刘一诺

第二次作业及实验报告

人工智能 2204 U202213275 刘一诺

1. 分别用 3×3 大小的均值滤波器和中值滤波器对图像进行平滑处理，给出处理步骤及结果图像。（注：图像边框像素保留不变）

$$B=\begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 10 & 2 & 3 & 4 \\ 5 & 2 & 6 & 8 & 8 \\ 5 & 5 & 7 & 0 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$ 3×3 的均值和中值滤波器

均值：$B'=$

$$\begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 3.33 & 4.22 & 4.33 & 4 \\ 5 & 4.78 & 4.78 & 5.11 & 8 \\ 5 & 5.33 & 5.44 & 6.78 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

$$g(m,n)=\frac{1}{9}\sum_{i\in Z}\sum_{j\in Z} f(m+i,n+j) \quad Z=\{-1,0,1\}$$

中值：$B''=$

$$\begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 2 & 2 & 4 & 4 \\ 5 & 5 & 5 & 5 & 8 \\ 5 & 5 & 7 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

① 1 1 1 2 (2) 2 5 6 10

② 1 2 2 2 (2) 3 4 6 8

③ 1 2 3 3 (4) 4 6 8 8

④ 1 2 2 2 (5) 5 5 6 7

⑤ 0 2 2 4 (5) 5 6 7 8

⑥ 0 2 4 4 (5) 7 8 8 8

⑦ 5 5 5 5 (5) 5 6 7 7

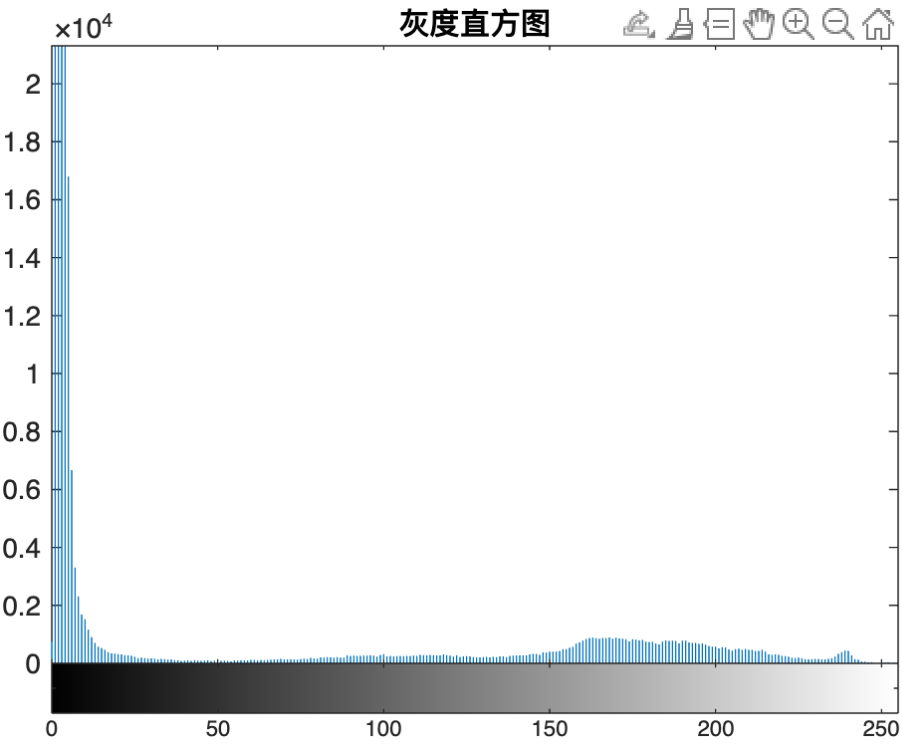
⑧ 0 5 5 5 (5) 6 7 7 8

⑨ 0 5 5 7 (7) 8 8 8 9

2.1对上述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。

```
filepath = '/MATLAB Drive/低照度图像.jpg';
image = imread(filepath);
% 将图像转换为灰度图像
grayImage = rgb2gray(image);
imshow(grayImage);
```

