二、实验内容和原理

1、直方图均衡化 Histogram equal

这种方法通常用来增加图像的全局对比度，尤其是当图像的有用数据的对比度相当接近的时候。通过这种方法，亮度可以更好地在直方图上分布。这样就可以用于增强局部的对比度而不影响整体的对比度，直方图均衡化通过有效地扩展常用的亮度来实现这种功能。

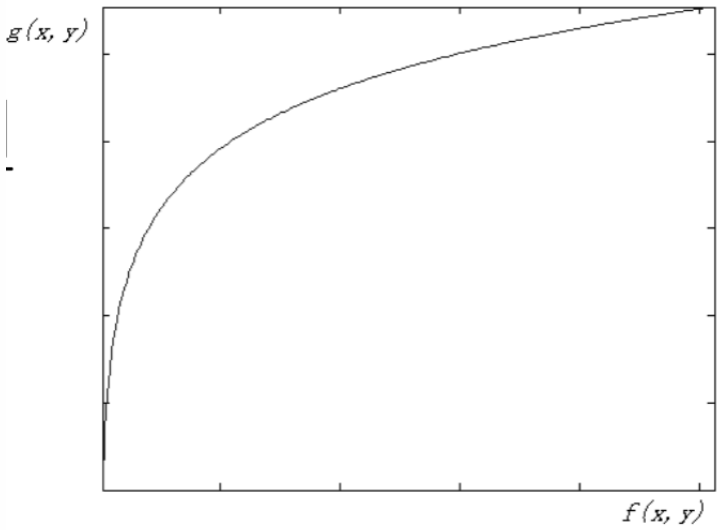
这种方法对于背景和前景都太亮或者太暗的图像非常有用，这种方法尤其是可以带来X光图像中更好的骨骼结构显示以及曝光过度或者曝光不足照片中更好的细节。这种方法的一个主要优势是它是一个相当直观的技术并且是可逆操作，如果已知均衡化函数，那么就可以恢复原始的直方图，并且计算量也不大。这种方法的一个缺点是它对处理的数据不加选择，它可能会增加背景噪声的对比度并且降低有用信号的对比度。

直方图均衡化可以理解为是一种映射，这种映射会将某些出现概率很低的灰度值都映射到一个灰度值，以达到归并的效果，减少了灰度值的个数，从而拉开对比度。原理可以追溯到：任一概率密度函数，其对应的概率分布函数是一个0-1的均匀分布。对离散的概率值求概率分布函数，可以获得一系列的新的概率值，对某些位置的相近的概率值可以认为是同一个灰度值，从而达到了简并的效果

$$\int_{-\infty}^{r} P(r)dr = F(r) \sim U(0,1) \tag{1}$$

2、对数变换 Logarithm

$$g(x,y) = a + \frac{\ln\left[f(x,y)+1\right]}{b\ln c} = a + d \cdot \ln\left[f(x,y)+1\right] \tag{2}$$



