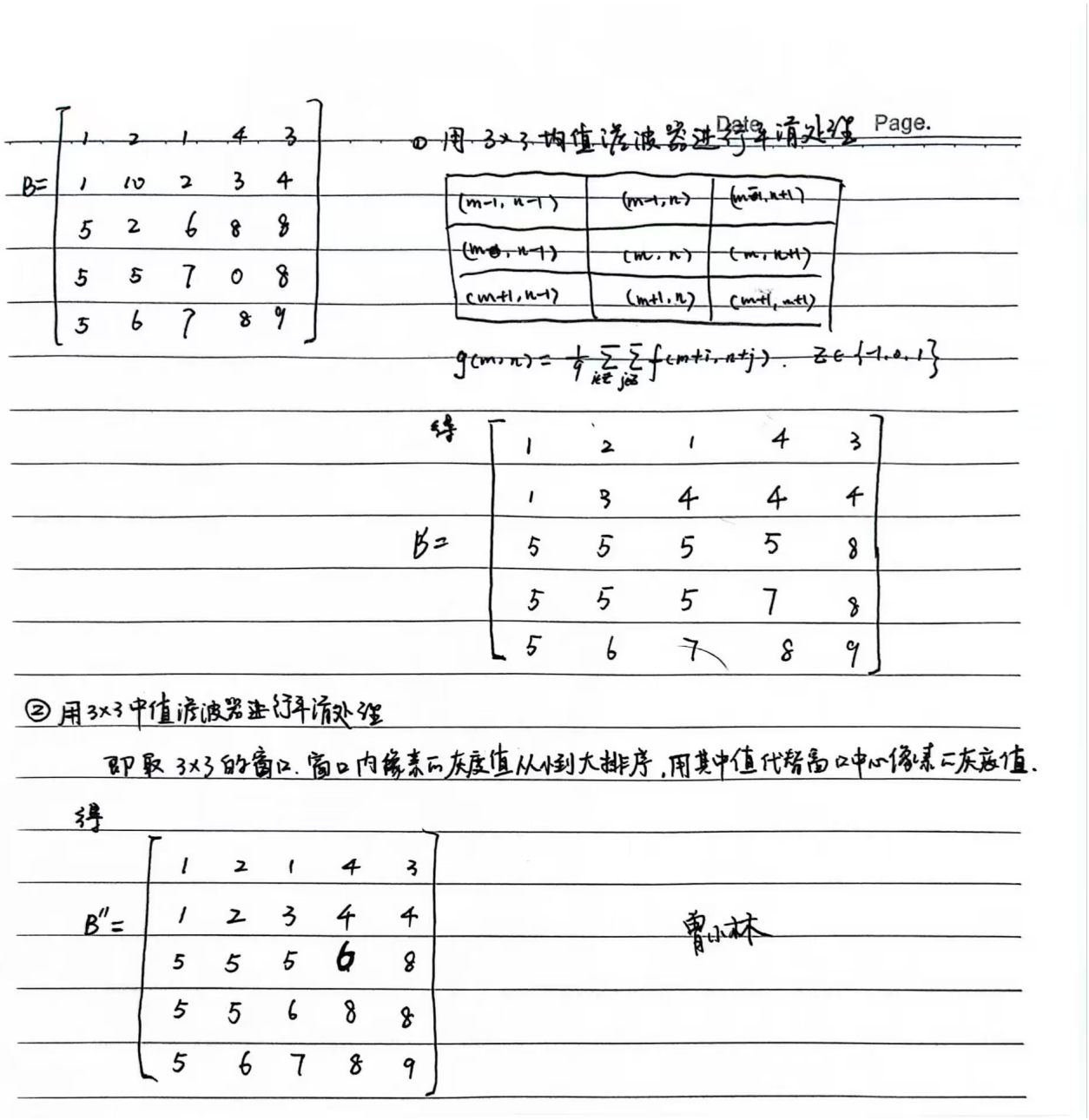
一、

第一题解答如下：



二、

以下是实现所有问题的主程序代码：

```
% 读取图像并转换为灰度图像
```

```
img = imread('低照度图像.jpg');  % 读取图像
gray_img = rgb2gray(img);  % 转换为灰度图像

% 创建result文件夹，如果不存在的话
result_folder = 'result';
if ~exist(result_folder, 'dir')
    mkdir(result_folder);
end

% 显示原始图像和灰度图像
figure, imshow(img), title('原始图像');
% 保存原始图像
imwrite(img, fullfile(result_folder, '原始图像.jpg'));

figure, imshow(gray_img), title('灰度图像');
% 保存灰度图像
imwrite(uint8(gray_img), fullfile(result_folder, '灰度图像.jpg'));

% 计算灰度直方图
hist = my_histogram(gray_img);
figure, bar(hist), title('灰度直方图');
% 保存直方图图像（需要转换为可保存的格式）
hist_img = getframe(gcf);
imwrite(hist_img.cdata, fullfile(result_folder, '灰度直方图.jpg'));

% 计算并显示傅里叶变换频谱幅度图
F = fft2(double(gray_img));  % 计算2D FFT
F_shifted = fftshift(F);  % 将低频移动到中心
F_magnitude = 20 * log(1 + abs(F_shifted));  % 计算频谱幅度
figure, imshow(F_magnitude, []), title('傅里叶频谱幅度图');
% 保存傅里叶频谱幅度图
imwrite(uint8(F_magnitude), fullfile(result_folder, '傅里叶频谱幅度
图.jpg'));

% 直方图均衡化后的图像
equalized_img = my_hist_equalization(gray_img);
%figure, imshow(equalized_img), title('直方图均衡化后的图像');