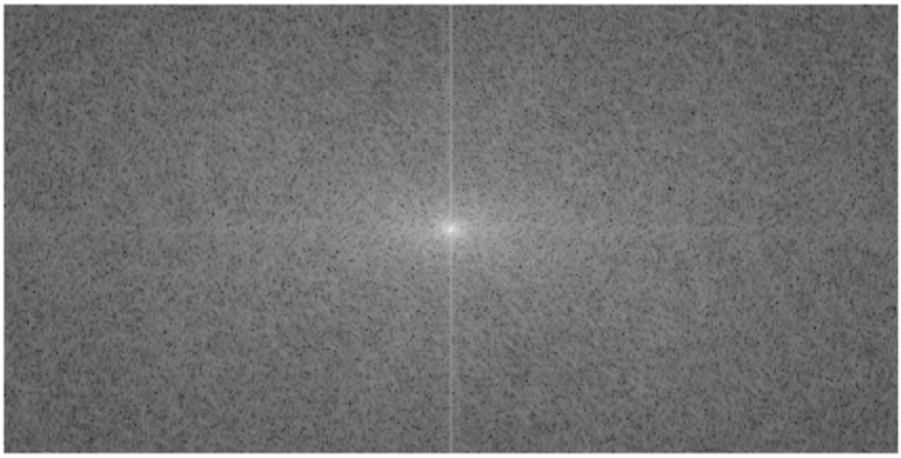

```
% 灰度直方图函数
function customGrayHist(grayImage)
    % 初始化一个 256 个元素的数组，用于统计每个灰度值的出现次数
    histogram = zeros(1, 256);
    % 获取图像的行数和列数
    [rows, cols] = size(grayImage);
    % 遍历图像中的每一个像素，统计灰度值出现的次数
    for r = 1:rows
        for c = 1:cols
            pixelValue = grayImage(r, c);  % 获取当前像素的灰度值
            histogram(pixelValue + 1) = histogram(pixelValue + 1) + 1; % 灰度值从 0 到 255，
但索引从 1 开始
        end
    end
    bar(0:255, histogram, 'BarWidth', 1, 'FaceColor', [0.2 0.2 0.5]);
    xlim([0 255]);
    xlabel('灰度值');
    ylabel('频率');
    title('灰度直方图');
end
```




```
% 快速傅立叶变换函数
function F = myFFT1D(x)
    N = length(x);
    % 检查是否为 2 的幂，不是则补齐
    if mod(N, 2) ~= 0
        % 零填充到最近的 2 的幂
        nextPow2 = 2^nextpow2(N);
        x = [x, zeros(1, nextPow2 - N)];
        N = length(x);
    end
    if N <= 1
        F = x;
    else
        % 递归计算偶数和奇数项的 FFT
        even = myFFT1D(x(1:2:end));  % 偶数项
        odd = myFFT1D(x(2:2:end));   % 奇数项
        % 组合结果
        F = zeros(1, N);
        for k = 1:N/2
            t = exp(-2i * pi * (k-1) / N) * odd(k);  % 旋转因子
            F(k) = even(k) + t;
            F(k + N/2) = even(k) - t;
        end
    end
end
```

```
function F = myfft2(image)
    [M, N] = size(image);
    M_padded = 2^nextpow2(M);  % 行数补齐到 2 的幂次
    N_padded = 2^nextpow2(N);  % 列数补齐到 2 的幂次
    padded_image = zeros(M_padded, N_padded);
    padded_image(1:M, 1:N) = image;
    % 初始化结果矩阵
    F = zeros(M_padded, N_padded);
    % 对每一行应用 1D FFT
    for i = 1:M_padded
        F(i, :) = myFFT1D(padded_image(i, :));
    end
    % 对每一列应用 1D FFT
    for j = 1:N_padded
        F(:, j) = myFFT1D(F(:, j).');  % 转置处理
    end
end
```

