華中科技大學

数字图像处理课后作业

第二次作业

学　　号: U202215138

姓　　名: 康王梓宝

院系: 人工智能与自动化学院

班　　级: 人工智能 2204 班

报告日期: 2024.10.19

目录

一、 实验目的 1
二、实验环境 1
三、实验内容 1
1. 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 1
2. 对低照度图像进行灰度化，计算并显示图像的灰度直方图和离散傅里叶变换频谱幅度图； 2
3. 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 4
四、源代码 5

一、实验目的

对低照度图像进行多种图像处理操作，验证均值滤波器、中值滤波器、直方图均衡化和同态滤波的效果，分析不同滤波方法对图像的平滑和对比度增强效果

二、实验环境

Matlab 仿真

三、实验内容

1. 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

均值滤波后的矩阵：

1.0000	2.0000	1.0000	4.0000	3.0000
1.0000	3.3333	4.2222	4.3333	4.0000
5.0000	4.7778	4.7778	5.1111	8.0000
5.0000	5.3333	5.4444	6.7778	8.0000
5.0000	6.0000	7.0000	8.0000	9.0000

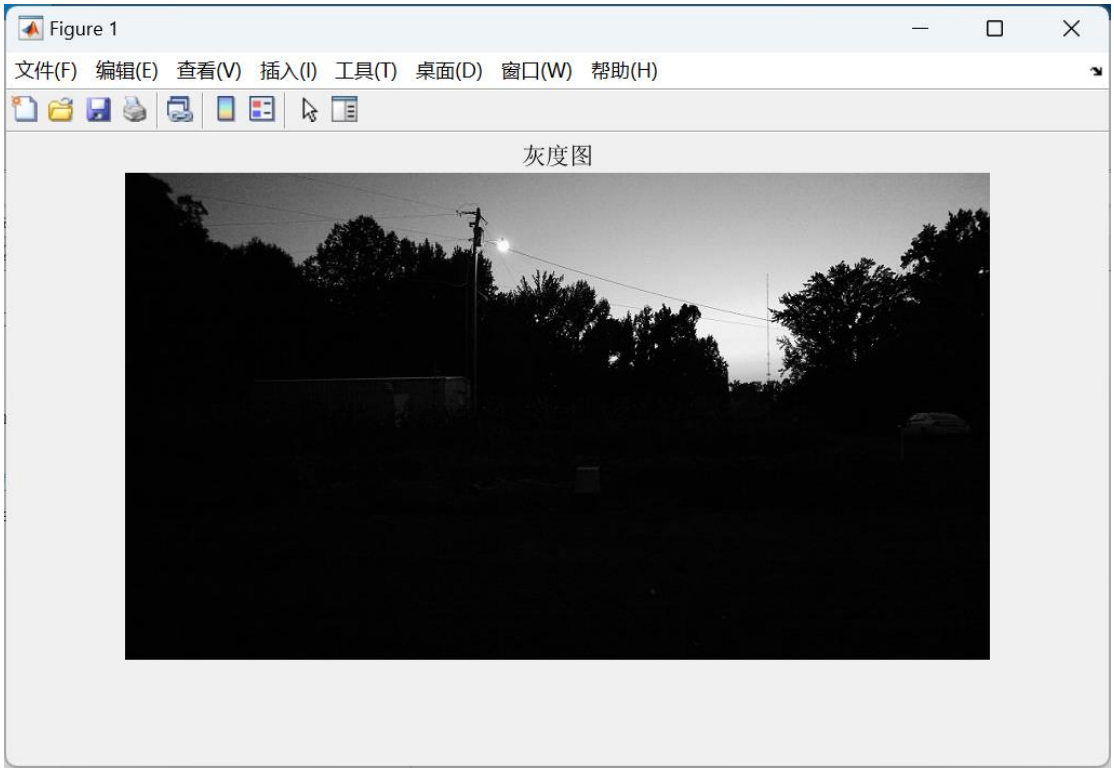
中值滤波后的矩阵：

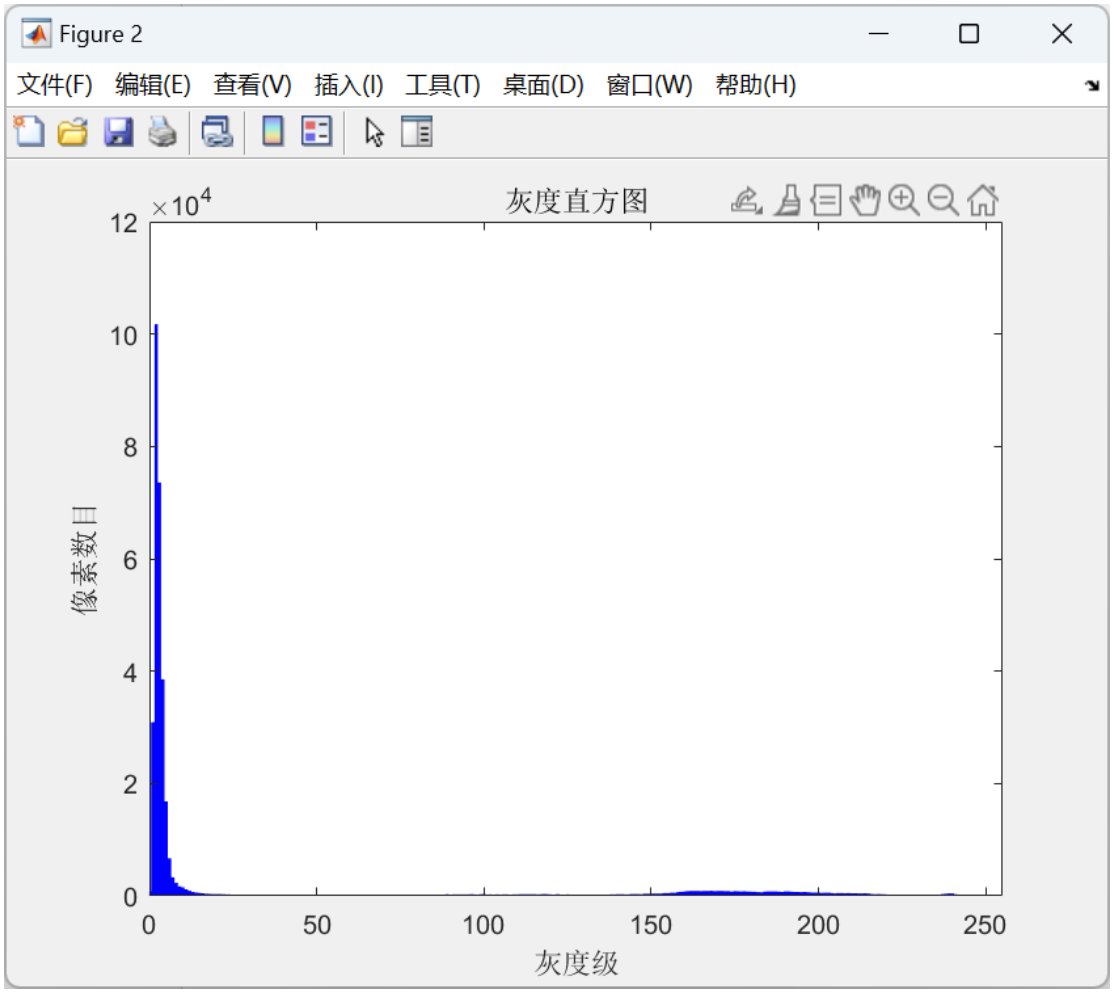
1	2	1	4	3
1	2	3	4	4
5	5	5	6	8
5	5	6	8	8
5	6	7	8	9

