**实验报告**

专业：计算机科学与技术

姓名：薛柔

学号：3220104854

日期：2023/10/29

课程名称： 图像信息处理 指导老师： 宋明黎 成绩：

实验名称： 图像的对数增强和直方图均衡化

**一、实验目的和要求**

1. 学会对图像进行对数增强。

2. 学会对图像进行直方图均衡化。

**二、实验内容和原理**

实验内容：

1. 对图像进行对数操作

2. 对图像的直方图进行均衡化操作

实验原理：

1. 对数增强操作

对图像进行对数变换的原理是通过将图像的亮度值转换为对数值来调整图像的对比度和亮度。对数变换符合人眼的感受阈值，可以用来扩展图像的灰度级范围，同时增强较暗区域的细节。

对数变换的公式为：

Ld =log(Lw + 1)/log(Lmax + 1)

Ld是显示亮度，Lw是真实世界亮度，Lmax是场景中的最亮值。

这个映射能够确保不管场景的动态范围是怎么样的，其最大值都能映射到 1（白），其他的值能够比较平滑地变化。

2. 直方图均衡化

2.1 图像的直方图

将图像的灰度分为多个值，以8个为例（灰度值分别为0，1，2，3，4，5，6，7），则每个像素都有其对应的灰度值。统计每个灰度值像素的数量，并计算占总像素数的比例。以灰度值为横坐标，对应像素比例为纵坐标，画出柱状图，即为该图像的直方图。

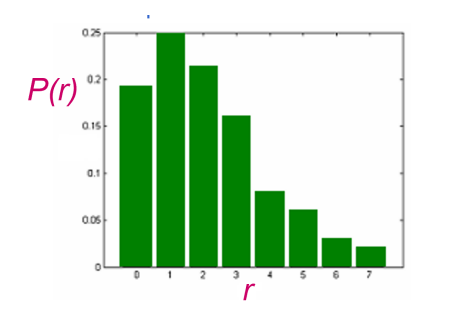


图2.1 8灰度图像的直方图

P(r) 表示在给定图像的总像素数中，不同灰度级rk的像素数所占的比例。而且有

∑P(rk) = 1

对一般的灰度级有256个的图像，则将[0,7]变为[0,255]，同理得到灰度直方图。

彩色直方图则表示给定图像中不同 r、g、b 等级的像素数占总像素数的比例。

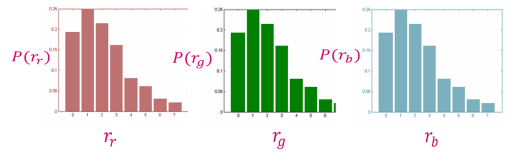


图2.2 彩色图像的直方图

图像直方图由于其计算代价较小，且具有图像平移、旋转、缩放不变性等众多优点，广泛地应用于图像处理的各个领域，特别是灰度图像的阈值分割、基于颜色的图像检索以及图像分类。

一张图的直方图唯一，而两张不同的图可以拥有相同的直方图。

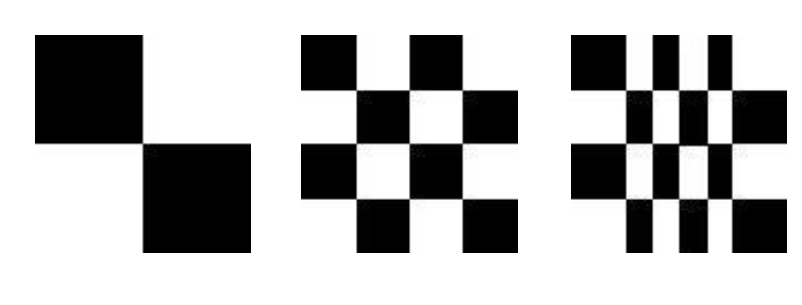


图2.3直方图相同的不同图像

2.2 直方图均衡化

直方图均衡化是将原图像通过某种变换，得到一幅灰度直方图为均匀分布的新图像的方法。直方图均衡化方法的基本思想是对在图像中像素个数多的灰度级进行展宽，而对像素个数少的灰度级进行缩减。从而达到清晰图像的目的。

