专业：_自动化（电气）_
姓名：_____潘盛琪_____
学号：___3170105737___
日期：_____
地点：**生工食品学院机房**

浙江大学实验报告

课程名称：**计算机图像处理与机器视觉** 指导老师：**饶秀勤** 成绩：_____

实验名称：**直方图变换** 实验类型：**设计型**

一、实验目的和要求

1. 从图像灰度级的分布可以看出一幅图像的灰度分布特性。本次实验需要作出所给图像的直方图，并进行直方图均衡化处理等操作。
2. 实现卷积功能

二、计算机配置与软件处理平台

硬件：

Windows 版本

Windows 10 家庭中文版

© 2018 Microsoft Corporation。保留所有权利。

系统

处理器:	AMD Ryzen 7 PRO 2700U w/ Radeon Vega Mobile Gfx 2.20 GHz
已安装的内存(RAM):	8.00 GB (6.93 GB 可用)
系统类型:	64 位操作系统，基于 x64 的处理器
笔和触控:	没有可用于此显示器的笔或触控输入

软件：基于 matlab

三、算法描述

1. 直方图计算

具体算法：

1.获取彩色图像每个像素某一颜色分量的灰度值存入数组f(x, y)中；
2.获取图像的高度m(行)和宽度n（列）；
3.计算每级灰度的像素个数存入hd数组中,数组下标值为灰度值

```
for (j 从1 到 m)
    {for( i 从1 到 n
        {  k = f(i, j)
           hd(k) = hd(k) + 1
        }
     }
4.绘制直方图，每级灰度的像素个数用垂直线表示
   for( i 从 0 到 255)
      {从（i,hd(i)）到 （i,0）画线
  }
```

程序实现

2. 累计直方图

直方图均衡化处理是以**累积分布函数变换法**为基础的直方图修正法。假定变换函数为

$$s = T(r) = \int_0^r p_r(\omega)d\omega$$

S----累积分布函数

式中：ω是积分变量，而 $\int_0^r p_r(\omega)d\omega$就是$r$的累积分布函数。

这里，**累积分布函数是 *r* 的函数**，并且**单调**地从0增加到255，所以这个变换函数满足关于 $T(r)$ 在$0 \le r \le 255$内单值单调增加。在$0 \le r \le 255$内有$0 \le T(r) \le 255$的两个条件。

3. 直方图均衡化处理

变换函数 $s = T(r) = \int_0^r p_r(\omega)d\omega$

对式中的r求导，则 $\dfrac{ds}{dr} = p_r(r)$

再把结果代入下式有

$$p_s(s) = \left[p_r(r) \cdot \frac{dr}{ds} \right]_{r=T^{-1}(s)} = \left[p_r(r) \cdot \frac{1}{ds/dr} \right]_{r=T^{-1}(s)}$$

$$= \left[p_r(r) \cdot \frac{1}{p_r(r)} \right] = 1$$

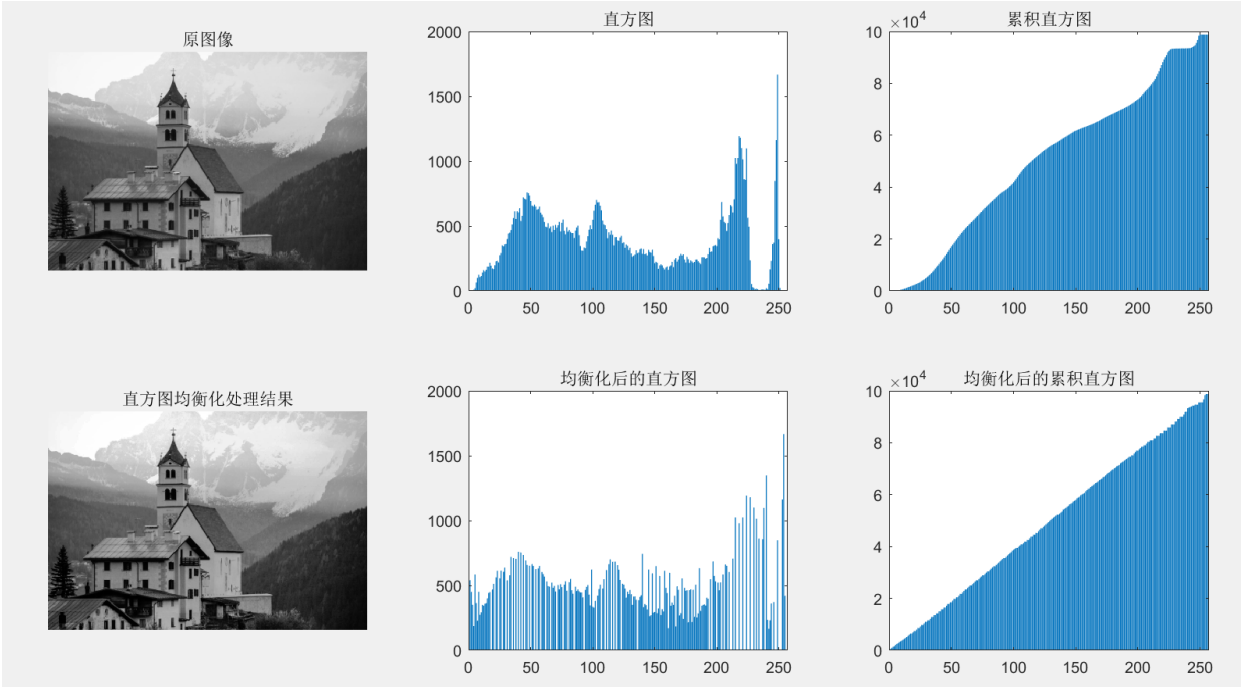
由上面的推导可见，在变换后的变量 s 的定义域内的概率密度是均匀分布的。**因此，用*r* 的累积分布函数作为变换函数，可产生一幅灰度级分布具有均匀概率密度的图像。**

```
For j = 0 To m- 1
   For i = 0 To n - 1
     k = f (i, j):  hd(k) = hd(k) + 1
   Next i
 Next j        '提取各像素的灰度值，并统计该灰度值的像素数

For i = 0 To 255
    p(i) = hd(i) / (m *n)   '计算该灰度级频率
    For j = 0 To i
       q(i) = q(i) + p(j)     '累计该灰度级前所有的灰度级的频率值
    Next j
       q(i) = Int(q(i) * 100) / 100   取整
 Next I

For j = 0 To m - 1
   For i = 0 To n – 1
      k = q(f(i, j)) * 255: hd1(k) = hd1(k) + 1 '取变换后的新灰度
  Next i
Next j
```

