**实验报告**

装 订 线

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**

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**

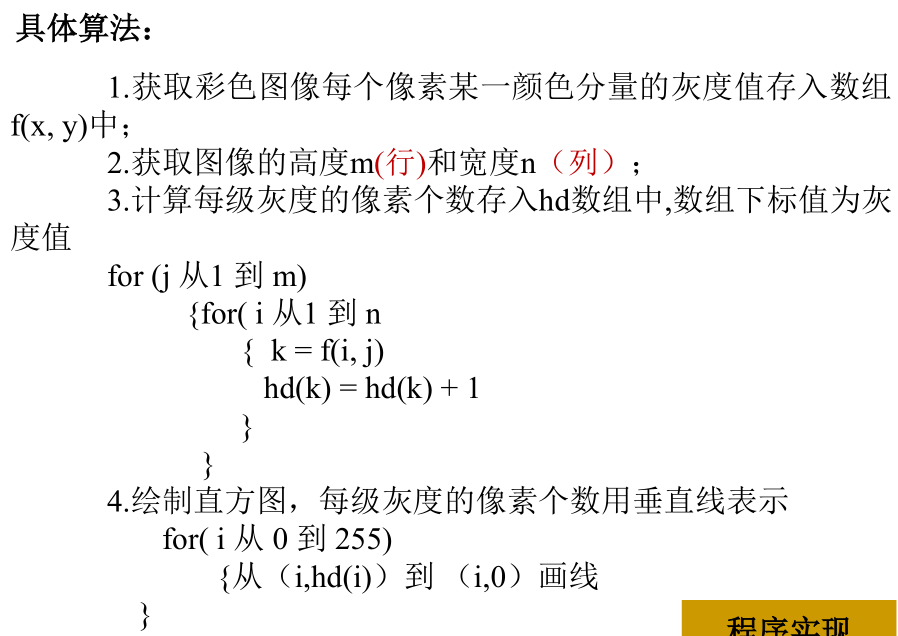
硬件：



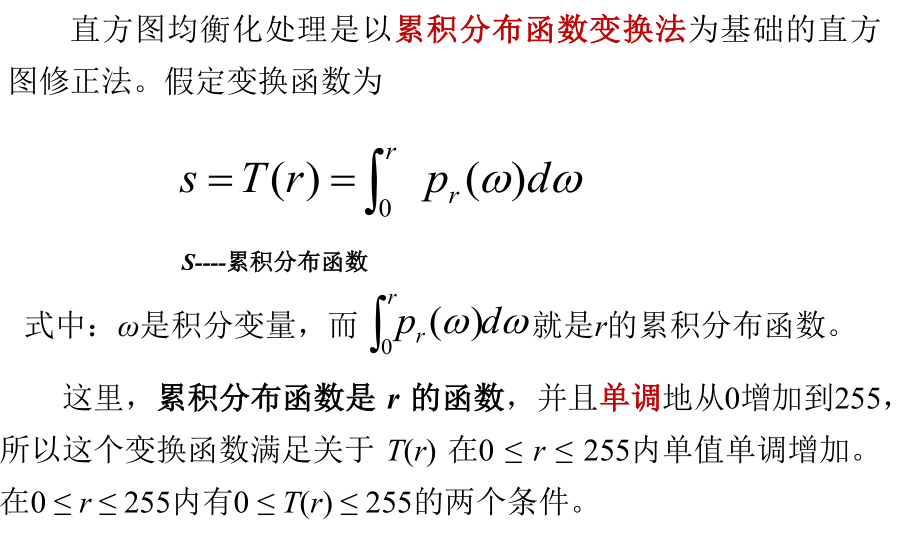
软件：基于matlab

**三、算法描述**

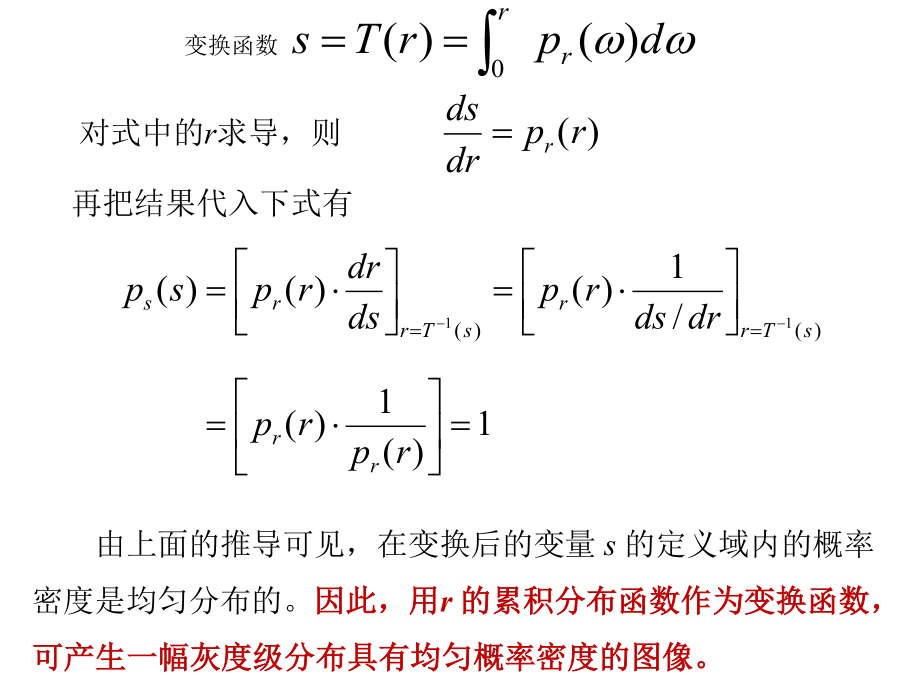
1. 直方图计算

