滤波：

先创建初始矩阵和输出矩阵：

```
A = [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];
[m,n]=size(A);
% 初始化输出矩阵
mean_filtered = zeros(m, n);
median_filtered = zeros(m, n);
```

在原图分别进行滤波，保留边缘：

```
% 手动实现均值滤波
for i = 1:m
    for j = 1:n
        if i < 2 || i > m-1 || j < 2 || j > n-1
            % 边缘像素直接复制
            mean_filtered(i, j) = A(i, j);
        else
            % 提取3x3的邻域
            neighborhood = A(i-1:i+1, j-1:j+1);
            % 计算平均值并赋值给中心像素
            mean_filtered(i, j) = round(mean(neighborhood(:)));
        end
    end
end

% 手动实现中值滤波
for i = 1:m
    for j = 1:n
        if i < 2 || i > m-1 || j < 2 || j > n-1
            % 边缘像素直接复制
            median_filtered(i, j) = A(i, j);
        else
            % 提取3x3的邻域
            neighborhood = A(i-1:i+1, j-1:j+1);
            % 计算中位数并赋值给中心像素
            median_filtered(i, j) = median(neighborhood(:));
        end
    end
end
```

分别以矩阵形式输出两幅图像：

```
% 显示原始图像
A

% 显示均值滤波后的图像
mean_filtered

% 显示中值滤波后的图像
median_filtered
```

```
>> filter
警告: 函数 filter 与某个 MATLAB 内置函数同名。建议您重命名该函数以避免潜在的名称冲突。

A =

     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


mean_filtered =

     1     2     1     4     3
     1     3     4     4     4
     5     5     5     5     8
     5     5     5     7     8
     5     6     7     8     9


median_filtered =

     1     2     1     4     3
     1     2     3     4     4
     5     5     5     6     8
     5     5     6     8     8
     5     6     7     8     9
```

图像处理

先下载图像，读入并灰度化：

```
I = imread('low_light_image.jpg');
if size(I, 3) == 3
    I_gray = rgb2gray(I);
else
    I_gray = I;
end
```

计算灰度直方图

```
figure;
subplot(1,3,1);
imshow(I_gray);
title('Original Gray Image');
subplot(1,3,2);
imhist(I_gray);
title('Gray Histogram');
```

计算离散傅里叶变换：

```
% 将灰度图像转换为双精度浮点数，以进行傅里叶变换
I_double = double(I_gray);
% 计算图像的二维快速傅里叶变换（FFT）
I_fft = fft2(I_double);
% 将FFT结果的直流分量移到频谱的中心
I_fft_shift = fftshift(I_fft);
% 计算频谱的幅度
I_fft_magnitude = abs(I_fft_shift);
% 显示频谱的对数幅度图，以便更好地观察
figure;
imshow(log(1 + I_fft_magnitude), []);
title('FFT Magnitude Spectrum');
```

手动实现直方图均衡化：

```
% 计算累积分布函数（CDF）
cdf = cumsum(counts) / numel(I_gray);
% 将 CDF 映射到新的灰度级
I_histeq = uint8(gray_levels(1 + (cdf * (numel(gray_levels) - 1))));
```

% 显示直方图均衡化后的图像

figure;

imshow(I_histeq);

title('Manual Histogram Equalization Result');

同态滤波：

```
% 设置同态滤波参数
rH = 2; % 高频增益
rL = 0.2; % 低频增益
c = 1.5; % 锐化系数
D0 = 80; % 截止频率

% 对图像进行对数变换，以将乘性噪声转换为加性噪声
I_log = log(double(I_gray) + 1);
% 对图像进行零填充，以避免 wrap-around effects
[M, N] = size(I_log);
P = 2^nextpow2(M);
Q = 2^nextpow2(N);
I_padded = padarray(I_log, [P-M, Q-N], 'post');

% 计算频域滤波器
[X, Y] = meshgrid(0:Q-1, 0:P-1);
D = sqrt((X - round(P/2)).^2 + (Y - round(Q/2)).^2);
H = (rH - rL) .* (1 - exp(-c * (D.^2) / (D0^2))) + rL;
H = ifftshift(H); % 反中心化滤波器


% 对图像进行傅里叶变换
I_fft_padded = fft2(I_padded);
% 应用同态滤波器
I_filtered = I_fft_padded .* H;
% 进行逆傅里叶变换
I_filtered = ifft2(I_filtered);
% 取实部，并进行指数变换以恢复图像
I_homo = exp(real(I_filtered(1:M, 1:N))) - 1;
% 显示同态滤波后的图像
figure;
imshow(I_homo);
title('Manual Homomorphic Filtering Result');
```

