第一次技术文档

刘畅 15061183

2018 年 3 月 19 日

1 程序说明

使用语言： JAVA

编程环境： Windows 8.1 + Eclipse Neon.3 Release (4.6.3)

相关库：

java.awt

java.io

java.util

javax.imageio

主要方法：

方法名	作用
void grayscale(String outputLabel) ..	生成灰度图
void globalStretch(String outputLabel, int lowerScale, int upperScale) ..	提供变化后的灰度范围，对灰度图进行全局线性拉伸（压缩）
void localStretch(String outputLabel, int oriLowerScale, int oriUpperScale, int lowerScale, int upperScale) ..	提供原灰度图灰度范围和变化后的灰度范围，对灰度图进行局部线性拉伸，可能改变图片整体灰度范围
void piecewiseStretch(String outputLabel, int oriLowerScale, int oriUpperScale, int lowerScale, int upperScale) ..	提供原灰度图灰度范围和变化后的灰度范围，对灰度图进行局部线性拉伸，不会改变图片整体灰度范围
void histogramCorrection(String outputLabel, int partNum) ..	给定分段数，对灰度图进行直方图均衡处理

2 图片读取

2.1 BMP格式图片结构

通过查找资料，了解了BMP位图文件的格式，括号里为该部分所占字节数：

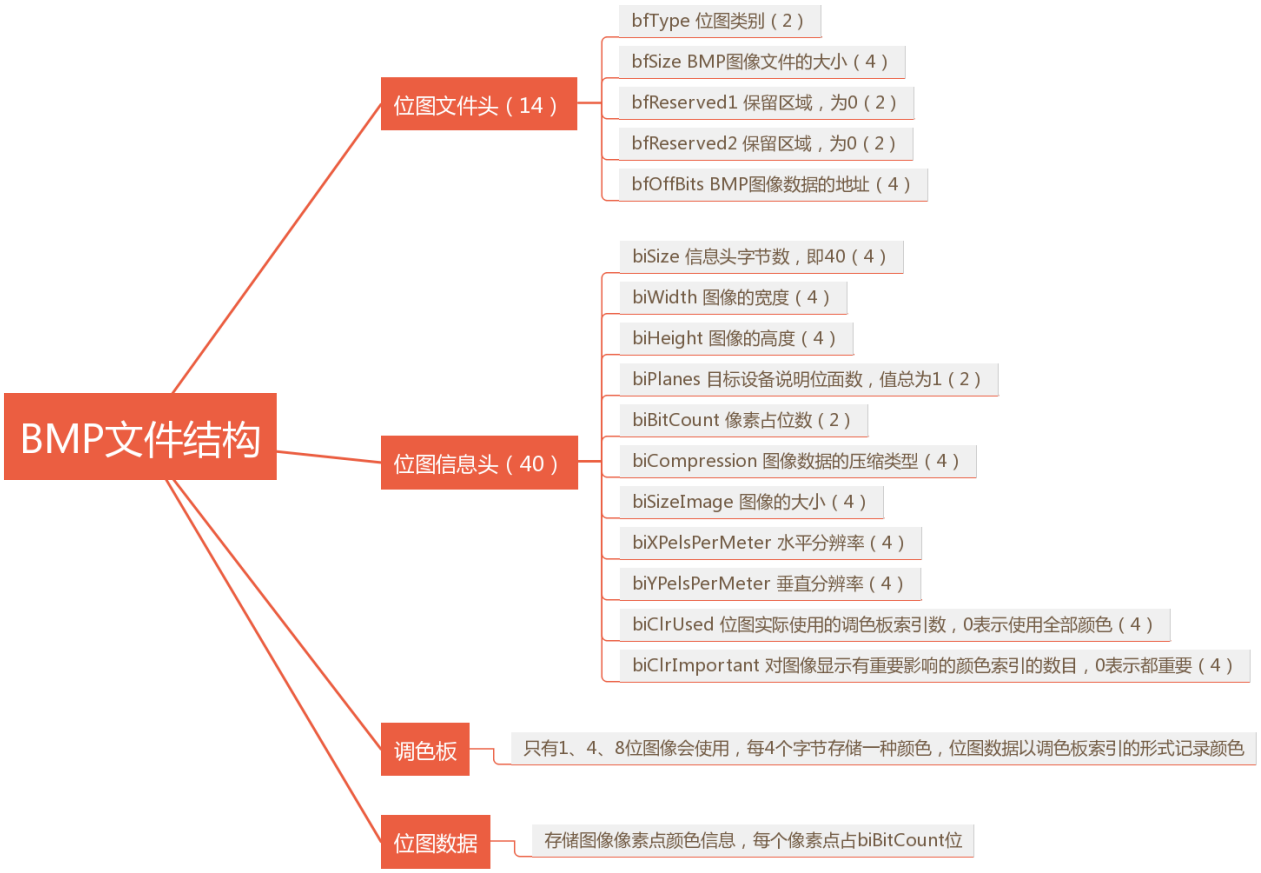


图 1: BMP文件结构图示

在BMP文件中，像素点的颜色是用RGB或ARGB的形式记录的，其中A表示Alpha，代表像素点的透明度。在读取BMP文件的时候，应当考虑到位数、压缩类型等问题，而且目前主要的图像格式还有JPEG、PNG、GIF等，想要兼顾全部类型的图片是个比较复杂的工作。

