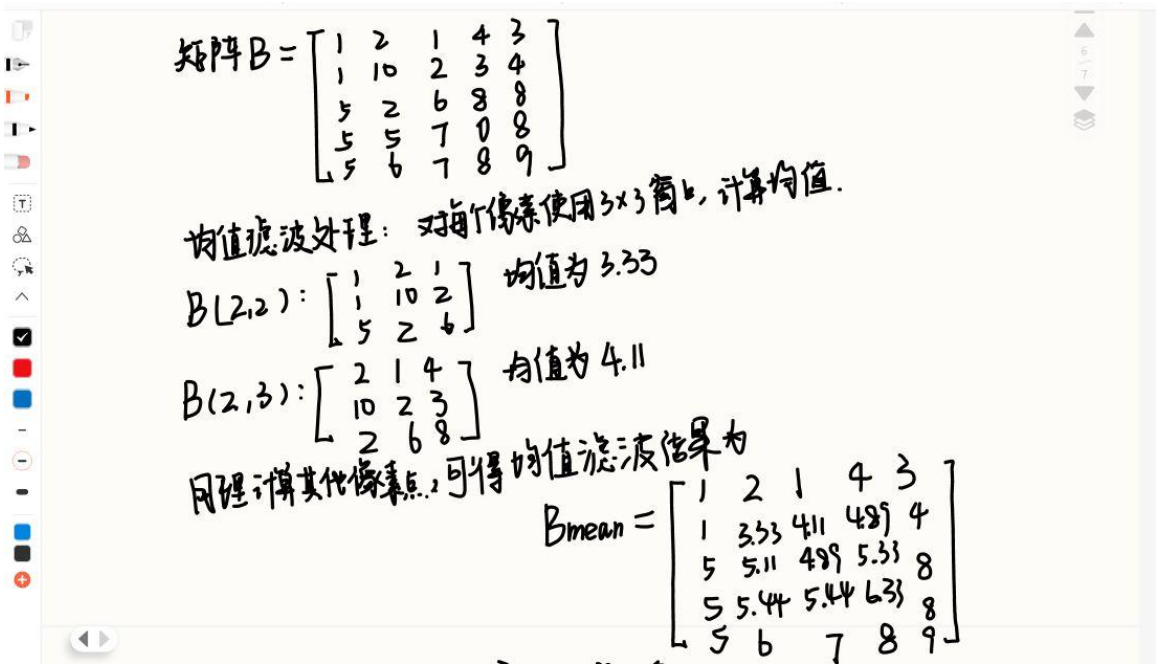
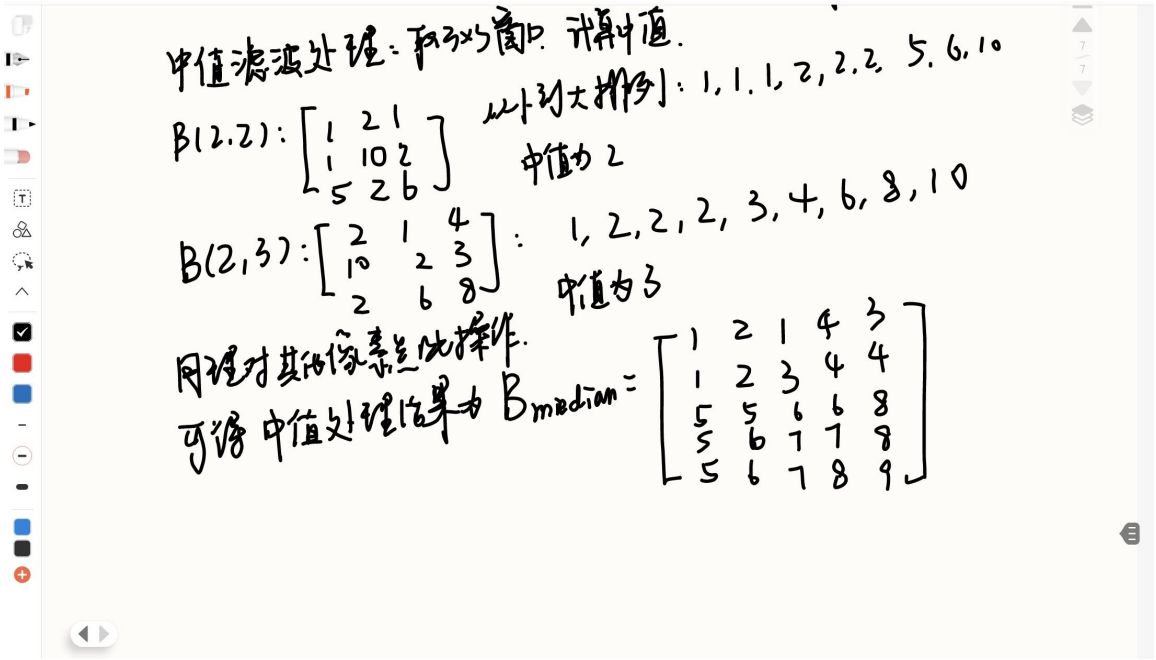
一、