对数变化就是将原本的灰度域通过对数函数映射到新的灰度域中，以达到改善对比度的效果或增加整体亮度的效果。

三、实验步骤与分析

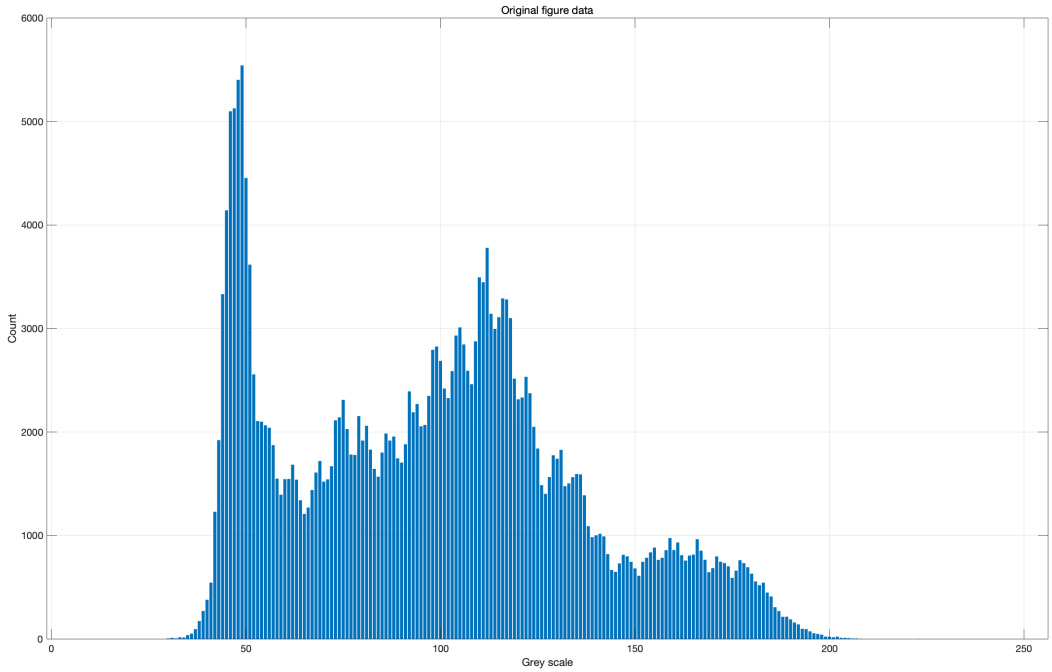
1、读入灰度图

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

typedef struct FILEHEADER{
    unsigned int   bfSize;         // Size of Figure
    unsigned short bfReserved1;    // Reserved1, must be 00
    unsigned short bfReserved2;    // Reserved2, must be 00
    unsigned int   bfOffBits;      // Offsets from header to info
}BMPFILEHEADER;

typedef struct INFOHEADER{
    unsigned int   biSize;         // Size of this structure
    unsigned int   biWidth;        // Width
    unsigned int   biHeight;       // Height
    unsigned short biPlanes;       // always is 1
    unsigned short biBitCount;     // Kind of BMP, bit/pixel = 1 4 8
16 24 32
    unsigned int   biCompression;
    unsigned int   biSizeImage;
    unsigned int   biXPelsPerMeter;
    unsigned int   biYPelsPerMeter;
    unsigned int   biClrUsed;
    unsigned int   biClrImportant;
}BMPINFOHEADER;

typedef struct YUVSPACE{
    unsigned char Y;
    unsigned char U;
    unsigned char V;
}YUV;

int main(){
    FILE *yuvfp,*histfp;

    unsigned short bftype;
    // head includes head information of BMP file.
    BMPFILEHEADER head;
    // info includes figure information of BMP file.
    BMPINFOHEADER info;

```

```
    // read F1.bmp
    if((yuvfp=fopen("RGB2YUV.bmp","rb")) == NULL){
        printf("OPEN FAILED!\n");
        return 0;
    }

    // bftype must be 0x4d42.
    fread(&bftype,1,sizeof(unsigned short),yuvfp);
    if(bftype != 0x4d42){
        printf("This figure is not BMP!\n");
        Sleep(1000);
        return 0;
    }

    // read HEADER
    fread(&head,1,sizeof(BMPFILEHEADER),yuvfp);
    fread(&info,1,sizeof(BMPINFOHEADER),yuvfp);

    // We takes 24 bitcount bmp for experiment item.
    if(info.biBitCount != 24){
        printf("This BMP is not 24 biBitCount!\n");
        Sleep(1000);
        return 0;
    }
////////////////////////////////////////////////////////////////////////
//////////////
    int size = info.biSizeImage / 3;

    // define array.
    YUV yuv[size];

