華中科技大學

数字图像处理作业报告

平滑处理
直方图均衡化
同态滤波

院　　系：人工智能与自动化学院

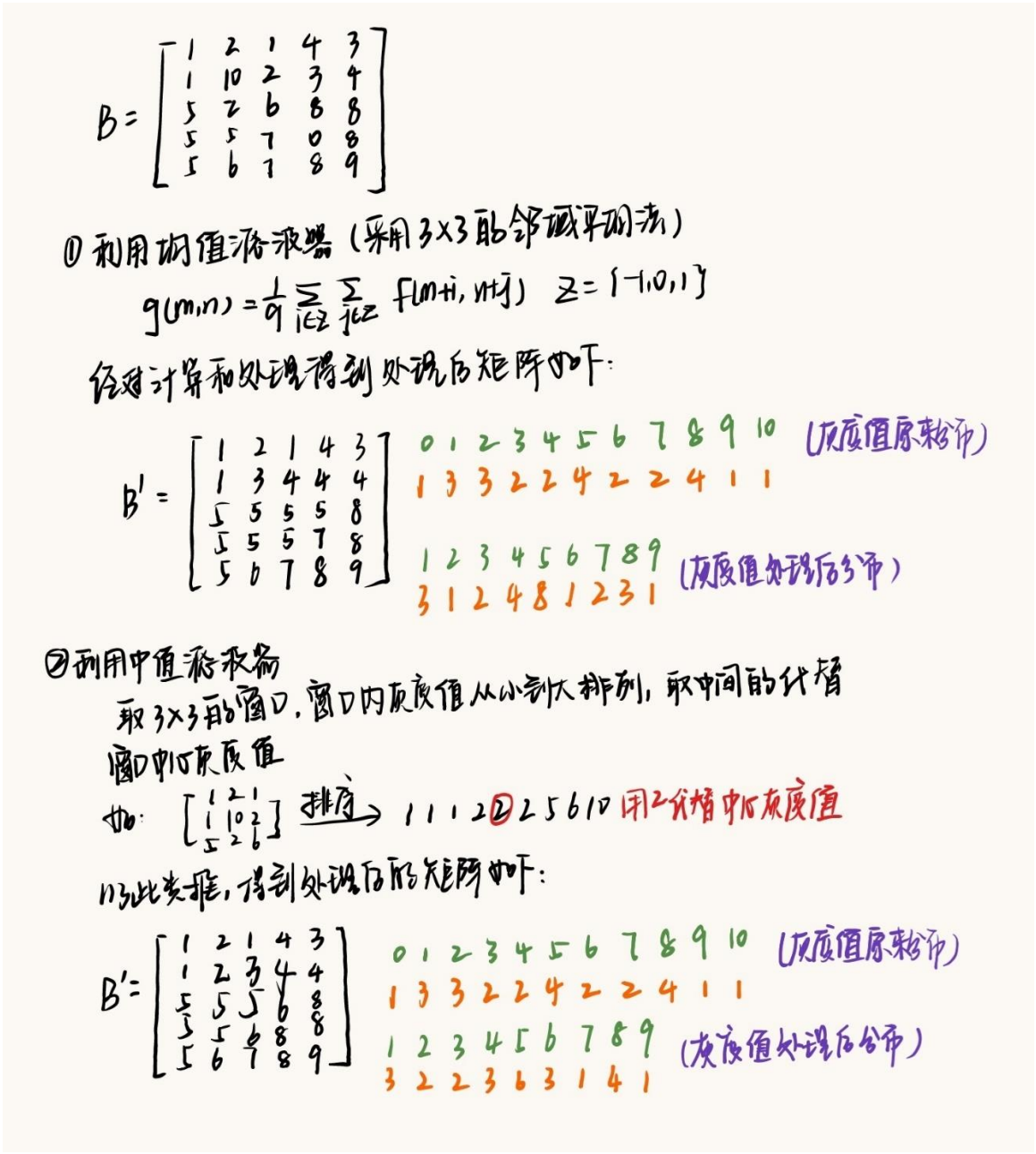
班　　级：人工智能 2204 班

姓　　名：陆慧敏

学　　号：U202215199

1. 理论作业

使用均值滤波器和中值滤波器对图像进行平滑处理。



2. 编程作业

1. 对低照度图像进行灰度化，计算并显示以上低照度图像的灰度直方图和离散傅里叶变换

频谱幅度图；

2. 对低照度图像分别进行直方图均衡化和同态滤波操作，并对两种算法的最终效果进行对比。

2.1 灰度直方图和离散傅里叶变换频谱幅度图

首先，使用 MATLAB 中的 imread 函数读取 RGB 彩色图片。接着，使用 imhist 函数计算图像灰度直方图并予以显示，然后使用 fft2 函数计算图像的二维傅里叶变换并使用 fftshift 函数进行频谱中心化，将最终的频谱图显示。