2.2 实际采用的读取方式

在实际编码过程中，我使用了javax.imageio库读取图片文件，可以通过内置的方法获取各个像素的ARGB值，简化了这部分工作。

3 准备工作：图片灰度化

3.1 实现过程

3.1.1 灰度矩阵的获取

在对一个宽度为w、高度为h图片进行**直方图均衡**和**灰度线性拉伸**处理之前，需要先对图像进行灰度化处

理。假设某像素灰度级为Y，根据像素的R、G、B值利用公式

$$Y = 0.3R + 0.59G + 0.11B$$

将每个点的灰度级Y记录在灰度矩阵$M(w,h)$中。

3.1.2 输出灰度图

根据灰度矩阵$M(w,h)$，建立一个宽度为w，高度为h的新图片，并将图片上任意一个像素点的R、G、B值都设定为$M(w,h)$中对应位置的灰度级Y，保证图片各个像素点R、G、B值相等。该图即为原图的灰度图。

3.2 处理效果

(a) 原彩色图　　(b) 处理后灰度图

图 2: 灰度处理效果

4 任务一：直方图均衡

4.1 算法实现

根据读取图像得到的灰度矩阵$M(w,h)$，可以计算出第i种灰度所占的像素数量n_i。假设灰度级为L（在该程序中，取$L = 256$），则第k个灰度级均衡变换后的新灰度级s_k可由以下公式得到：