    // read information.
    fread(yuv,size,sizeof(YUV),yuvfp);
    fclose(yuvfp);
```

2、计算P(r)

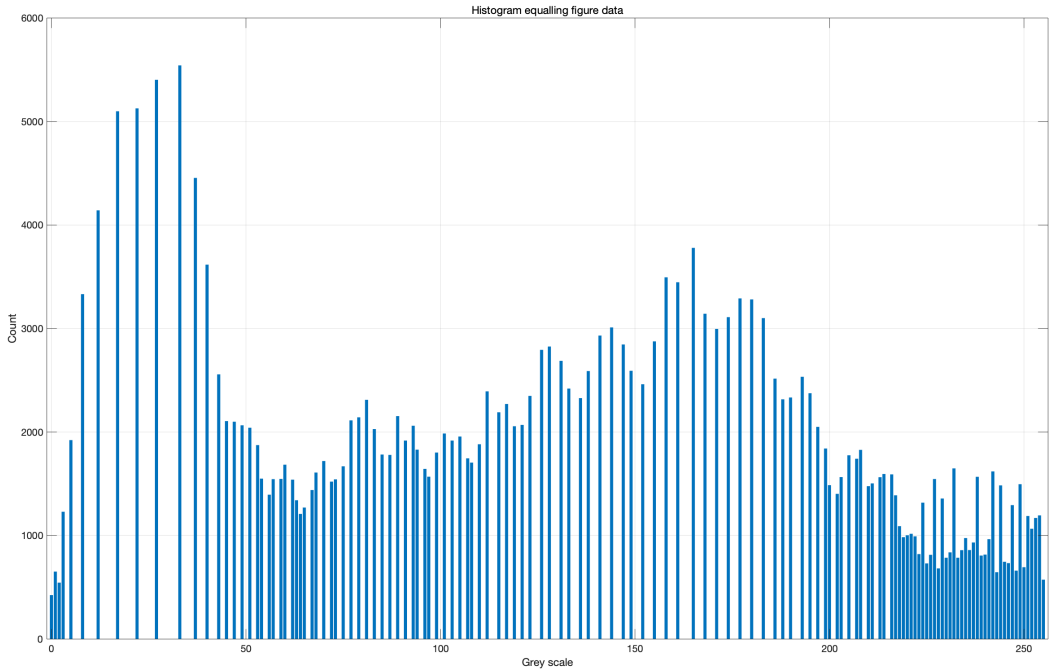
```
    int i;
    // histcount[] stores the count of grey scale in bmp.
    int histcount[256] = {0};
    // p_rk[] stores the possibility of each grey scale.
    float p_rk[256];

    // count the frequency.
    for(i=0;i<=size-1;i++)
        histcount[yuv[i].Y]++;

    // calculate the possiblity.
    for(i=0;i<=255;i++)
        p_rk[i] = (float)histcount[i] / (float)size;
```



- Lena图的原始直方图

3、转换映射

```
    // calculate the accumulative possibility.
    for(i=0;i<=255;i++){
        if(!i) sk[i] = p_rk[0];
        else sk[i] = sk[i-1] + p_rk[i];
    }

    // reshape sk[i] which belong to [0,1] into ssk[i] which belong to [0,255].
    for(i=0;i<=255;i++){
        ssk[i] = (int)(sk[i] * 255 + 0.5);
    }
```



- Lena图的直方图均衡化后的直方图

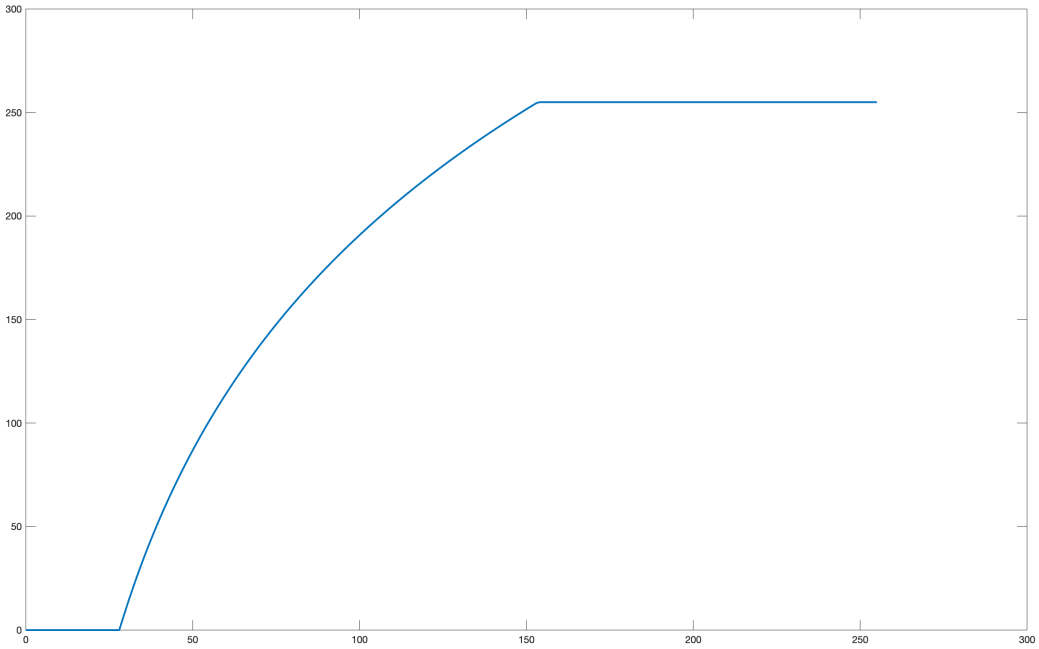
4、输出