直方图均衡化的关键在于找到一个映射T，令s=T(r)，能让下图左边的直方图变为右边更均衡的直方图。

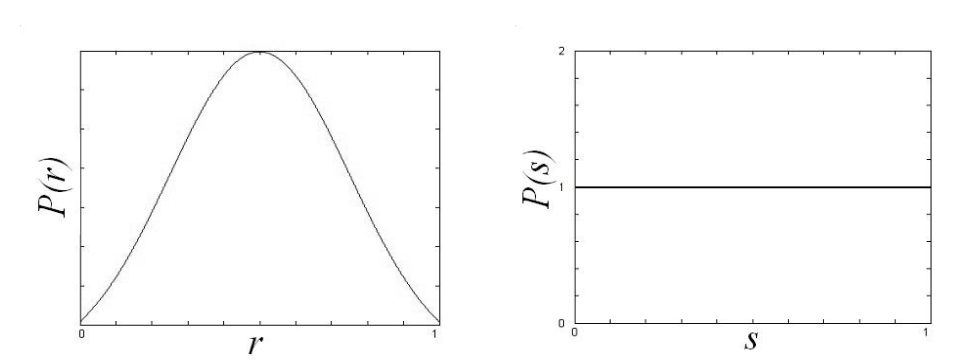
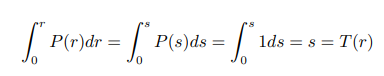


图2.4 直方图均衡化的图像变化

连续情况的T：



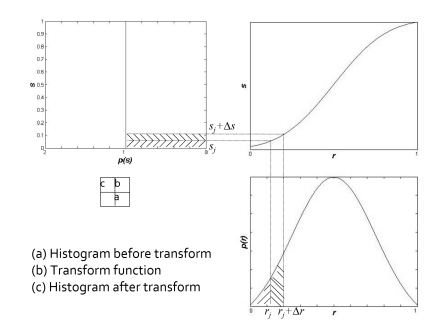
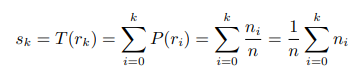


图2.5 连续直方图的均衡化

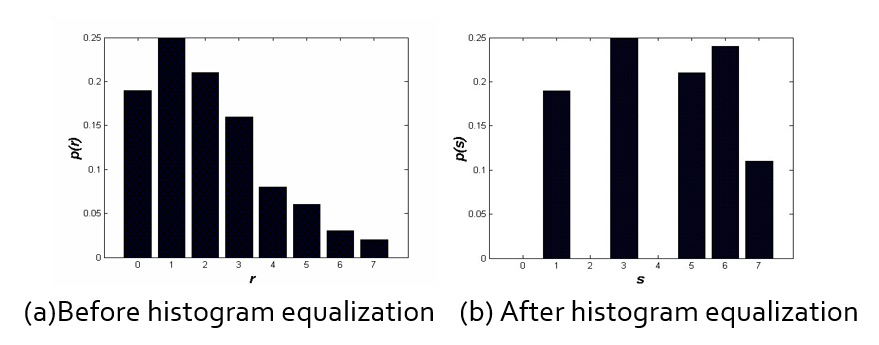
可以推出，离散情况下的T为：



直方图均衡化的过程即为找到下面为例的对照表：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *k* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *rk* | 0 | 1/7 | 2/7 | 3/7 | 4/7 | 5/7 | 6/7 | 1 |
| *nk* | 790 | 1023 | 850 | 656 | 329 | 245 | 122 | 81 |
| *P(rk)* | 0.19 | 0.25 | 0.21 | 0.16 | 0.08 | 0.06 | 0.03 | 0.02 |
| *sk* | 0.19 | 0.44 | 0.65 | 0.81 | 0.89 | 0.95 | 0.98 | 1.00 |
| *~sk* | 1/7 | 3/7 | 5/7 | 6/7 | 6/7 | 1 | 1 | 1 |
| *sk* | *s*1 | *s*3 | *s*5 | *s*6 | *s*6 | *s*7 | *s*7 | *s*7 |
| *nk* | 790 | 1023 | 850 | 985 |  | 448 |  |  |
| *P(sk)* | 0.19 | 0.25 | 0.21 | 0.24 |  | 0.11 |  |  |

