1. 均值滤波器处理

处理步骤：

对于矩阵中的每个非边界像素，计算其 3×3 邻域内所有像素的平均值。

将计算得到的平均值赋给中心像素。

边界像素保持不变。

得：

$$B_{\text{mean}} = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 3.33 & 3.33 & 4.33 & 4 \\ 5 & 4.78 & 6 & 7.33 & 8 \\ 5 & 5 & 7 & 0 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

2. 中值滤波器处理

处理步骤：

对于矩阵中的每个非边界像素，找出 3×3 邻域内所有像素的中值。

将这个中值赋给中心像素。

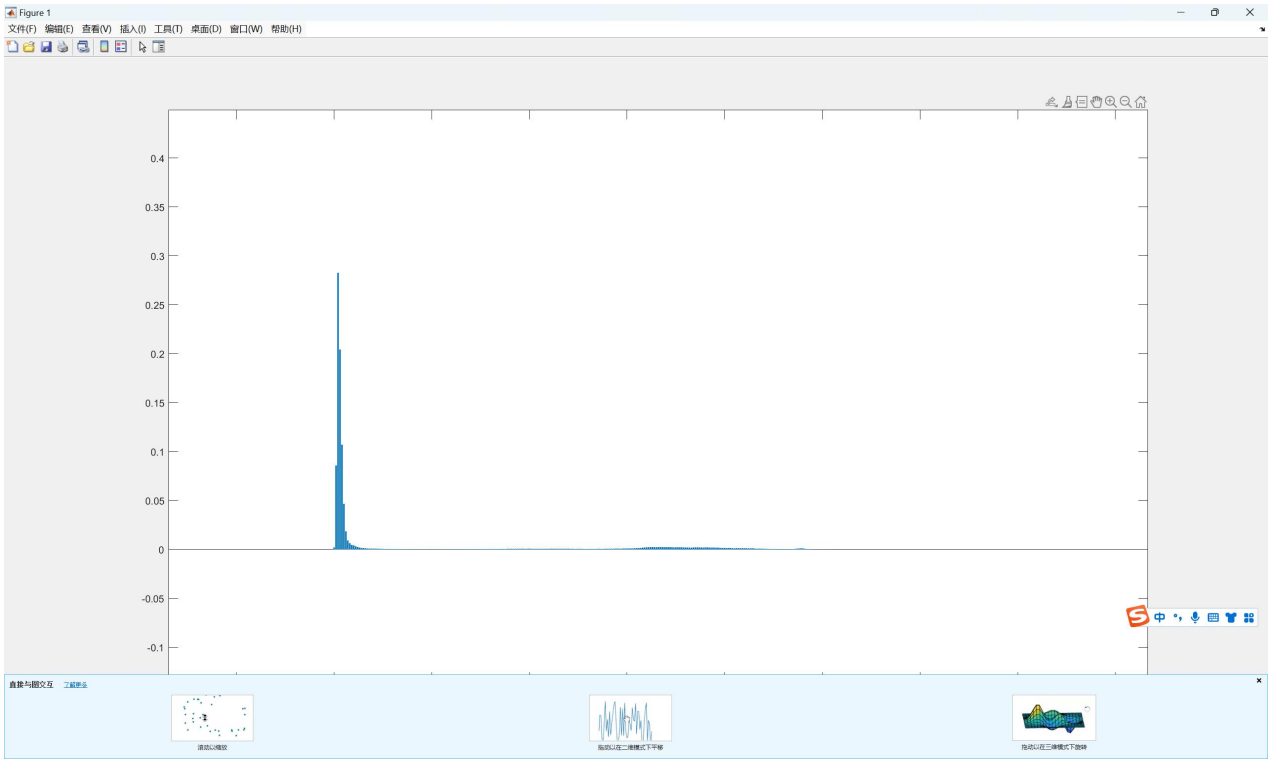
边界像素保持不变。

得：

$$B_{\text{median}} = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 6 & 6 & 4 & 4 \\ 5 & 6 & 6 & 7 & 8 \\ 5 & 5 & 7 & 0 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