灰度图、灰度直方图和离散傅里叶变换频谱幅度图结果如下：

图 1 灰度图

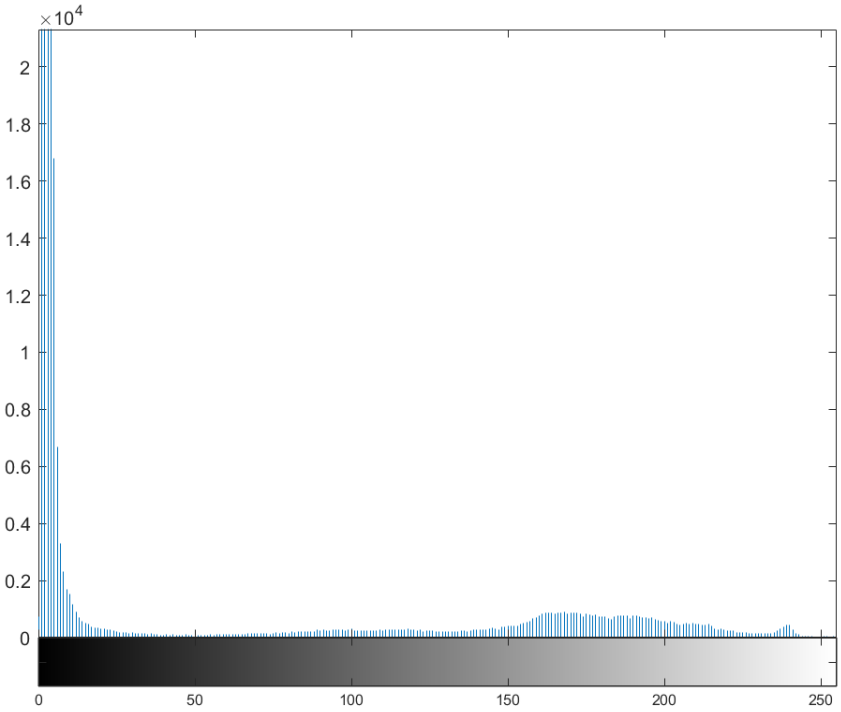


图 2 灰度直方图

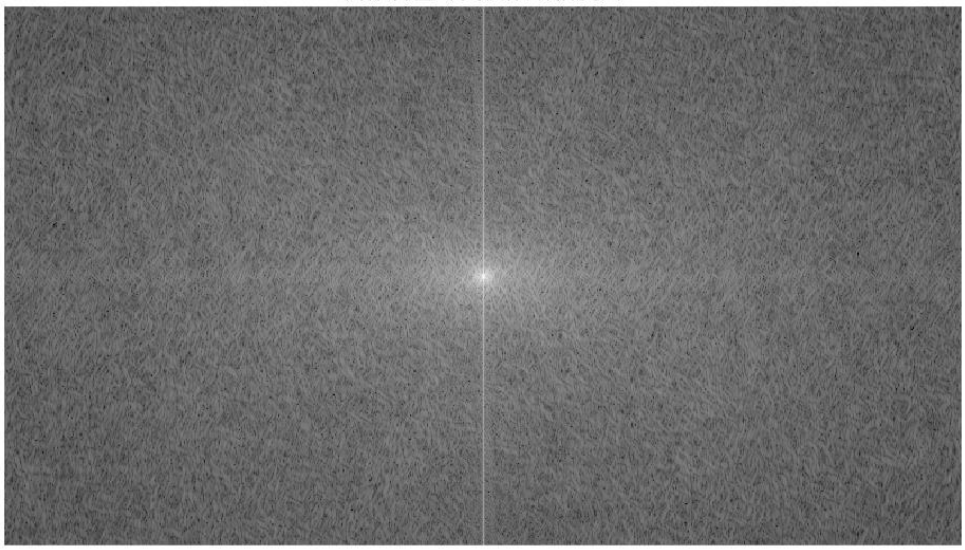


图 3 离散傅里叶变换频谱幅度图

2.2 直方图均衡化

要求不使用内置函数完成直方图均衡化。在前面已经得到原始图像的灰度图，创建一个 256×1 大小的矩阵用来存放不同灰度级对应图像中像素的个数，使用双层 for 循环遍历