$$s_k = (L-1)\sum_{i=0}^{k}\frac{n_i}{wh}$$

4.2 主要步骤

(1) 读取图片

(2) 将图片转变为灰度矩阵

(3) 根据灰度矩阵计算出各个灰度级的像素数量

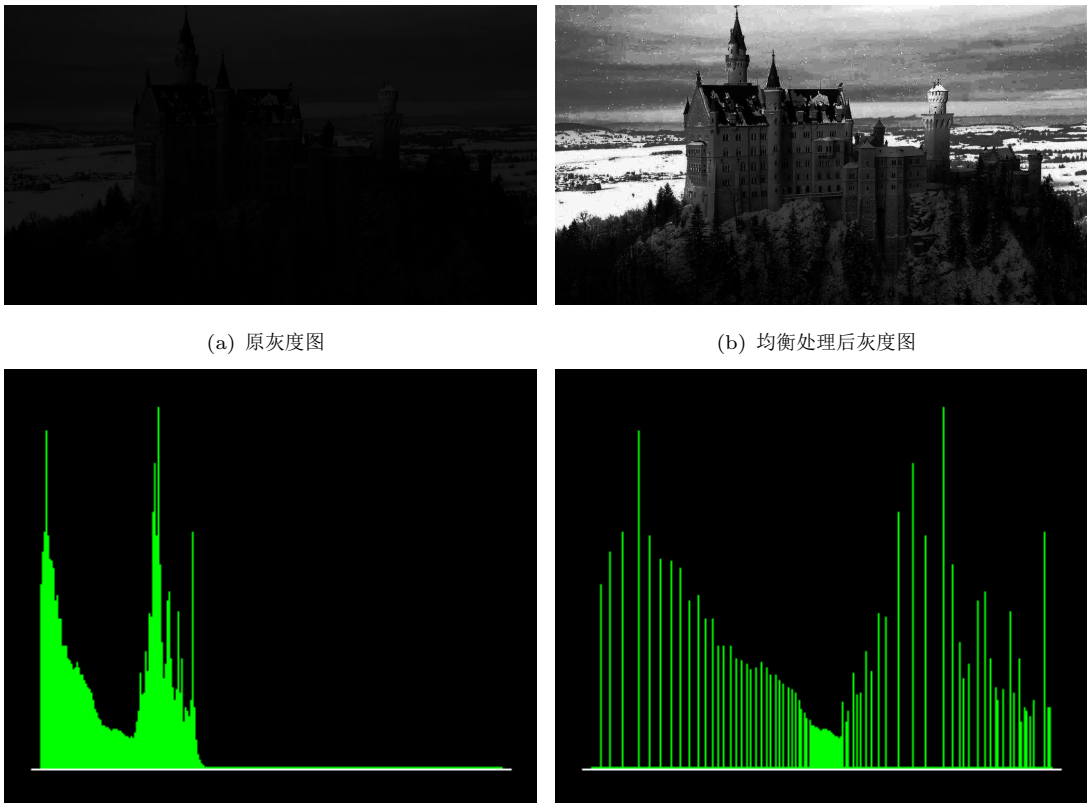
(4) 利用公式计算得到原灰度级与新灰度级的映射关系

(5) 根据映射关系将原灰度矩阵转换为新灰度矩阵

(6) 输出图片

4.3 处理效果

4.3.1 图片一



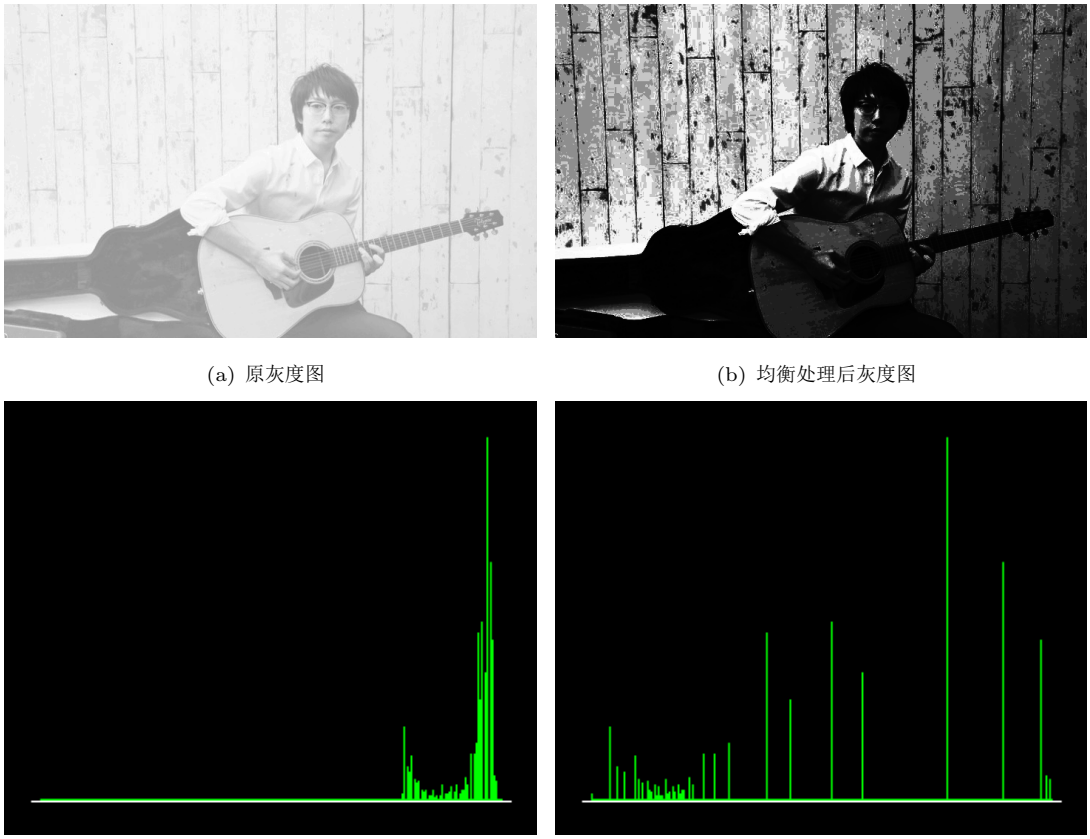
(a) 原灰度图 (b) 均衡处理后灰度图

(c) 原灰度分布 (d) 均衡处理后灰度分布

图 3: 图片二直方图均衡效果

可以看到，整体偏暗的城堡画面经过直方图均衡处理后变得清晰许多，直方图也的确变得更分散了，这符合算法的预期。

4.3.2 图片二



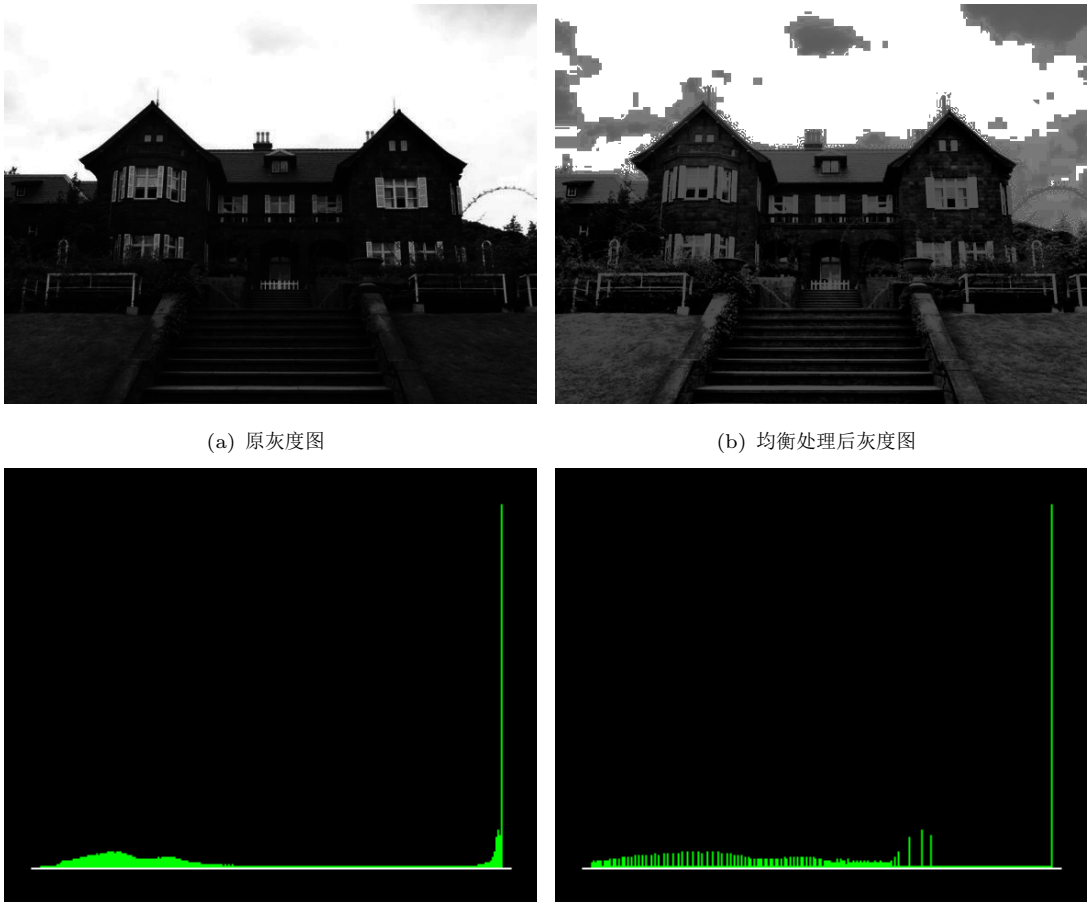
(a) 原灰度图 (b) 均衡处理后灰度图

(c) 原灰度分布 (d) 均衡处理后灰度分布

图 4: 图片二直方图均衡效果

在图片二中，直方图均衡的结果虽然正确，但其效果并不理想，主要体现在均衡结果中人物部分的灰度级过高，以至于看不清具体细节。