% 保存直方图均衡化后的图像
imwrite(uint8(equalized_img), fullfile(result_folder, '直方图均衡化后
的图像.jpg'));

% 设置同态滤波参数
low_freq_gain = 0.5;  % 低频增益，通常设置为小于 1 的值以压缩光照影响
```

```
high_freq_gain = 0.8;  % 高频增益，用于增强细节
cutoff = 10;  % 滤波器的截止频率，控制高频和低频的分割点

% 调用同态滤波函数
filtered_img = my_homomorphic_filter(gray_img, low_freq_gain,
high_freq_gain, cutoff);
%figure, imshow(filtered_img), title('同态滤波后的图像');
% 保存同态滤波后的图像
imwrite(uint8(filtered_img), fullfile(result_folder, '同态滤波后的图
像.jpg'));
```

接下来是对主程序中自己实现的函数的定义：

- 计算灰度直方图的函数：

```
function hist = my_histogram (gray_img)
%   计算图像灰度直方图
    hist = zeros(1,256);
    [rows,cols] = size(gray_img);
    for i = 1:rows
        for j = 1:cols
            pixel_value = gray_img(i,j);
            hist(pixel_value + 1) = hist(pixel_value + 1) + 1;
        end
    end
end
```

- 直方图均衡化的函数：

```
function hist = my_histogram (gray_img)
%   计算图像灰度直方图
    hist = zeros(1,256);
    [rows,cols] = size(gray_img);
    for i = 1:rows
        for j = 1:cols
            pixel_value = gray_img(i,j);
            hist(pixel_value + 1) = hist(pixel_value + 1) + 1;
        end
    end
end
```