整个图像得到不同灰度级像素个数。接着使用 cumsum 函数来计算累积和，该函数时计算从数组起始位置到当前元素的累积和，累积和之后进行归一化操作，将累积和除以图像像素点总个数。然后进行均衡化直方图的映射，将归一化后的矩阵乘以 255 进行四舍五入便可以得到[0，255]的灰度级，将图像中原灰度级替换成新的灰度级。

经过上述这些步骤可以得到直方图均衡化后的图像和均衡化后灰度图像直方图结果如下：

图 4 直方图均衡化后的图像（不使用内置函数）

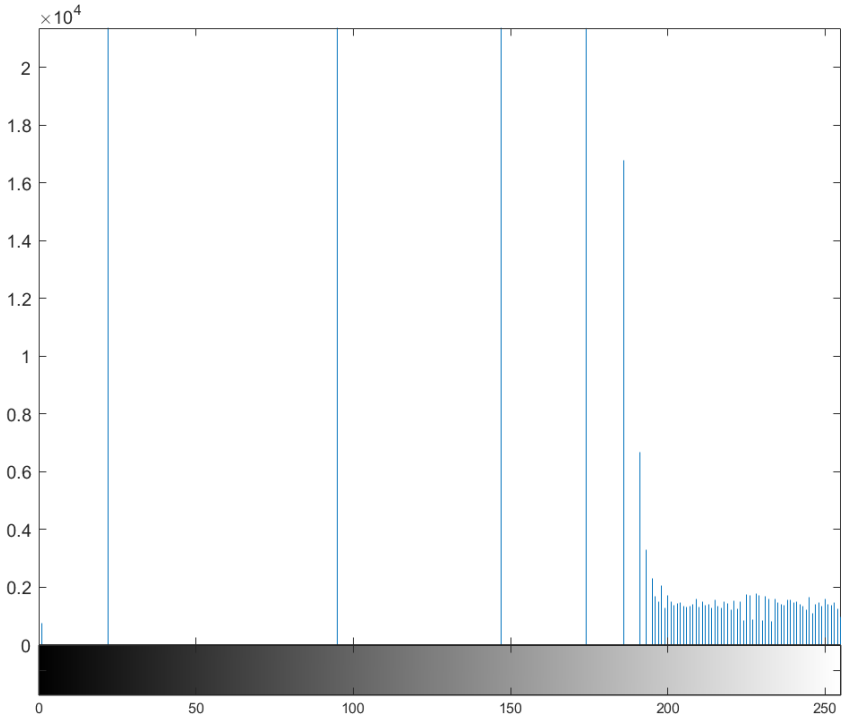


图 5 均衡化后灰度图像直方图（不使用内置函数）

使用 histeq 内置函数可以得到如下的直方图均衡化后的图像和灰度直方图：

图 6 直方图均衡化后的图像（使用内置函数）

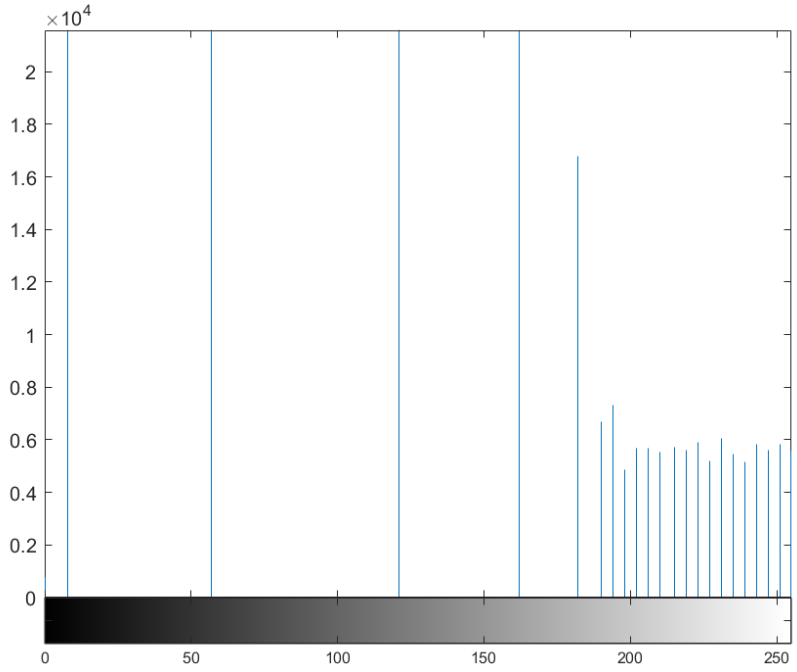


图 7 均衡化后灰度图像直方图（使用内置函数）

从最后的效果图可以看出使用内置函数得到的图像要比自己写的函数得到的图像要偏暗一些，从灰度直方图可以看出使用内置函数可以看出对一些灰度级进行了合并，可能是由于内置函数进行了对直方图均衡化进行了优化。