4.3.3 图片三



(a) 原灰度图 (b) 均衡处理后灰度图

(c) 原灰度分布 (d) 均衡处理后灰度分布

图 5: 图片三直方图均衡效果

在图片三中，天空部分出现了出现了不自然的像素块，效果也不是很理想。

4.4 改进

4.4.1 算法

从图3中可以看到，效果不理想的主要原因是图片整体灰度级较低，而图片中人物的灰度级相较偏高，导致均衡后灰度级大大上升。为了解决大量低灰度对高灰度的干扰，我尝试将处理前的灰度级进行分段操作，如低灰度区和高灰度区，然后在这两部分分别进行直方图均衡处理，避免了低灰度区对高灰度区的干扰。

假设灰度的分段数为p，首先采用以下公式计算出平均每段的像素数n_{avg}：

$$n_{avg} = \frac{wh}{p}$$

之后，根据n_avg对灰度级由低至高进行分段，尽可能保证每段内像素数总和相等。假设第k个灰度级所处段的

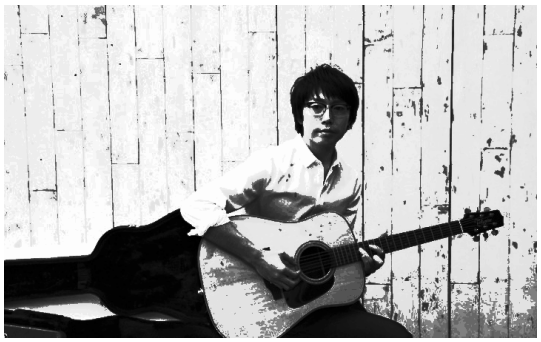
最低灰度级为r_{min}，最高灰度级为r_{max}，则其均衡变换后的新灰度级s_k可由以下公式得到：

$$s_k = r_{min} + (r_{max} - r_{min}) \sum_{i=r_{min}}^{k} \frac{n_i}{\sum_{j=r_{min}}^{k} n_j}$$