```
////////////////////////////////////////////////////////
    // Output BMP.
    if((histfp = fopen("Histogram_equal.bmp","wb")) == NULL){
        printf("Create image FAILED!\n");
        Sleep(1000);
        return 0;
    }
```

```
8
9       // Write header information and data into file.
10      fwrite(&bftype,1,sizeof(unsigned short),histfp);
11      fwrite(&head,1,sizeof(BMPFILEHEADER),histfp);
12      fwrite(&info,1,sizeof(BMPINFOHEADER),histfp);
13
14      for(i=0;i<=size-1;i++){
15          fwrite(&ssk[yuv[i].Y],1,sizeof(unsigned char),histfp);
16          fwrite(&ssk[yuv[i].Y],1,sizeof(unsigned char),histfp);
17          fwrite(&ssk[yuv[i].Y],1,sizeof(unsigned char),histfp);
18      }
19
20      fclose(histfp);
21      printf("Output Histeq Image successfully!\n");
22  ////////////////////////////////////////////////////////////
```

5、对数变化的主要代码

```
1       YUV inyuv[size];
2       YUV logyuv[size];
3
4       double a = -500;
5
6       for(i=0;i<=size-1;i++){
7           logyuv[i].Y = cut((int)(-500 + log(inyuv[i].Y+1)*150));
8           logyuv[i].U = inyuv[i].U;
9           logyuv[i].V = inyuv[i].V;
10      }
```



四、实验环境及运行方法

本实验运行在MACOS系统下VM Ware软件下Windows 10虚拟机的Dev C++中。源代码可编译成功。

五、实验结果展示

1、原始图片

原始的Lena图整体偏暗，对比度不强，但是边缘柔和，灰度颜色过度自然。

2、Histogram equality

经过直方图均衡化后，可以发现对比度显著提升了，直方图均衡化兼并了一些灰度值并且将分布范围拉伸到整个灰度域中。可以观察五官部分，明显立体清晰了，帽子的装饰毛部分也更加清楚。缺点是，背景中的噪声点变多了，整体的柔和度下降了，这也是灰度兼并后，灰度域基减少的副作用。

3、Logarithm

对数变换的话，对图像的亮度提升较佳。既整体提升了亮度，同时也能较好的在像素之间有区分度，画面的柔和效果上比直方图均衡化更好，但对比区分度上不如直方图均衡化。

六、心得体会

1、讨论

（1）直方图均衡化的效果。直方图均衡化能有效增强图片的对比度，使图片更加清晰。但是也存在缺点：直方图均衡化能够兼并灰度值出现次数较少的一些像素点，但是对于灰度值出现次数很高的像素点并未做很多改变，即无法调整灰度出现次数很高但是却不包含有效信息的点（比如有很多相同灰度值的噪声点）。并且从简并后的直方图可以看出，灰度值的数量大幅减小，相应地，在图像上显示的平滑度、柔和度就下降了。

（2）对数化的最重要的好处是提升总体的亮度/降低总体亮度并且增加对比度。对数函数在像素灰度值较低处，是近似线性的，在灰度值较高的地方增长是很缓慢的，这样既保留了原图像的柔和特征，又能够增加总体的亮度值。但是参数的选择很困难，很容易出现过亮、过暗、对比度变差的问题。

2、心得

本次实验我们主要完成了直方图均衡化和对数变换，从效果上来说，两者都有优缺点，但也不可避免的说明了：图像的柔和关系和对比度是两个端点，在提升对比度的同时不可避免地要牺牲柔和度，反之亦然。实验还是很有意思的，特别是运用了概率论与数理统计的知识，有一种学科融会贯通的感觉。