k为灰度，rk为原图k灰度的像素比例，sk为转换后图像k灰度的像素比例。例如，~s0=1/7，则令所有灰度为0的值变为1。处理后的直方图变化如下：



**三、实验步骤与分析**

1. 对数增强；

先用实验1的方式读取到图像数据并灰度化，这里不多做赘述。下面先进行RGB to YUV操作，然后对灰度Y用原理中提到的公式实现对数增强：

1. **int** maxY=0;//获取最大Y值
2. **for**(**int** i=0;i<pixel\_num;i++){
3. //a[]是原图数据
4. //RGB to YUV
5. b[i].Y=a[i].red\*0.299+a[i].green\*0.587+a[i].blue\*0.114;
6. b[i].U=a[i].red\*-0.147+a[i].green\*-0.289+a[i].blue\*0.435;
7. b[i].V=a[i].red\*0.615+a[i].green\*-0.515+a[i].blue\*-0.100;
8. //将Y控制在[0,255]
9. **if**(b[i].Y<0)    b[i].Y=0;
10. **if**(b[i].Y>255)  b[i].Y=255;
11. **if**(b[i].Y>maxY) maxY=b[i].Y;
12. }
13. //对Y值进行对数增强
14. **for**(**int** i=0;i<pixel\_num;i++){
15. //类型会自动转化
16. b[i].Y=log(b[i].Y+1)/log(maxY+1)\*255;
17. //灰度图可以表示为RGB分量都等于Y的图
18. new\_pixel[i].red=new\_pixel[i].green=new\_pixel[i].blue=b[i].Y;
19. }

输出灰度图像后，YUV to RGB，输出变换后的彩图：

1. //将灰度图转化为彩图输出
2. **for**(**int** i=0;i<pixel\_num;i++){
3. //YUV->RGB
4. **int** temp\_red=b[i].Y+1.140\*b[i].V;
5. **int** temp\_green=b[i].Y-0.395\*b[i].U+0.581\*b[i].V;
6. **int** temp\_blue=b[i].Y+2.000\*b[i].U;
7. //防止溢出
8. **if**(temp\_red>255) temp\_red=255;
9. **if**(temp\_red<0) temp\_red=0;
10. **if**(temp\_green>255) temp\_green=255;
11. **if**(temp\_green<0) temp\_green=0;
12. **if**(temp\_blue>255) temp\_blue=255;
13. **if**(temp\_blue<0) temp\_blue=0;
15. new\_pixel[i].red=temp\_red;
16. new\_pixel[i].green=temp\_green;
17. new\_pixel[i].blue=temp\_blue;
18. }
19. //将增强的彩图输出
20. //打开一个bmp空白图像并写入数据
21. fp = fopen("logenhance.bmp", "wb");
22. // fwrite(数据块首地址,元素大小, 元素个数,文件指针)
23. //与原来的图相比，文件头相同
24. fwrite(&bmfh, 14, 1, fp);//文件头
25. fwrite(&bmih, 40, 1, fp);//信息头
26. //写入位图数据
27. fwrite(&new\_pixel, 3, pixel\_num, fp);
28. fclose(fp);

2. 直方图均衡化

运用原理中提到的公式，先对灰度进行直方图均衡化：

1. //RGB->YUV
2. YUV b[pixel\_num];
3. **int** histogram[256]={0};//记录直方图不同灰度像素数
5. **for**(**int** i=0;i<pixel\_num;i++){
6. b[i].Y=a[i].red\*0.299+a[i].green\*0.587+a[i].blue\*0.114;
7. b[i].U=a[i].red\*-0.147+a[i].green\*-0.289+a[i].blue\*0.435;
8. b[i].V=a[i].red\*0.615+a[i].green\*-0.515+a[i].blue\*-0.100;
9. //将Y控制在[0,255]
10. **if**(b[i].Y<0)    b[i].Y=0;
11. **if**(b[i].Y>255)  b[i].Y=255;
12. **if**(b[i].Y>maxY) maxY=b[i].Y;
13. //记录直方图
14. **int** temp=b[i].Y;
15. }
17. //直方图均衡化操作
18. PIXEL new\_pixel[pixel\_num];
19. **int** s[256]={0};//前i个灰度的像素数量的和
21. **for**(**int** i=0;i<256;i++){
22. //通项公式
23. **if**(i==0){
24. s[0]=histogram[0];
25. }**else**{
26. s[i]=s[i-1]+histogram[i];
27. }
28. **for**(**int** j=0;j<pixel\_num;j++){
29. **int** temp=b[j].Y;
30. //改变Y值为i的像素为s[i]\*1.0/pixel\_num\*255
31. **if**(temp==i){
32. new\_pixel[j].red=new\_pixel[j].green=new\_pixel[j].blue=s[i]\*1.0/pixel\_num\*255;
34. }
35. }
36. }