2.3 同态滤波

首先，设置相关参数控制低频增益和高频增益以及截止频率。接着，对灰度图像进行取对数操作，将图像的乘法性噪声转换为加法性噪声，再对图像进行二维快速傅里叶变换操作。然后用上面设置的三个参数设置同态滤波函数，常见的一种同态滤波函数如下图所示，将傅里叶变换后的图像和同态滤波函数相乘，再进行傅里叶逆变换取指数。经过上述处理后需要将处理后的值映射到[0，255]的范围内，先计算一个 range 值，即处理后数据的最大值减去最小值，然后对这些值基于 range 进行归一化处理再乘以 255 进行映射，从而得到同态滤波后的图像。

经过同态滤波后的图像和灰度直方图如下图所示：

图 8 同态滤波后的图像

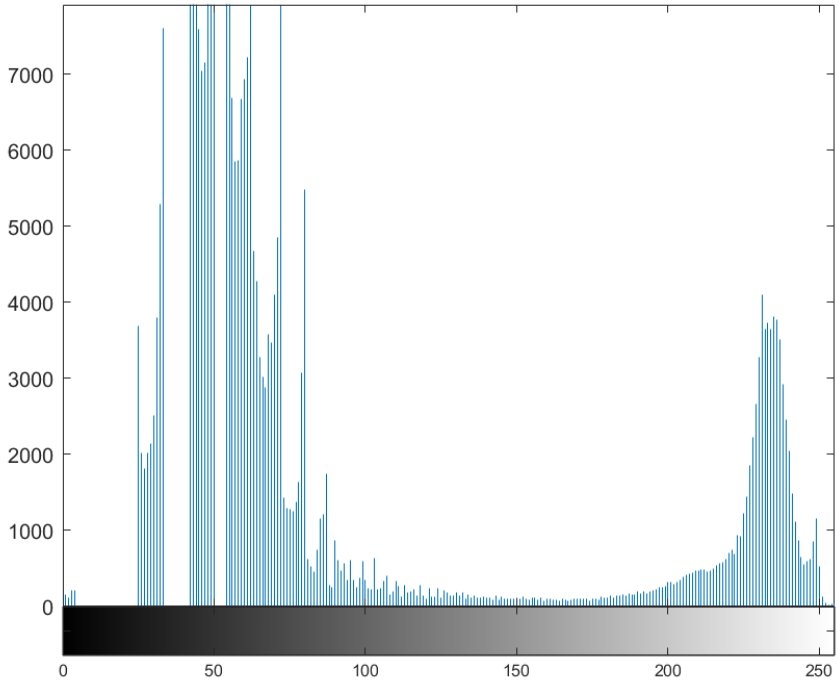


图 9 同态滤波后的灰度直方图

2.4 两种算法比较

从两种算法的最终图像效果图可以看出直方图均衡化能够使图像的亮度更亮但同时也

放大了噪声，使得图像成像不太自然，从图中可以看出直方图均衡化使得边缘信息有丢失，导致边缘处理较为粗糙。

相比之下同态滤波的效果要比直方图均衡化好很多，同态滤波通过将图像的照射分量和反射分量分离来增强图像，同态滤波在增加图像亮度的同时也增加了图像的对比度，相对减少了噪声，可以更好的保留图像的细节。但缺点是同态滤波需要自己调节参数，不能自适应，而直方图均衡化无需调参可以节省时间，并且同态滤波涉及到了傅里叶变换和傅里叶反变换，需要更多的计算资源。

3. 心得体会

在完成均值滤波器、中值滤波器的理论学习以及直方图均衡化和同态滤波的编程实践后，我深刻体会到了图像处理领域中算法设计与应用的精妙。完成均值滤波器的作业，我明白了其背后的数学原理，即通过邻域平均来减少随机噪声的影响。

中值滤波器则在处理椒盐噪声方面表现出色，它通过替换像素值为邻域内的中值，有效地保留了边缘信息。这种非线性的滤波方式让我认识到，在处理特定类型的噪声时，非线性方法往往比线性方法更为有效。

在编程实践中，直方图均衡化让我见识到了如何通过调整图像的灰度分布来增强图像对比度。这种方法直观且易于实现，但同时也让我意识到了它在边缘处理方面的局限性。

同态滤波则是一种更为复杂的技术，它通过在频域中对图像的照射和反射分量进行分离和增强，有效地改善了图像的光照条件。编程实现这一算法的过程中，我学会了如何运用傅里叶变换和对数变换等数学工具，这不仅锻炼了我的编程能力，也加深了我对图像形成过程的理解。

