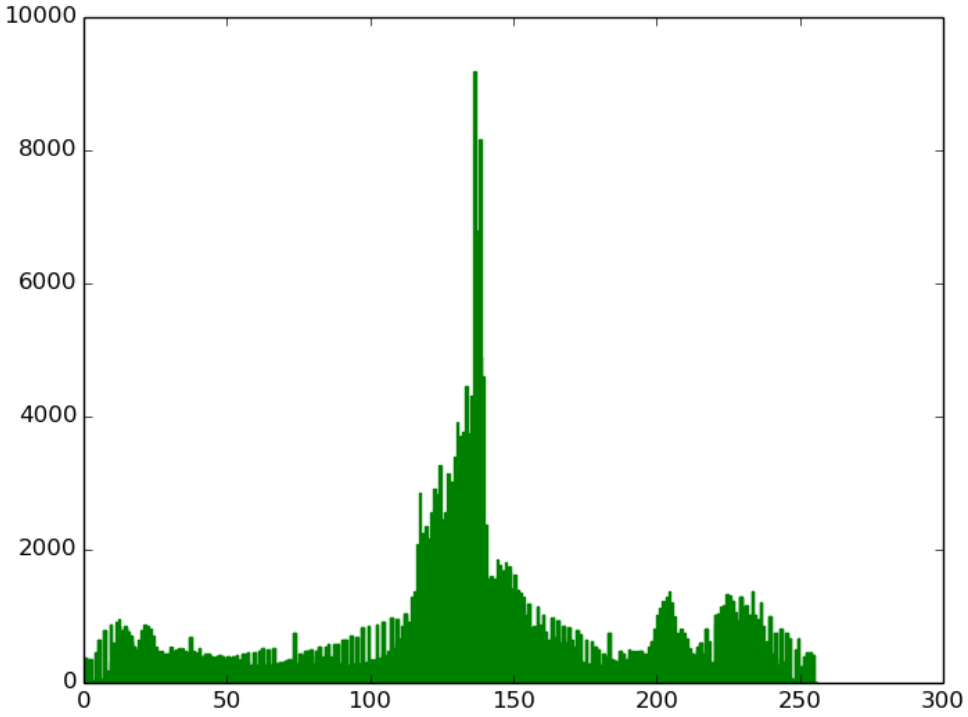
拉普拉斯滤波器是一种各向同性滤波器，这种滤波器的响应与滤波器作用的图像的突变方向无关。而且他其实突出了纹理。

通过拉普拉斯可以增强图像中灰度突变处的对比度，最终结果是使图像中的细节部分得到了增强，并良好地保留了图像的背景色调。

3. high-boost filter

标定前：

标定后：



4. **设计思路：**

a) 加权均值滤波器(这次好像没有加权的)

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{s=-b}^{b} w(s,t)}$$

主要就是根据这个公式得到的，其中 w 是滤波器矩阵，f 为图像矩阵。

这里我觉得奇怪的是如果按照公式来算的话，结果是这样的

这个 11 阶矩阵滤波器的结果可以看到右边和下面都有黑边，我也不知道这个是不是正确的，所以自己做相关操作的时候稍微移动了一下图片，让四边都有一下黑边，可能是我计算方法的不同(其实我在 Linux 下看到是没有的，不知道为什么 Windows 下面有黑边)。

b) 拉普拉斯滤波器

$$\Delta^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1)$$
$$+ f(x,y-1) - 8(f,y)$$
$$g(x,y) = f(x,y) + c[\Delta^2 f(x,y)]$$

实际上第一条公式里面系数的值是随着滤波器矩阵的变化而变化的，变化的结果有点难看，可是我觉得这是拉普拉斯本身强调的是图像中灰度的突变不强调灰度级缓慢变化导致的。如果想改善这种情况，可以对图像进行增强处理。

c) 高提升滤波器

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

$$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

$\bar{f}(x,y)$指的是模糊图像，这里使用了 3 阶均值过滤器结果作模糊图像，而 k 则取 4。在其他滤波器的基础上实现高提升滤波器其实不是很难。k 小于 5 的时候其实效果挺好的，突然想起自己忘记做 k = 1 和 k < 1 的两个不是高滤波图像对比了，hhh。

d) 标定

一开始我是没有意识到需要标定的，后来想到 0-255 的范围觉得有必要搞一下，就让灰度值大于 255 的等于 255，小于 0 的等于 0.可是这样我感觉十分不友好，应该是整个现有区间放大或缩小到 0-255 这一区间里面，公式参照书本 P_{46} 页得到。

$$f_m = f - \min(f)$$

$$f_s = K[f_m / \max(f_m)]$$

在这里，K = 255，这样就可以将整个区间变成了 0-255 区间了。代码中我使用了一条公式计算全部了，嗯，任性。最后得出来拉普拉斯的图和书本上面的月亮图差不多效果。