这里可以直接将灰度图转化为彩图输出（同1. 对数增强），我又尝试了直接对RGB三个通道进行直方图均衡化：

1. //直接对RGB彩色通道进行直方图均衡化操作
2. **int** sR[256]={0};//R通道直方图的积分
3. **int** sG[256]={0};//B通道直方图的积分
4. **int** sB[256]={0};//G通道直方图的积分
5. sR[0]=histogramR[0];
6. sG[0]=histogramG[0];
7. sB[0]=histogramB[0];
9. **for**(**int** i=0;i<256;i++){
10. //通项公式
11. **if**(i!=0){
12. sR[i]=sR[i-1]+histogramR[i];
13. sG[i]=sG[i-1]+histogramG[i];
14. sB[i]=sB[i-1]+histogramB[i];
15. }
17. //改变RGB值为i的像素为s[i]\*1.0/pixel\_num\*255
18. **for**(**int** j=0;j<pixel\_num;j++){
19. **if**(a[j].red==i){
20. new\_pixel[j].red=sR[i]\*1.0/pixel\_num\*255;
21. }
22. **if**(a[j].green==i){
23. new\_pixel[j].green=sG[i]\*1.0/pixel\_num\*255;
24. }
25. **if**(a[j].blue==i){
26. new\_pixel[j].blue=sB[i]\*1.0/pixel\_num\*255;
27. }
28. }
29. }

**四、实验环境及运行方法**

* 实验环境：Windows 11 系统

gcc 10.3.0 (tdm64-1) x86\_64-w64-mingw32

* 运行方法：先将dip\_hw3.exe文件和saber.bmp放在同一个目录下，运行dip\_hw3.exe文件，会得到5张新的图像，分别命名为logenhance\_gray.bmp，logenhance.bmp，histogram\_gray.bmp，histogram\_color.bmp，histogram\_RGB.bmp即为对数增强后的灰度图和彩图，以及灰度直方图均衡化后的灰度图和彩图，RGB通道直方图均衡化的图像。当终端出现“Successfully open the image”时说明我们成功打开了图像，否则会输出“BMP Image Not Found!”。

如果运行不成功，可以将文件夹中的.c源文件重新编译运行。

**五、实验结果展示**

对数增强的输出结果：



输入图像：saber.bmp

对数增强后的灰度图：logenhance\_gray.bmp

彩图：logenhance.bmp

直方图均衡化的输出结果：



灰度均衡化的灰度图：histogram\_gray.bmp

灰度均衡化的彩图：histogram\_color.bmp

RGB通道均衡化的图像：histogram\_RGB.bmp

**六、心得体会**

从结果图可以看出，对数增强加强了图像的亮度，可以让暗的地方差距拉大，细节更多，而亮的地方差距会减小，因此我选择了一张背景较暗的图像。一开始我选择了的一张很亮的输入图，输出效果较差，很不清晰。这是一个试错的过程。



图6.1 输入的色调很亮的图



图6.2 对数增强后输入的不清晰的结果图

从直方图均衡化结果图可以看出，均衡灰度的图像色调较暖，偏红，而均衡RGB通道的图像颜色偏冷，偏向蓝绿。

实验过程中，我一开始直接对b[i].Y进行操作，例如对标号为1的像素，b[1].Y初始为50，均衡化之后变为70，我后面遍历到灰度为70的像素时，又将b[1].Y变为90。这样过后，整张图大部分为白，小部分为黑，操作错误。后来我选择不在b[i].Y上直接操作，而是利用中间变量，得到结果。

实验后，为了验证实验结果的正确性，我在photoshop上进行了直方图均衡化操作（色调均衡），结果如下：



图6.3 photoshop的直方图均衡化结果

这与均衡灰度图后转化为彩图的结果相近。说明实验操作基本正确。我的结果偏红的原因可能是经验公式的不同和精度的偏差。

经过这两种方法后，我们都能在较暗的背景中看到更多细节，两种方法都能有效地提高图像可视信息。