二、编程作业

1.代码部分

```
%读取低照度图像
img = imread('D:\picture2.jpg'); %  替换为你的图像路径

%将图像转换为灰度图像
grayImg = custom_rgb2gray(img); %  自定义灰度转换函数
```

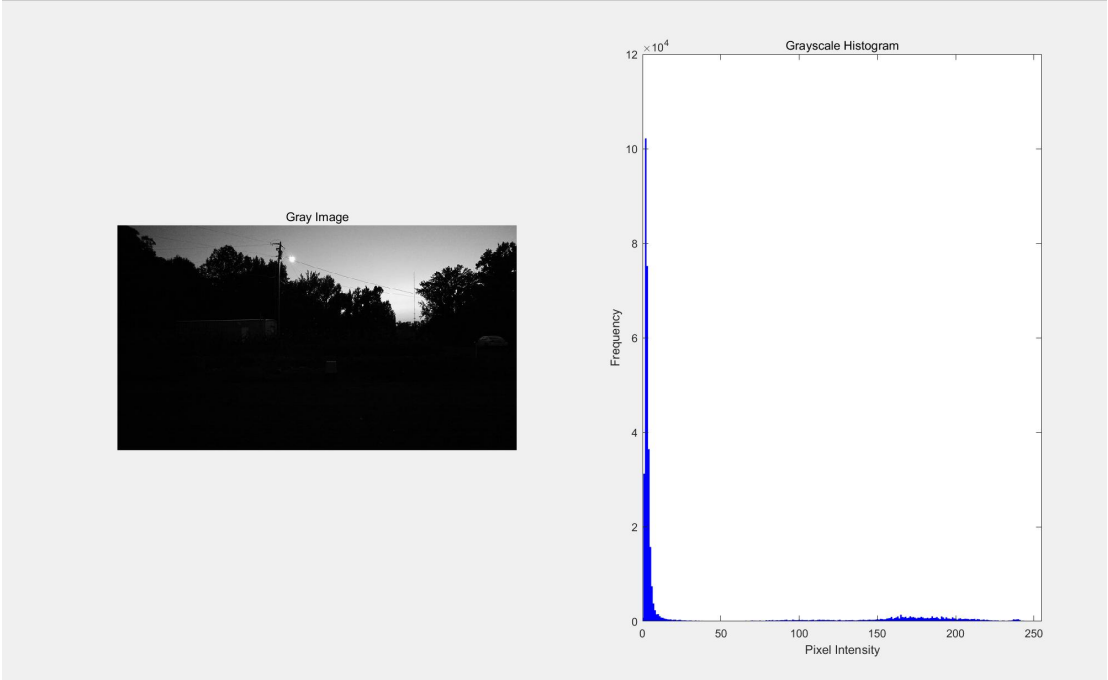
```
%计算灰度直方图
[counts, binLocations] = custom_imhist(grayImg); %  自定义直方图函数

%显示灰度图像及其直方图
figure;
subplot(1, 2, 1);
imshow(grayImg);
title('Gray Image');

subplot(1, 2, 2);
bar(binLocations, counts, 'BarWidth', 1, 'FaceColor', 'b');
xlim([0 255]);
title('Grayscale Histogram');
xlabel('Pixel Intensity');
ylabel('Frequency');

%计算并显示离散傅里叶变换频谱幅度图
magnitudeSpectrum = optimized_fft2(grayImg); %  优化傅里叶变换函数

%显示幅度谱图
figure;
imshow(magnitudeSpectrum, []);
title('Magnitude Spectrum');

%进行直方图均衡化
equalizedImg = optimized_histeq(grayImg); %  优化直方图均衡化函数

%显示直方图均衡化结果
figure;
subplot(1, 2, 1);
imshow(equalizedImg);
title('Histogram Equalized Image');

%进行同态滤波
filteredImg = optimized_homomorphic_filter(grayImg); %  优化同态滤波函数

%显示同态滤波结果
subplot(1, 2, 2);
imshow(filteredImg);
title('Homomorphic Filtered Image');
```

```
%对比直方图均衡化和同态滤波
figure;
subplot(1, 2, 1);
imshow(equalizedImg);
title('Histogram Equalized Image');

subplot(1, 2, 2);
imshow(filteredImg);
title('Homomorphic Filtered Image');

function grayImg = custom_rgb2gray(img)
    %RGB 转灰度函数
    grayImg = uint8(0.2989 * img(:,:,1) + 0.5870 * img(:,:,2) + 0.1140 * img(:,:,3));
end

function [counts, binLocations] = custom_imhist(grayImg)
    %直方图函数
    counts = histcounts(grayImg, 0:256);
    binLocations = 0:255;
end

function magnitudeSpectrum = optimized_fft2(img)
    % 二维傅里叶变换函数
    img_double = double(img); % 转换为 double 以进行傅里叶运算
    fftImg = fftshift(fft2(img_double)); % 使用二维傅里叶变换
    magnitudeSpectrum = log(1 + abs(fftImg)); % 计算幅度谱并对数变换
end

function equalizedImg = optimized_histeq(grayImg)
    % 直方图均衡化函数
    % 利用归一化 CDF 实现直方图均衡化
    counts = histcounts(grayImg, 0:256);
    cdf = cumsum(counts) / numel(grayImg); % 计算归一化的累计分布函数
    equalizedImg = uint8(255 * cdf(double(grayImg) + 1)); % 根据 CDF 映射像素值
end

function filteredImg = optimized_homomorphic_filter(grayImg)
    % 同态滤波函数
    grayImg_double = double(grayImg) + 1; % 计算对数前，避免 0 值
    logImg = log(grayImg_double);
```