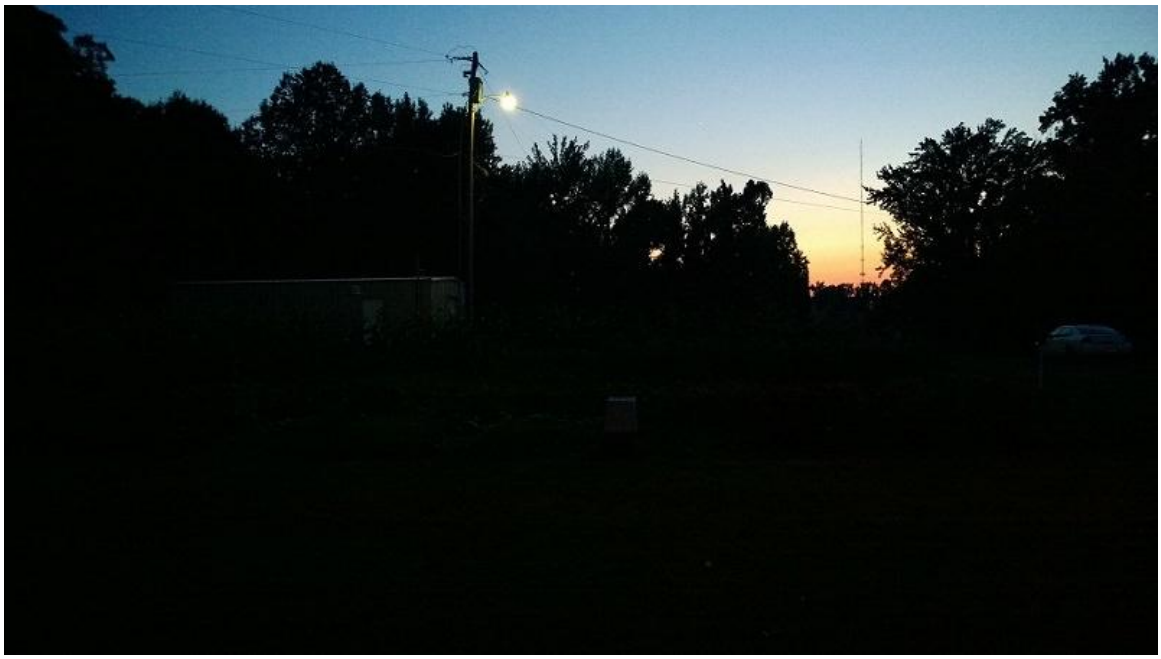
- 同态滤波的函数：

```
[function output_img = my_homomorphic_filter(gray_img,
low_freq_gain, high_freq_gain, cutoff)
    % 输入参数：
    % gray_img：输入的灰度图像
    % low_freq_gain：低频增益（用于控制光照分量的压缩）
    % high_freq_gain：高频增益（用于控制细节的增强）
    % cutoff：滤波器的截止频率]()

    % 1. 将图像转换到对数域
    log_img = log(double(gray_img) + 1);  % 为了避免 log(0)，加上一个常
数1

    % 2. 进行傅里叶变换
    F = fft2(log_img);
    F_shifted = fftshift(F);  % 将低频移动到频谱中心

    % 3. 设计高通滤波器
    [rows, cols] = size(gray_img);
    [X, Y] = meshgrid(1:cols, 1:rows);
    centerX = ceil(cols / 2);
    centerY = ceil(rows / 2);
    D = sqrt((X - centerX).^2 + (Y - centerY).^2);  % 距离中心的距离
    H = (high_freq_gain - low_freq_gain) * (1 - exp(-(D.^2) / (2 *
cutoff^2))) + low_freq_gain;

    % 4. 对频域图像进行滤波
    F_filtered = F_shifted .* H;

    % 5. 进行傅里叶反变换
    F_inv_shifted = ifftshift(F_filtered);  % 将低频移动回原位
    img_filtered_log = real(ifft2(F_inv_shifted));  % 取实部

    % 6. 将对数域图像转换回普通图像
    output_img = exp(img_filtered_log) - 1;  % 恢复图像并减去之前加的1

    % 7. 归一化结果以便显示
    output_img = uint8(255 * mat2gray(output_img));  % 将图像归一化到
[0, 255]

    figure, imshow(output_img), title('同态滤波后的图像');
```

```
end
```