结果:

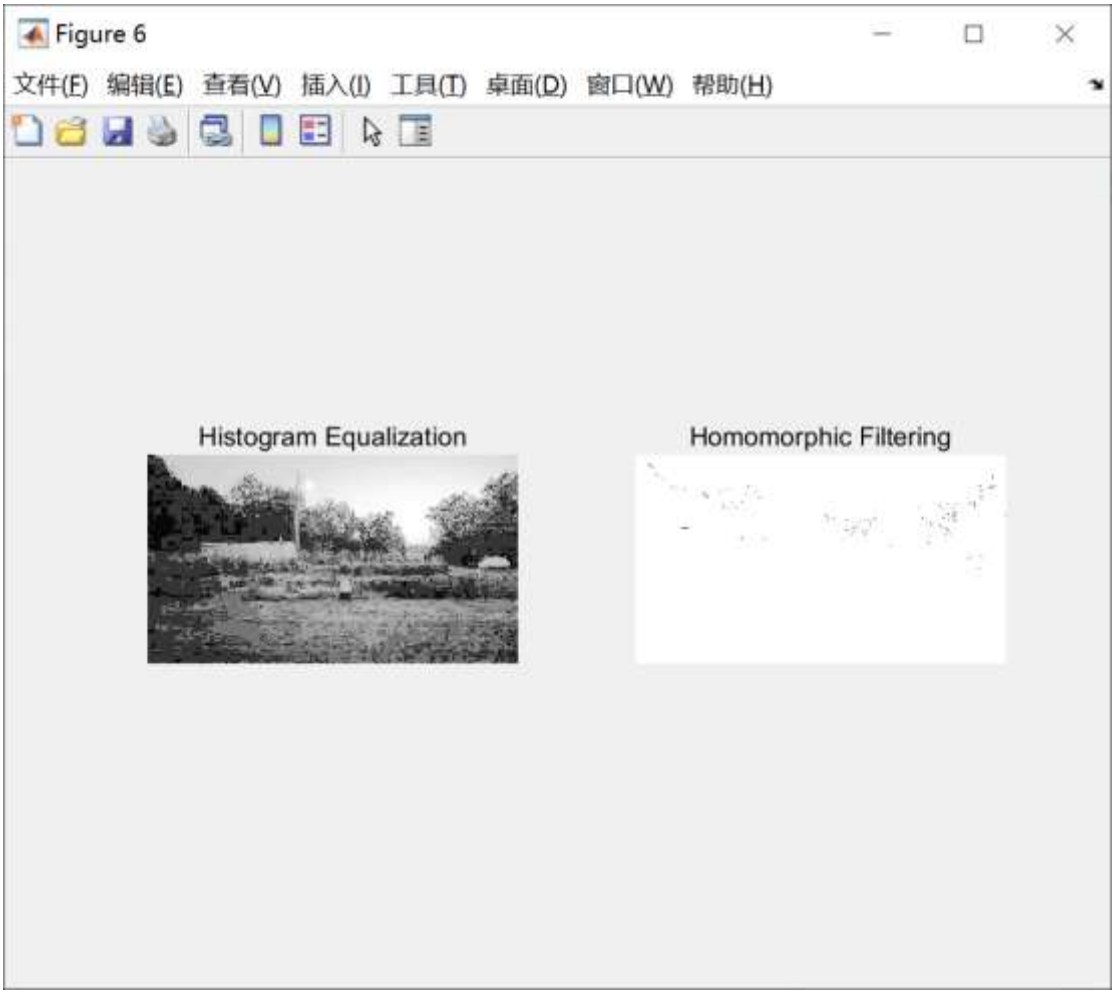
Figure 6
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Histogram Equalization
Homomorphic Filtering