傅里叶变换的幅值谱（频率中心移动到图像中心）



2.2 对以上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；

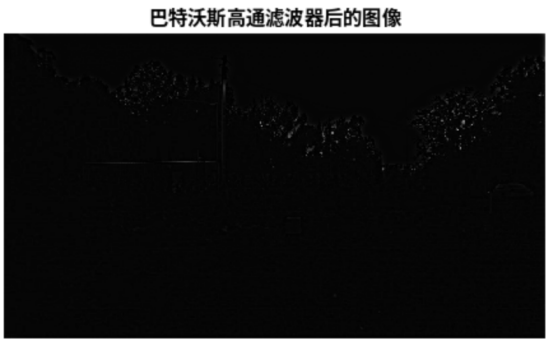
```
% 直方图均衡化
equalizedImage = customHistEqualization(grayImage);
figure;
imshow(equalizedImage);
title('直方图均衡化后的图像');

function equalizedImage = customHistEqualization(grayImage)
    % 手动实现直方图均衡化
    % grayImage: 输入的灰度图像
    [rows, cols] = size(grayImage);
    totalPixels = rows * cols;
```

```
    % 1. 计算灰度直方图
    histogram = zeros(1, 256);
    for r = 1:rows
        for c = 1:cols
            pixelValue = grayImage(r, c);
            histogram(pixelValue + 1) = histogram(pixelValue + 1) + 1;  % 索引从 1 开始
        end
    end
    % 2. 计算累积分布函数 (CDF)
    cdf = zeros(1, 256);
    cdf(1) = histogram(1);
    for i = 2:256
        cdf(i) = cdf(i-1) + histogram(i);
    end
    % 3. 归一化 CDF
    cdf_min = cdf(1);  % 最小的非零 CDF 值
    cdf_normalized = round((cdf - cdf_min) / (totalPixels - cdf_min) * 255);
    % 4. 将原图像中的每个像素值映射到新的灰度值
    equalizedImage = zeros(rows, cols, 'uint8');
    for r = 1:rows
        for c = 1:cols
            equalizedImage(r, c) = cdf_normalized(grayImage(r, c) + 1);
        end
    end
end
```

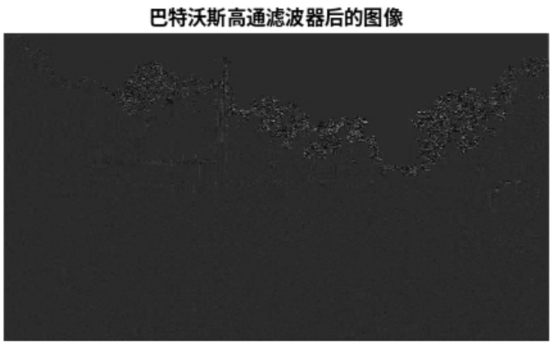
直方图均衡化后的图像

```
% 同态滤波
filepath = '/MATLAB Drive/低照度图像.jpg';
image = imread(filepath);
```

```
% 转换为灰度图像
grayImage = rgb2gray(image);

% 同态滤波操作
logImage = log(double(grayImage) + 1);
fftImage = fft2(logImage);
fftImageShifted = fftshift(fftImage);

% 构建巴特沃斯高通滤波器
[M, N] = size(grayImage);
D0 = 1000;  % 截止频率
n = 2;    % 巴特沃斯阶数
[X, Y] = meshgrid(1:N, 1:M);
centerX = ceil(N/2);
centerY = ceil(M/2);
D = sqrt((X - centerX).^2 + (Y - centerY).^2);
H = 1 ./ (1 + (D0 ./ D).^(2*n));  % 巴特沃斯高通滤波器

% 应用巴特沃斯高通滤波器
filteredFFT = H .* fftImageShifted;

% 傅里叶逆变换
filteredFFTShifted = ifftshift(filteredFFT);
filteredImage = real(ifft2(filteredFFTShifted));

% 转回线性域并进行拉伸
homomorphicImage = exp(filteredImage) - 1;

% 手动对像素值进行拉伸，归一化到 0-255
minValue = min(homomorphicImage(:));
maxValue = max(homomorphicImage(:));
homomorphicImage = (homomorphicImage - minValue) / (maxValue - minValue);
homomorphicImage = uint8(255 * homomorphicImage);

% 显示调整后的同态滤波图像
figure;
imshow(homomorphicImage);
title('巴特沃斯高通滤波器后的图像');
```



(1)D0=30



(2)D0=300