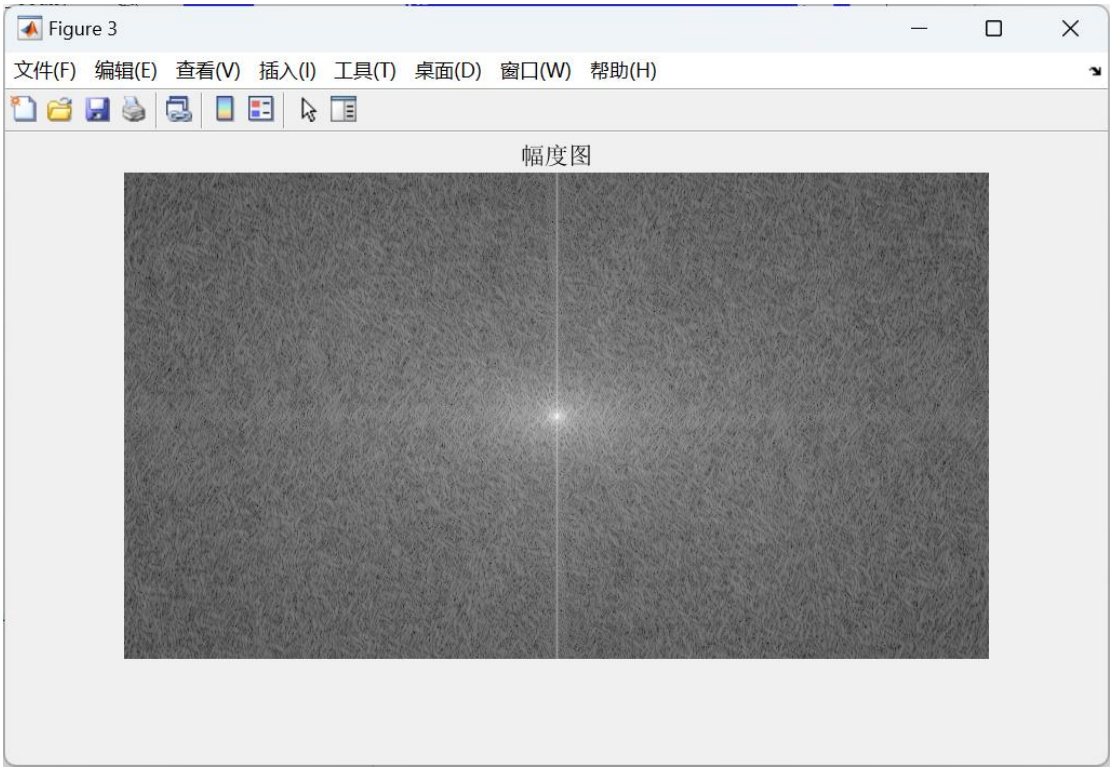
滤波器处理步骤：

1. 对每个像素点，获取其 3×3 邻域内的所有像素值。
2. 计算这些像素值的均值，并将均值赋给中心像素点。

2. 对低照度图像进行灰度化，计算并显示图像的灰度直方图和离散傅里叶变换频谱幅度图；



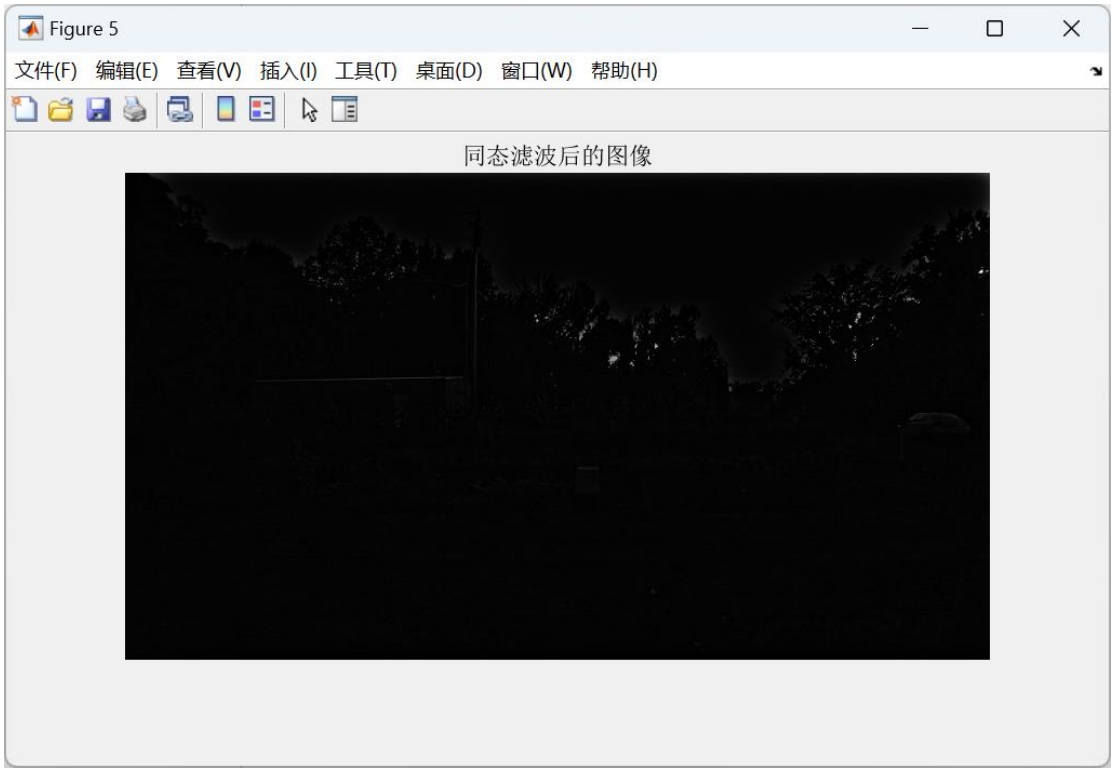


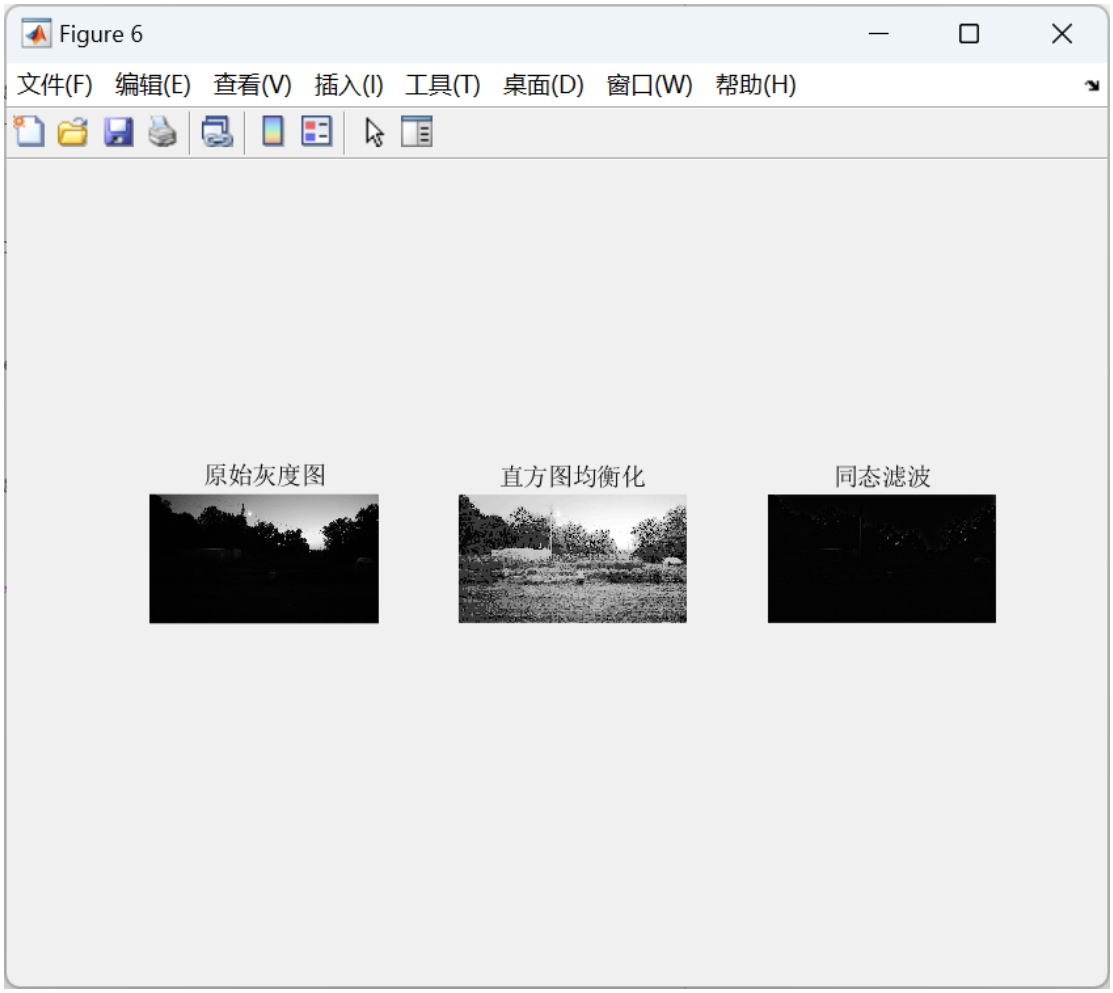


实验步骤：

1. 对每个像素点，获取其 3×3 邻域内的所有像素值。计算这些像素值的均值，并将均值赋给中心像素点。
2. 计算灰度图像中每个灰度级别的像素数量。遍历灰度图像中的每个像素，记录各灰度值的出现次数。将计算得到的灰度级分布绘制成直方图，
3. 对灰度图像进行对数变换，使用 FFT 函数对图像进行二维傅里叶变换通过移位操作将频谱的零频成分移动到中心，并计算幅度谱，使用对数尺度显示幅度谱。