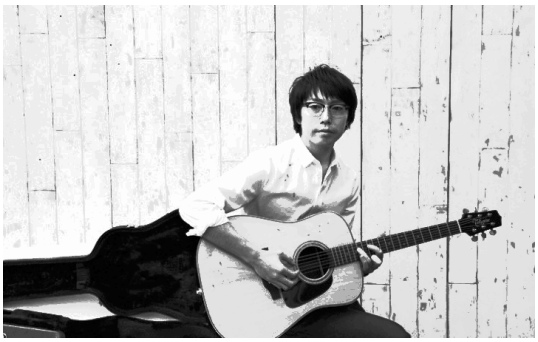
4.4.2 效果

(a) 直方图均衡后灰度图（p=1）

(b) 直方图均衡后灰度图（p=2）



(c) 直方图均衡后灰度图（p=3）

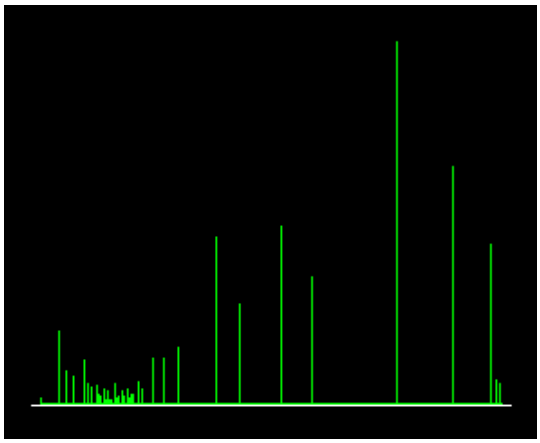


(d) 直方图均衡后灰度图（p=4）

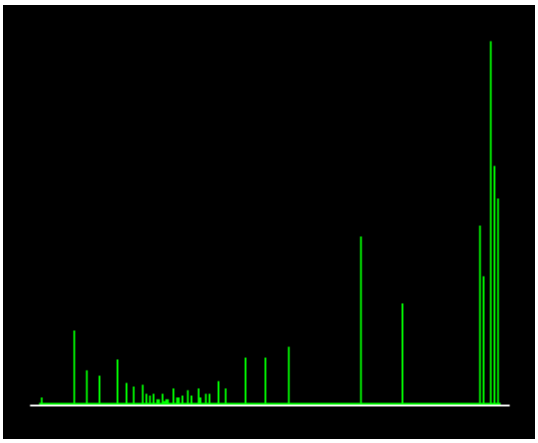
(e) 直方图均衡后灰度图（p=5）

(f) 原灰度图

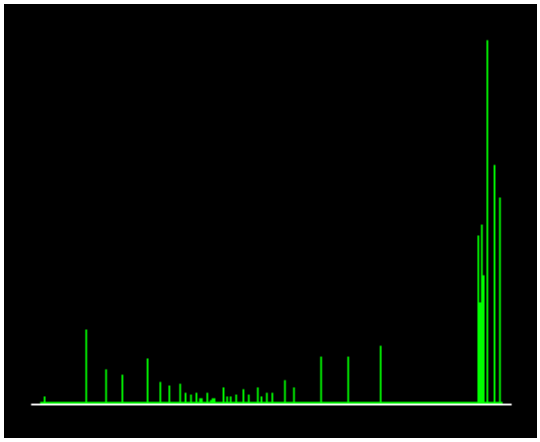
图 6: 图片二直方图均衡优化处理后灰度图效果



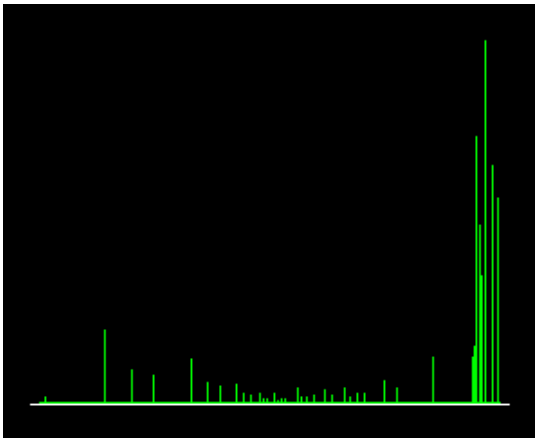
(a) 直方图均衡后直方图（p=1）



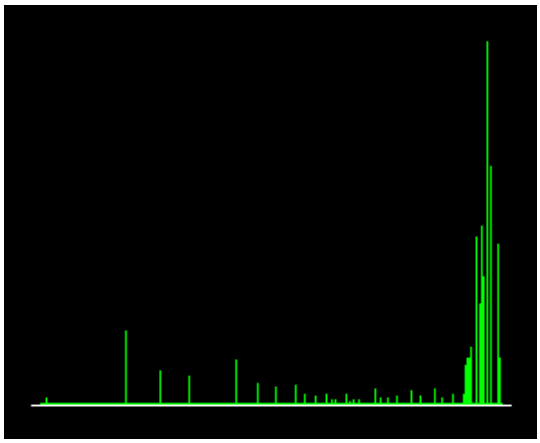
(b) 直方图均衡后直方图（p=2）



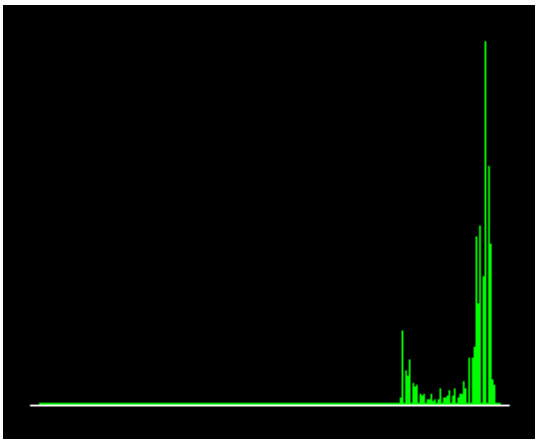
(c) 直方图均衡后直方图（p=3）



(d) 直方图均衡后直方图（p=4）



(e) 直方图均衡后直方图（p=5）



(f) 原直方图

图 7: 图片二直方图均衡优化处理后灰度分布

从结果可以发现，随着分段数p的增大，均衡后的灰度图越来越接近于原图；随着p的减小，均衡后的灰度图越来越接近于优化前的均衡灰度图（$p = 1$）。就效果而言，取$p = 4$与$p = 5$的均衡灰度图效果优于前三幅，说明这种优化是合理的。