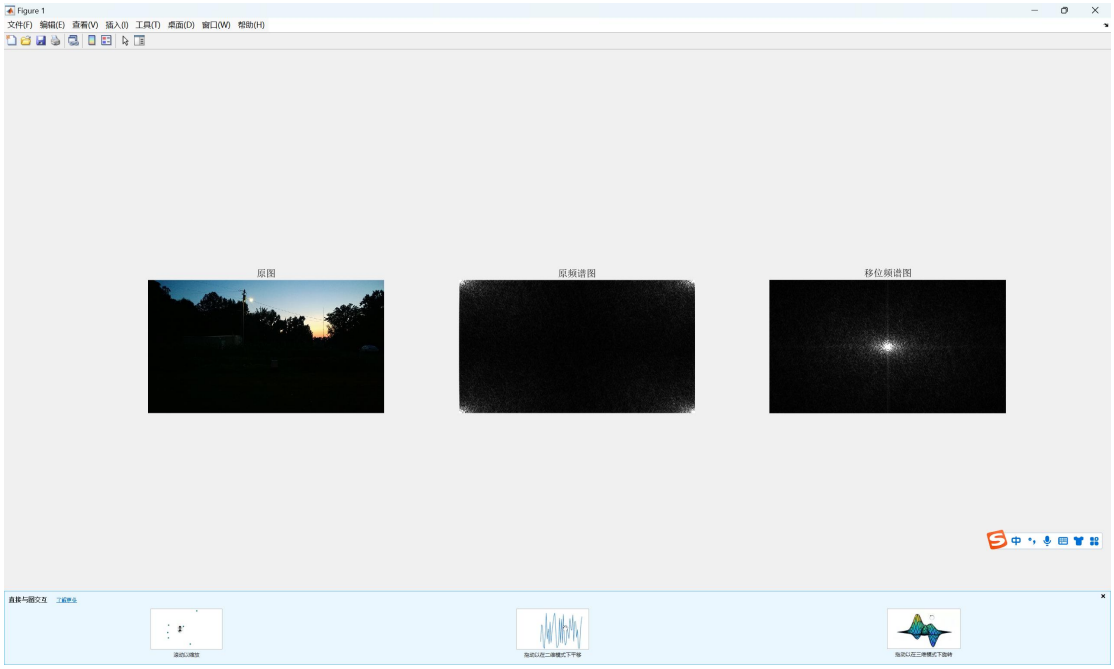
灰度：

```
function H= my_imhist(img)
% 输入:灰度图像img
% 输出:图像的直方图H
% 创建长度为256的数组hist
hist = zeros(256,1);
% 遍历整个图像矩阵，对于每个像素值，增加hist数组中相应值的计数器
[M,N]= size(img);
for i = 1:M
for j= 1:N
k= img(i,j);
hist(k+1)= hist(k+1)+ 1;
end
end
% 将hist数组中的值除以图像中像素的总数，得到每个像素值的归一化直方图
H=hist /(M*N);
```



傅里叶：

```
function H= my_imhist(img)
% 输入:灰度图像img
% 输出:图像的直方图H
% 创建长度为256的数组hist
hist = zeros(256,1);
% 遍历整个图像矩阵，对于每个像素值，增加hist数组中相应值的计数器
[M,N]= size(img);
for i = 1:M
for j= 1:N
k= img(i,j);
hist(k+1)= hist(k+1)+ 1;
end
end
% 将hist数组中的值除以图像中像素的总数，得到每个像素值的归一化直方图
H=hist /(M*N);
end
```