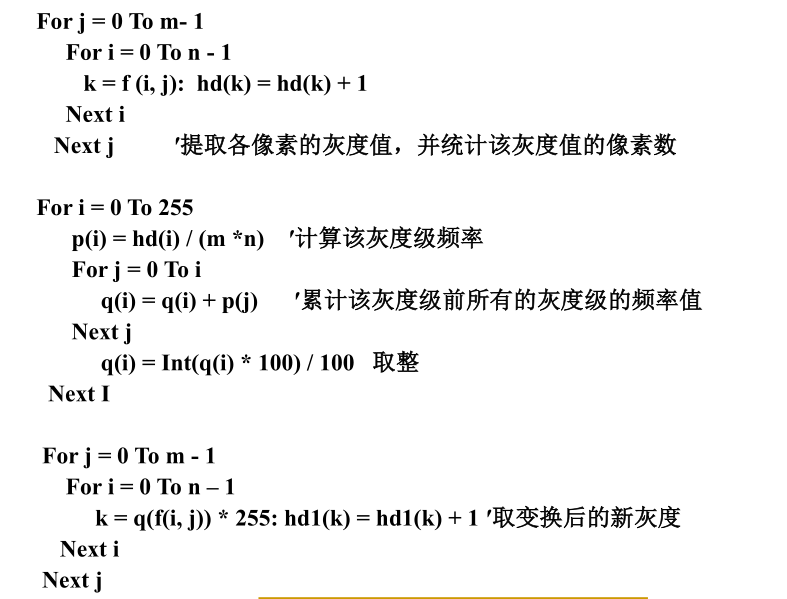


1. 累计直方图



1. 直方图均衡化处理



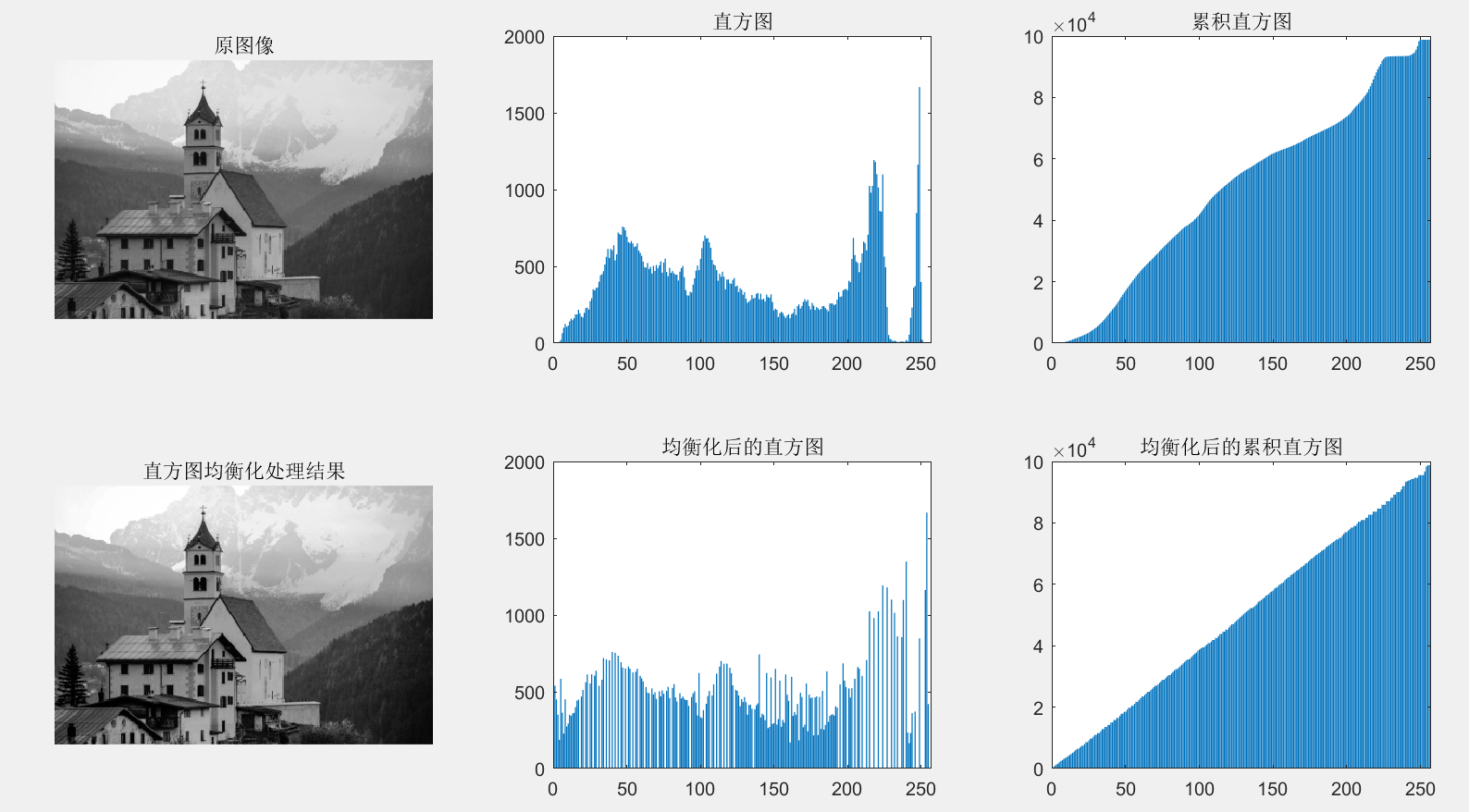


**四、结果与讨论**

**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');

1. 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

1. 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');