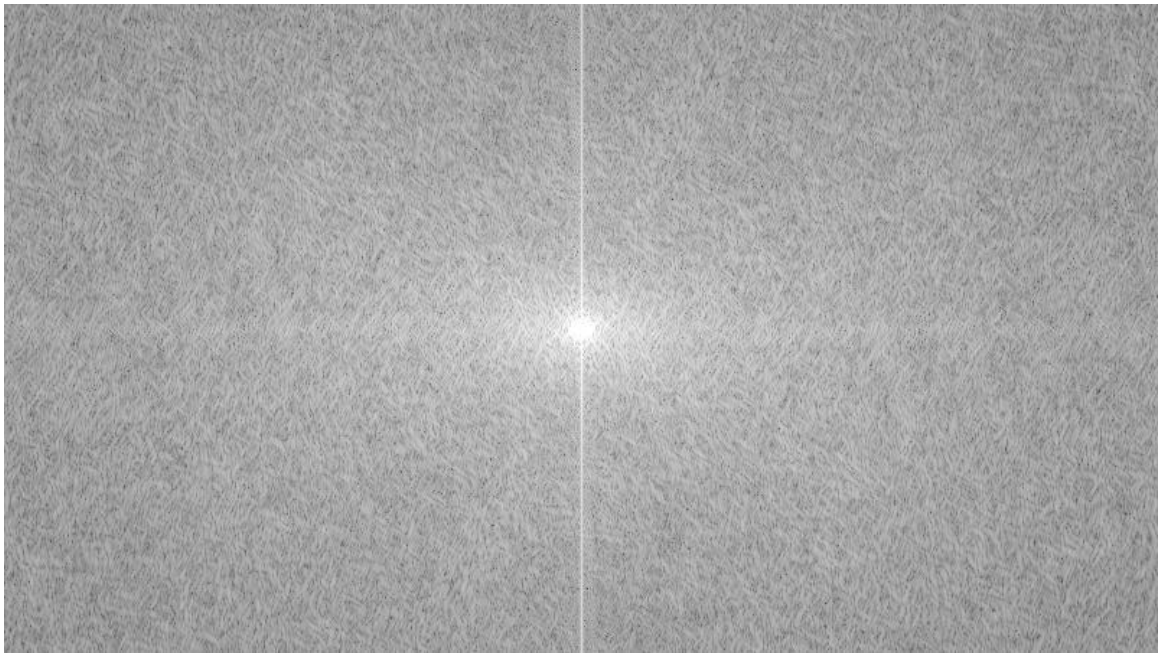
以下试运行的结果：

1. 原始图像：

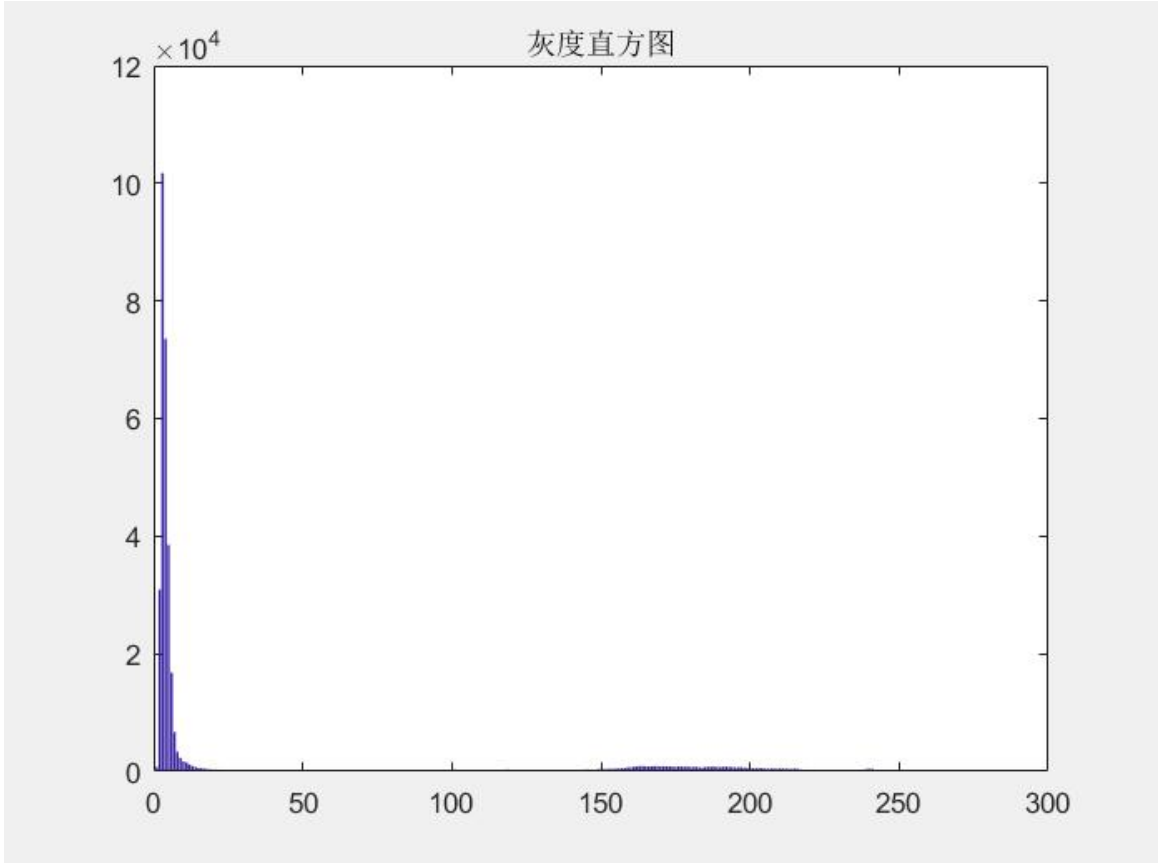


2. 灰度图像：

3. 傅里叶频谱图：



4. 灰度直方图：



5. 直方图均衡化后的图像：

6. 同态滤波后的图像：

对直方图均衡化和同态滤波最终效果对比分析：

直方图均衡化：

- 对比度增强：从处理结果来看，直方图均衡化的主要效果是提升了图像的整体对比度。它重新分布了灰度值，使得暗部区域变亮，细节更加明显。这种处理方式尤其适用于图像中明暗对比不明显的区域。
- 细节增强：通过拉伸灰度值，更多的图像细节在明亮和暗部区域都得到了增强，使得场景更加清晰。不过，由于直方图均衡化对整个图像应用相同的变换，容易导致某些区域过亮或过暗。

可以注意到，在原始灰度图中就是阴暗对比不是很明显，所以在这里对比度增强和细节增强的效果还是比较明显的。

同态滤波：

- 照明均匀化：经过同态滤波处理后，暗部区域进一步压缩，而高频部分（如边缘和细节）得到了加强。同态滤波通常用于抑制光照变化的影响，特别是对于不均匀光照下的图像可以有效减少过亮或过暗区域的影响。
- 噪声引入：同态滤波可能会引入一些高频噪声，尤其是在细节区域的增强上。图像中可能出现颗粒感或伪影，尤其在处理非常暗的区域时会更为明显。

结论：

1. 直方图均衡化适合在场景对比度不足的情况下使用，能够有效提升图像的对比度和细节，适用于图像整体增强。但如果图像本身照明不均匀，可能会出现局部过度增强的问题。
2. 同态滤波则更加适合处理光照不均匀的场景，它能够在压缩不均匀照明的同时增强细节，但可能会引入噪声，因此适用于需要同时处理照明和细节增强的情况。
3. 从效果上看，如果我们想要整体提升对比度并增加细节，直方图均衡化会有更直接的效果。而如果我们更关注光照均匀性和局部细节增强，同态滤波是更优的选择。