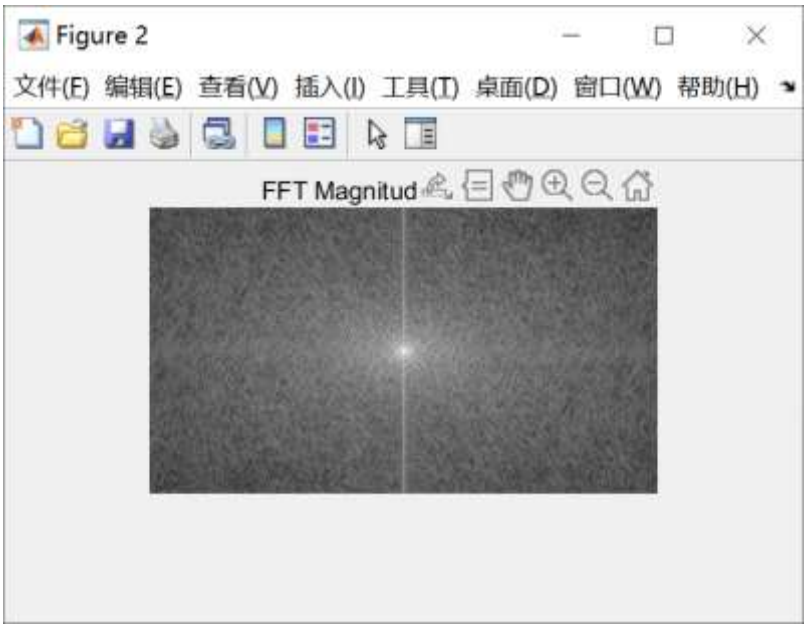
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
FFT Magnitud

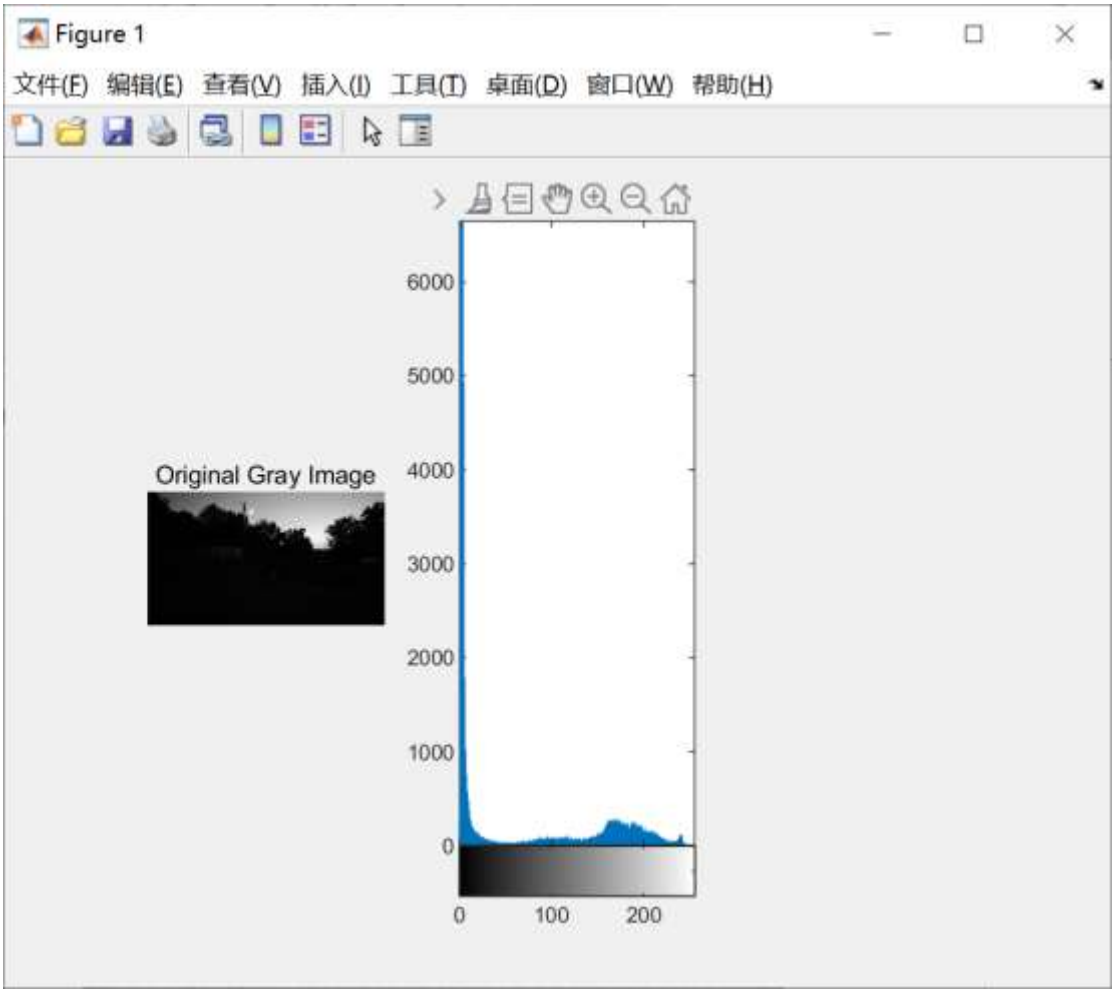
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Original Gray Image
6000
5000
4000
3000
2000
1000
0
0
100
200