(a) 直方图均衡后灰度图（p=1）

(b) 直方图均衡后灰度图（p=2）

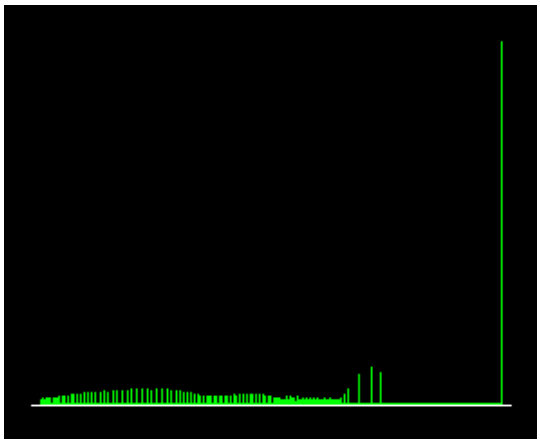
(c) 直方图均衡后灰度图（p=3）

(d) 直方图均衡后灰度图（p=4）

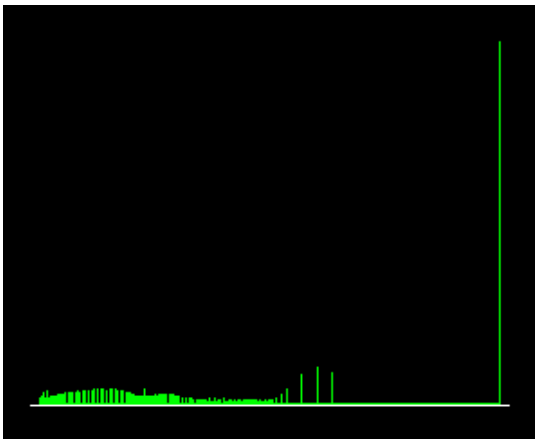
(e) 直方图均衡后灰度图（p=5）

(f) 原灰度图

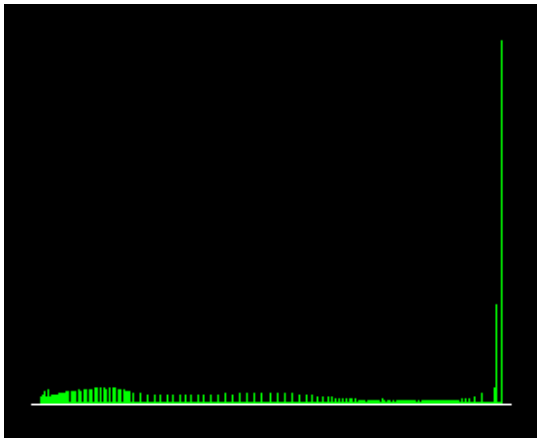
图 8: 图片三直方图均衡优化处理后灰度图效果



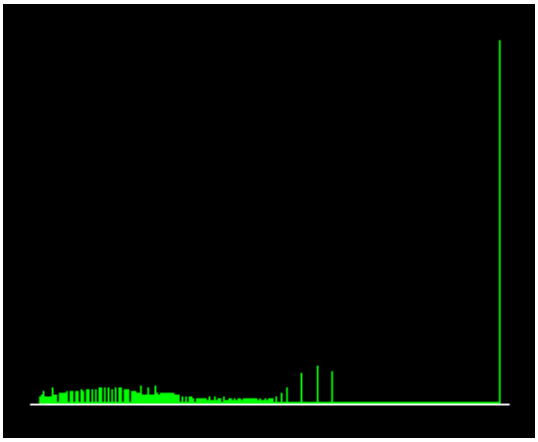
(a) 直方图均衡后直方图（p=1）



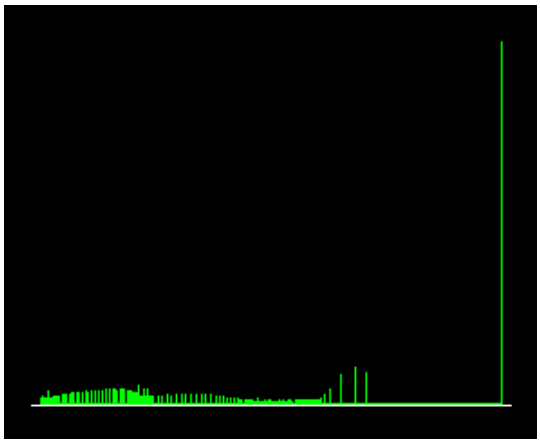
(b) 直方图均衡后直方图（p=2）



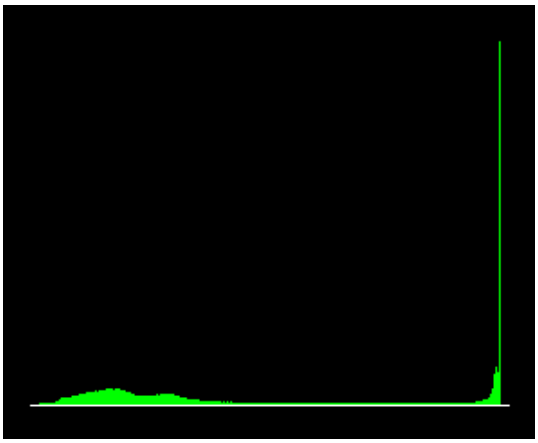
(c) 直方图均衡后直方图（p=3）



(d) 直方图均衡后直方图（p=4）



(e) 直方图均衡后直方图（p=5）



(f) 原直方图

图 9: 图片三直方图均衡优化处理后灰度分布

但是对于图片三，上述规律并不明显，主要体现在存在一个像素数十分庞大的灰度级255，由于灰度级是离散的，整体的均衡性导致了灰度级断层的出现，从而无法呈现出较浅的像素。但是这里取$p = 3$呈现出的效果仍然自然，说明在这种情况下，合适的段数p仍可以优化均衡的效果。

但这并不代表着对于所有图片，取$p=4$与$p=5$结果都为最好，比如在用图片一进行分段的时候，发现还是取$p=1$即不优化的效果最好，说明p的最终取值还要由不同图片的性质决定。

4.5 总结