总的来说，这次作业不仅让我掌握了图像处理的基本技能，还激发了我对图像分析和处理技术深层次探索的兴趣。

附　件

1、直方图均衡化和同态滤波程序（不使用内置函数）

```
% 读取图像
img = imread('C:\Users\28323\Desktop\低照度图像.jpg');


% 灰度化处理
grayImg = rgb2gray(img);


% 显示灰度图像
figure;
imshow(grayImg);
title('灰度图像');


% 计算并显示灰度直方图
figure;
imhist(grayImg);
title('灰度直方图');


% 计算离散傅里叶变换
dft = fft2(double(grayImg));
dftShifted = fftshift(dft); % 频谱中心化


% 计算频谱幅度
magnitudeSpectrum = log(1 + abs(dftShifted));


% 显示频谱幅度图
figure;
imshow(magnitudeSpectrum, []);
title('离散傅里叶变换频谱幅度图');
% 获取图像尺寸
[m, n] = size(grayImg);


% 计算直方图（每个灰度级对应像素个数）
```

```
histogram = zeros(256, 1);
for i = 1:m
    for j = 1:n
        pixelValue = grayImg(i, j);
        histogram(pixelValue + 1) = histogram(pixelValue + 1) + 1;
    end
end


% 计算累积分布函数 (CDF)
cdf = cumsum(histogram);
cdf = cdf / (m * n); % 归一化


% 直方图均衡化映射
equalizedImg = zeros(m, n);
for i = 1:m
    for j = 1:n
        % 映射到新值
        equalizedImg(i, j) = round(cdf(grayImg(i, j) +1) * 255);%四舍五入
    end
end


% 转换为 uint8 类型
equalizedImg = uint8(equalizedImg);


%显示直方图均衡化后的图像
figure;
imshow(equalizedImg);
title('直方图均衡化后的图像');


%显示均衡化后的灰度直方图
figure;
imhist(equalizedImg);
title('均衡化后灰度图像直方图');
%参数声明
rH = 0.04;
rL = 0.02;
c = 0.03;%介于 rH 和 rL 之间
```

```
D0 = 0.05;


%取对数
img_log = log(double(grayImg) + 1);


%平移到中心，判断语句代替指数计算
img_py = zeros(M, N);
for i = 1:M
   for j= 1:N
       if mod(i+j, 2) == 0
           img_py(i,j) = img_log(i, j);
       else
           img_py(i,j) = -1 * img_log(i, j);
       end
   end
end


% 对填充后的图像进行傅里叶变换
img_py_fft = fft2(img_py);


%同态滤波函数
img_tt = zeros(M, N);
deta_r = rH - rL;
D = D0^2;
m_mid=floor(M/2);%中心点坐标
n_mid=floor(N/2);


for i = 1:M
   for j =1:N
        dis = ((i-m_mid)^2+(j-n_mid)^2);
        img_tt(i, j) = deta_r * (1-exp((-c)*(dis/D))) + rL;
   end
end


%滤波
img_temp =   img_py_fft.*img_tt;
```

```
%反变换,取实部，绝对值
img_temp = abs(real(ifft2(img_temp)));


%指数化
img_temp = exp(img_temp) - 1;




%归一化处理
max_num = max(img_temp(:));
min_num = min(img_temp(:));
range = max_num - min_num;
img_after = zeros(M,N,'uint8');


for i = 1 : M
    for j = 1 : N
        img_after(i,j) = uint8(255 * (img_temp(i, j)-min_num) / range);
    end
end


figure;
imshow(img_after);
title('同态滤波处理后图像');


figure;
imhist(img_after);
title('同态滤波处理后灰度直方图');
```

2、直方图均衡化程序（使用内置函数）

```
% 读取图像
img = imread('C:\Users\28323\Desktop\低照度图像.jpg');


% 灰度化处理
grayImg = rgb2gray(img);
```

```
% 直方图均衡化
equalizedImg = histeq(grayImg);


% 显示原始灰度图像和均衡化后的图像
figure;
imshow(grayImg);
title('原始灰度图像');


imshow(equalizedImg);
title('直方图均衡化后的图像');


% 显示原始和均衡化图像的灰度直方图
figure;
imhist(grayImg);
title('原始灰度图像直方图');




imhist(equalizedImg);
title('均衡化后灰度图像直方图');
```

电子签名：