3. 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





四、源代码

1. 均值与中值滤波器

```
function [mean_filtered, median_filtered] =
filter_2d_matrix(input_matrix)
    % 对二维矩阵进行均值滤波和中值滤波
    % 输入参数:
    % input_matrix - 输入的二维矩阵
```

```
    % 输出参数:
    % mean_filtered - 经过均值滤波后的矩阵
    % median_filtered - 经过中值滤波后的矩阵

    [rows, cols] = size(input_matrix); % 获取行数和列数

    mean_filtered = input_matrix; % 初始化
    median_filtered = input_matrix;

    % 均值滤波
    for i = 2:rows-1
        for j = 2:cols-1
            window = input_matrix(i-1:i+1, j-1:j+1); % 取3x3
窗口
            mean_filtered(i, j) = mean(window(:)); % 计算均值
        end
    end

    % 中值滤波
    for i = 2:rows-1
        for j = 2:cols-1
            window = input_matrix(i-1:i+1, j-1:j+1); % 取3x3
窗口
            median_filtered(i, j) = median(window(:)); % 计
算中值
        end
    end
end
input_matrix = [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, median_filtered] =
filter_2d_matrix(input_matrix);


disp('均值滤波后的矩阵:');
disp(mean_filtered);
disp('中值滤波后的矩阵:');
disp(median_filtered);
```

2. 灰度直方图，离散傅里叶变换频谱幅度图，直方图均衡化与同态滤波

```
% 读取图像
input_image =
imread('D:\BaiduNetdiskDownload\homework_img_2.jpg');

% 灰度化处理
[rows, cols, channels] = size(input_image);
if channels == 3
    % 使用加权平均法将RGB图像转换为灰度图像
    gray_image = uint8(0.2989 * double(input_image(:,:,1))
+ 0.5870 * double(input_image(:,:,2)) + 0.1140 *
double(input_image(:,:,3)));
else
    gray_image = input_image;
end

% 灰度图
figure;
imshow(gray_image);
title('灰度图');

% 灰度直方图计算
hist_counts = zeros(1, 256);
for i = 1:rows
    for j = 1:cols
        intensity = gray_image(i, j);
```

```
        hist_counts(intensity + 1) = hist_counts(intensity
+ 1) + 1;
    end
end

% 显示灰度直方图
figure;
bar(0:255, hist_counts, 'BarWidth', 1, 'FaceColor', 'b');
xlim([0 255]);
title('灰度直方图');

xlabel('灰度级');

ylabel('像素数目');


% 计算DFT
fft_image = fft2(double(gray_image));

% 计算幅度
magnitude_spectrum = abs(fftshift(fft_image));

% 幅度图
figure;
imshow(log(1 + magnitude_spectrum), []);
title('幅度图');


% 直方图均衡化
he_image = histeq(gray_image);
figure;
imshow(he_image);
title('直方图均衡化后的图像');


% 同态滤波

% 构建同态滤波的高斯高通滤波器
```

```
sigma = 30; % 增大sigma值使得滤波器效果更加平滑

[M, N] = size(gray_image);
[X, Y] = meshgrid(1:N, 1:M);
centerX = ceil(N / 2);
centerY = ceil(M / 2);
D = ((X - centerX).^2 + (Y - centerY).^2);
H = 1 - exp(-D / (2 * sigma^2));

% 对数变换，加上一个小常数以避免数值误差

epsilon = 1;
log_image = log(double(gray_image) + 1 + epsilon);

% 频域滤波

fft_log_image = fft2(log_image);
filtered_fft_image = fft_log_image .* fftshift(H);

% 反傅里叶变换

filtered_image = real(ifft2(filtered_fft_image));

% 指数反变换

homo_image = exp(filtered_image) - 1;

% 调整归一化过程，确保低亮度区域的细节保留

min_val = min(homo_image(:));
max_val = max(homo_image(:));
homo_image = uint8((homo_image - min_val) / (max_val -
min_val) * 255);

% 显示同态滤波后的图像

figure;
imshow(homo_image);

title('改进后的同态滤波后的图像');


% 对比效果

figure;
```

```
subplot(1, 3, 1);
imshow(gray_image);
title('原始灰度图');

subplot(1, 3, 2);
imshow(he_image);
title('直方图均衡化');

subplot(1, 3, 3);
imshow(homo_image);
title('同态滤波');
```