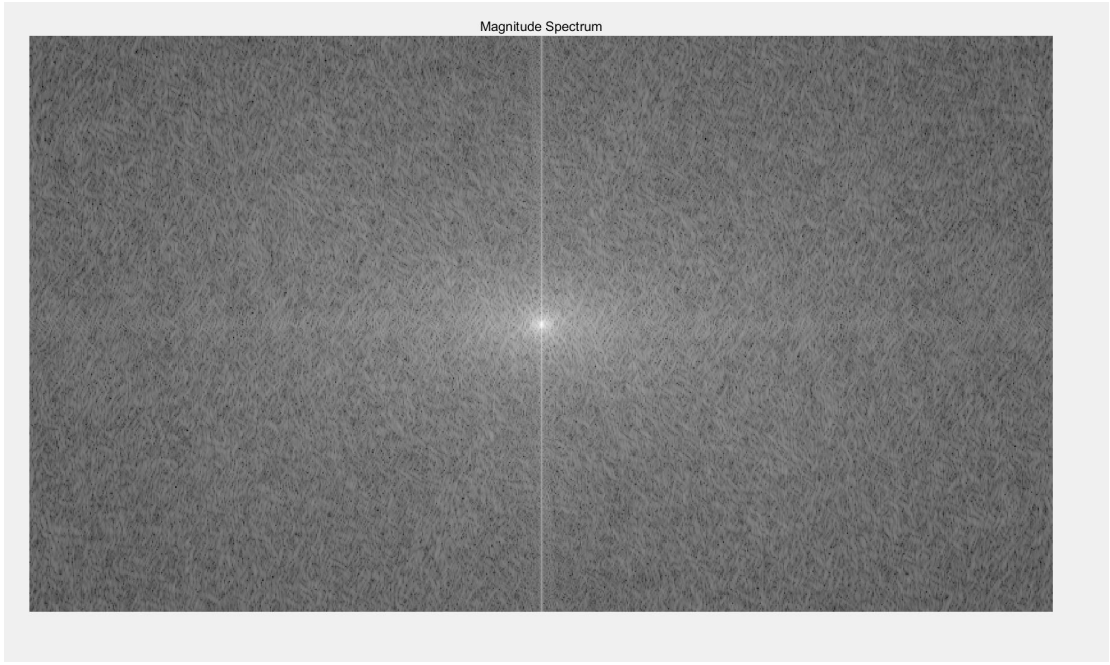
```
% 进行快速傅里叶变换
fftImg = fft2(logImg);

% 创建高斯高通滤波器
[rows, cols] = size(grayImg);
[X, Y] = meshgrid(1:cols, 1:rows);
centerX = ceil(cols / 2);
centerY = ceil(rows / 2);
D0 = 30; % 截止频率
highPassFilter = 1 - exp(-((X - centerX).^2 + (Y - centerY).^2) / (2 * D0^2));

% 应用滤波器并进行反傅里叶变换
filtered_fft = fftImg .* highPassFilter;
filteredImg = real(ifft2(filtered_fft));

% 取反对数并返回结果
filteredImg = exp(filteredImg) - 1;
filteredImg = uint8(filteredImg);
End
```

2. 运行结果图

（1）灰度图和灰度直方图



（2）傅里叶变换频谱幅度图



（3）直方图均衡化（左），同态滤波（右）

3. 效果对比

针对低照度图像，同态滤波可以更好的保留细节和避免过度亮度增强，直方图均衡化更适合整体亮度较低、对比度不够的图像，快速提升对比度，但容易丢失局部细节。。