两种图片结果的比较，使我们发现直方图均衡并不是万能的。经分析，优化前的算法不适合处理小主体图片，因为如果这样，和主体无关的部分（比如背景）覆盖的灰度就会成为图片的主要灰度，在均衡的时候，主要灰度的扩散会严重影响主体相关的灰度，导致背景细节被突出，主体细节被减弱。优化后的算法试图将较深和较浅的灰度分隔开，一定程度上保护了主体相关的灰度，对部分图片可以达到更好的效果。

同时，直方图均衡并不适合某种灰度级数量过多的情况，因为这样可能会导致灰度的断层，导致画面不够均匀。

5 任务二：灰度线性拉伸（压缩）

5.1 算法实现

将灰度图像$f(x,y)$中$[a,b]$的灰度范围变化为$[c,d]$，有两种处理方式。

若要使得变换后所有灰度都处于$[c,d]$内，则变换函数T_l可以表示为

$$g(x,y)=\begin{cases} c & ,0<=f(x,y)<a \\ \frac{d-c}{b-a}*(f(x,y)-a)+c & ,a\le f(x,y)\le b \\ d & ,b<f(x,y)<=255 \end{cases}$$

若要使得变换后整体灰度范围不变，则可以采用分段线性变换。假设原图灰度范围为$[Y_{min},Y_{max}]$则变换函数T_{pl}可以表示为

$$g(x,y)=\begin{cases} \frac{c-Y_{min}}{a-Y_{min}}*(f(x,y)-Y_{min})+Y_{min} & ,Y_{min}<=f(x,y)<a \\ \frac{d-c}{b-a}*(f(x,y)-a)+c & ,a\le f(x,y)\le b \\ \frac{Y_{max}-d}{Y_{max}-b}*(f(x,y)-b)+d & ,b<f(x,y)<=Y_{max} \end{cases}$$