四、结果与讨论

4.1 输入图像

4.2 输出图像



4.3 讨论

可以看到直方图均衡化使图像的灰度分布更均匀，这一点从直方图均衡化后的累计直方图可以直观地看出来。而从图像的角度而言，直方图均衡化使图像的对比度得到了明显的提升

五、结论

灰度直方图是灰度级的函数，它表示图像中具有某种灰度级的像素的个数。直方图均衡化处理可以使灰度图的直方图分布更均匀，

六、源程序

1. 直方图均衡化

```
%%%%%%%%%%%
%直方图显示%
%%%%%%%%%%%
img = imread('test.png');
img = rgb2gray(img);
[row, col] = size(img);%确定图像大小
hd = zeros(1,256);
for i = 1 : row
   for j = 1 : col
       k = img(i,j);%遍历每一个像素的灰度值
       hd(k) = hd(k) + 1;%hd(k)对应灰度为 k 的像素的个数
   end
end
%画图
subplot(2,3,1);
imshow(img);
title('原图像');
subplot(2,3,2);
grayscale = 1:256;
bar(grayscale, hd);
title('直方图');

%%%%%%%%%%%%%%
%画出累积直方图%
%%%%%%%%%%%%%%
leiji = hd;
for i = 2 :256
    leiji(i) = leiji(i) + leiji(i-1);
end
%画图
subplot(2,3,3);
bar(grayscale, leiji);
title('累积直方图');
```

```
%%%%%%%%%%%%%%%%%
%直方图均衡化处理%
%%%%%%%%%%%%%%%%%
bamp = zeros(1,256);
for i=1:256
    temp=0;
    for j=1:i
        temp=temp+hd(j);
    end
    bmap(i)=floor(temp*255/(row*col));
end
y=zeros(row,col);

for i=1:row
    for j=1:col
        y(i,j)=bmap(img(i,j)+1);
    end
end
y=uint8(y);
subplot(2,3,4);
imshow(y);
title('直方图均衡化处理结果')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%均衡化后的直方图%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[row, col] = size(y);%确定图像大小
hd = zeros(1,256);
for i = 1 : row
   for j = 1 : col
       k = y(i,j);%遍历每一个像素的灰度值
       if k == 0
           k = 1;
       end
       hd(k) = hd(k) + 1;%hd(k)对应灰度为 k 的像素的个数
   end
end

subplot(2,3,5);
bar(grayscale, hd);
title('均衡化后的直方图');

%%%%%%%%%%%%%%%%%%%%%%%%
%画出均衡化后的累积直方图%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%
leiji = hd;
for i = 2 :256
    leiji(i) = leiji(i) + leiji(i-1);
end
%画图
subplot(2,3,6);
bar(grayscale, leiji);
title('均衡化后的累积直方图');
```

2. 卷积（写成了函数）

```
%%%卷积函数，传入的第一个参数为输入图像，第二个参数为卷积核
%%%输出卷积结果
%%%对于图像边缘区域不做处理，也就是说输出图像会减小
function imgout = convolve(imgin, kernel)
    [row, col] = size( imgin );
    [~, kernelsize] = size( kernel );
    r = (kernelsize - 1) / 2;
    rowout = row - 2 * r;  %输出图像大小
    colout = col - 2 * r;  %输出图像大小
    if rowout <= 0 || colout <= 0
        return;
    end
    imgout = zeros(rowout, colout);%初始化输出数组
    for i = 1 : rowout
        for j = 1 : colout
            imgout(i, j) =  max(0, sum(sum(imgin(i : i + 2 * r, j : j + 2 *
r).*(kernel))));
        end
    end
end
```

3. 锐化（调用上述的卷积函数）

```
%%%%卷积操作%%%%
img = imread('test.png');
img = rgb2gray(img);
img = double(img);
kernel1 = [-1, 0, 1; -2, 0 ,2; -1, 0, 1];%Sobel 算子
kernel2 = kernel1';
imgout1 = convolve(img, kernel1);
imgout2 = convolve(img, kernel2);
imgout = (imgout1.*imgout2).^0.5;


subplot(2,1,1);
imshow(uint8(img));
```

```
title('原图像');
subplot(2,1,2);
imshow(uint8(imgout));
title('Sobel 算子处理后结果');
```