5.2 主要步骤

(1) 读取图片

(2) 将图片转变为灰度矩阵

(3) 根据提供的灰度级范围，得到原灰度级与新灰度级的映射关系

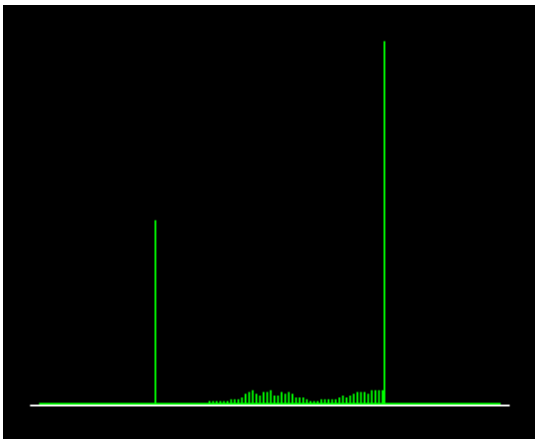
(4) 根据映射关系将原灰度矩阵转换为新灰度矩阵

(5) 输出图片

5.3 处理效果

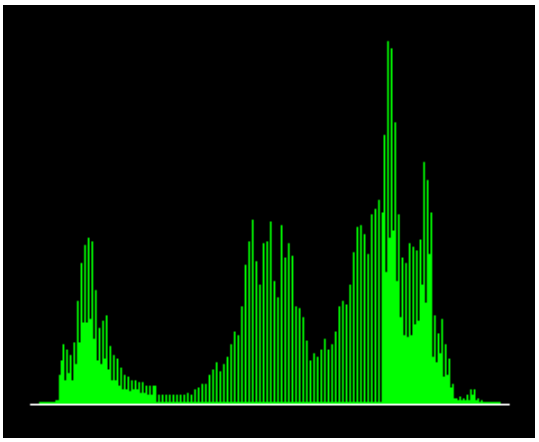
5.3.1 局部线性拉伸

(a) 改变范围拉伸后灰度图



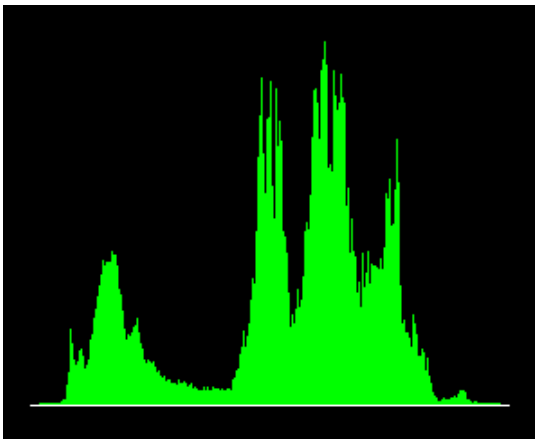
(b) 改变范围拉伸后直方图

(c) 不改变范围拉伸后灰度图



(d) 不改变范围拉伸后直方图

(e) 原灰度图



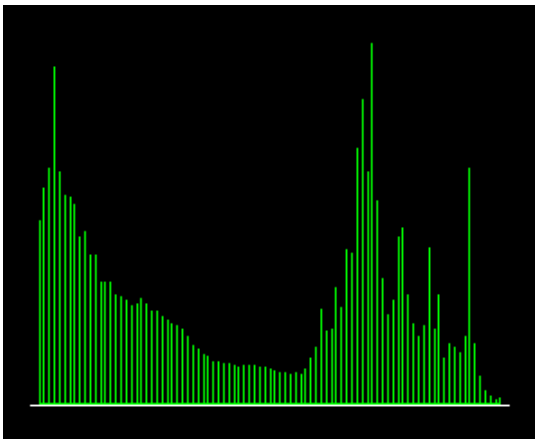
(f) 原直方图

图 10: 局部线性拉伸效果

该实例为将原灰度图[96, 159]范围的灰度级拉伸至[64, 191]范围的结果。

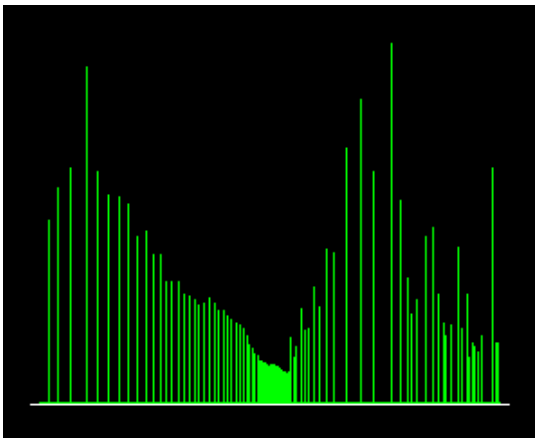
5.3.2 全局线性拉伸

(a) 线性拉伸后灰度图

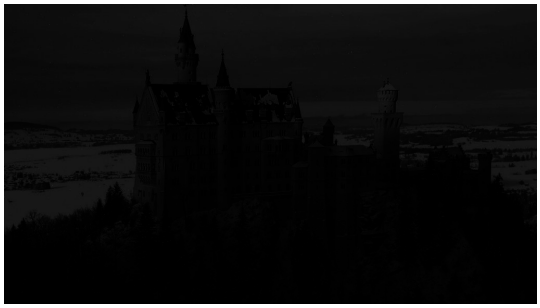


(b) 线性拉伸后直方图

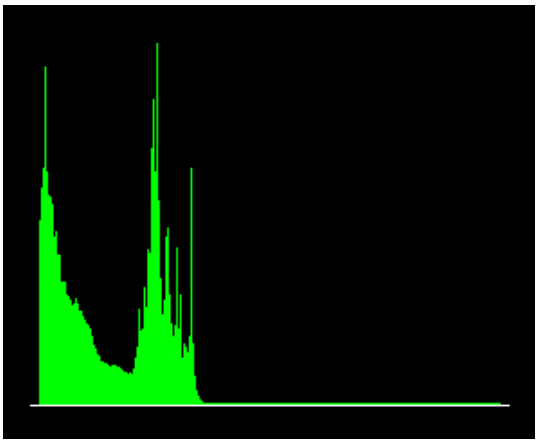
(c) 直方图均衡后灰度图



(d) 直方图均衡后灰度图



(e) 原灰度图



(f) 原直方图

图 11: 线性拉伸与直方图均衡效果对比

该实例为将原灰度图灰度范围拉伸至$[0, 255]$范围的结果，直方图均衡分段数$p = 1$。

5.4 总结

灰度线性调整可以自由地对某一区域灰度进行拉伸或压缩，有选择性地突出或抑制灰度区间，较为灵活。但是全局线性拉伸呈现出的效果不如直方图均衡清晰，说明灰度线性调整不适合于挖掘图像细节。