直方图

```
Img= imread('p.jpg');
Img=rgb2gray(Img);

%绘制原始图像的直方图
[height,width]=size(Img);
[counts1, x] = imhist(Img,256);
counts2 = counts1/height/width;
figure(1),
subplot(1,2,1),
imshow(Img);title('原始图像');
subplot(1,2,2),
stem(x, counts2); title('原始图像直方图');

%统计每个灰度的像素值累计数目
NumPixel = zeros(1,256);%统计各灰度数目，共256个灰度级
for i = 1:height
    for j = 1: width
    %对应灰度值像素点数量+1
    %NumPixel的下标是从1开始，而图像像素的取值范围是0~255，所以用NumPixel(Img(i,j) + 1)
    NumPixel(Img(i,j) + 1) = NumPixel(Img(i,j) + 1) + 1;
    end
end

%将频数值算为频率
ProbPixel = zeros(1,256);
for i = 1:256
    ProbPixel(i) = NumPixel(i) / (height * width * 1.0);
end

%函数cumsum来计算cdf，并将频率（取值范围是0.0~1.0）映射到0~255的无符号整数
```

```
%函数cumsum来计算cdf，并将频率（取值范围是0.0~1.0）映射到0~255的无符号整数
CumuPixel = cumsum(ProbPixel);
CumuPixel = uint8(255 .* CumuPixel + 0.5);

%直方图均衡。赋值语句右端，Img(i,j)被用来作为CumuPixel的索引
for i = 1:height
    for j = 1: width
        Img(i,j) = CumuPixel(Img(i,j)+1);
    end
end

%显示更新后的直方图
figure(2),
subplot(1,2,1),
imshow(Img); title('直方图均衡化图像');
[counts1, x] = imhist(Img,256);
counts2 = counts1/height/width;
subplot(1,2,2),
stem(x, counts2); title('直方图均衡化后图像的直方图');
```

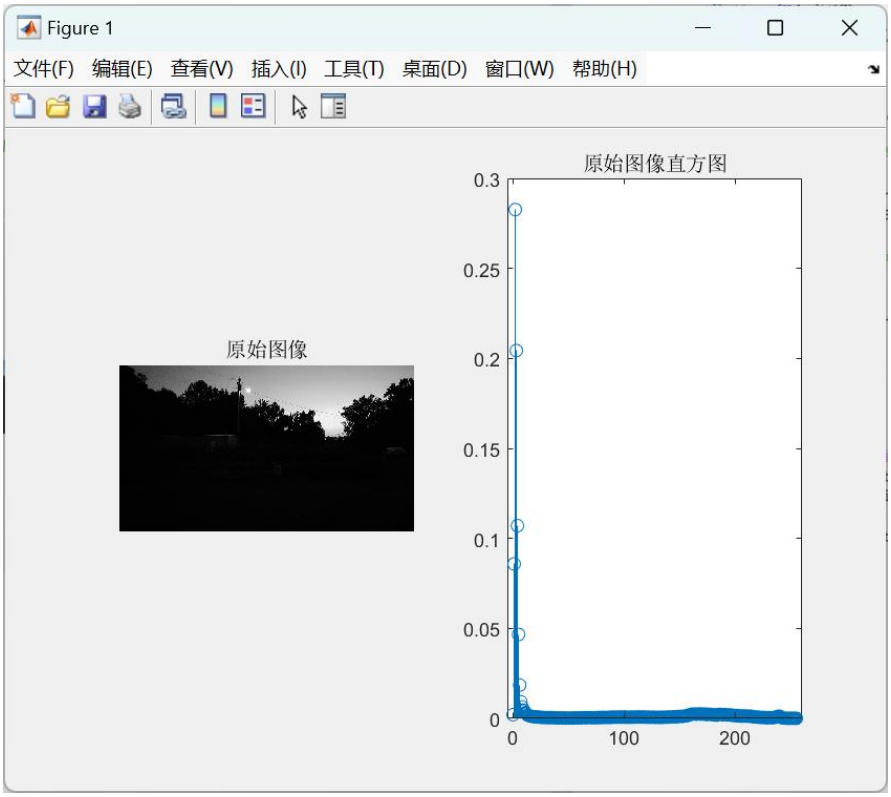


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原始图像
原始图像直方图
0.3
0.25
0.2
0.15
0.1
0.05
0
0
100
200

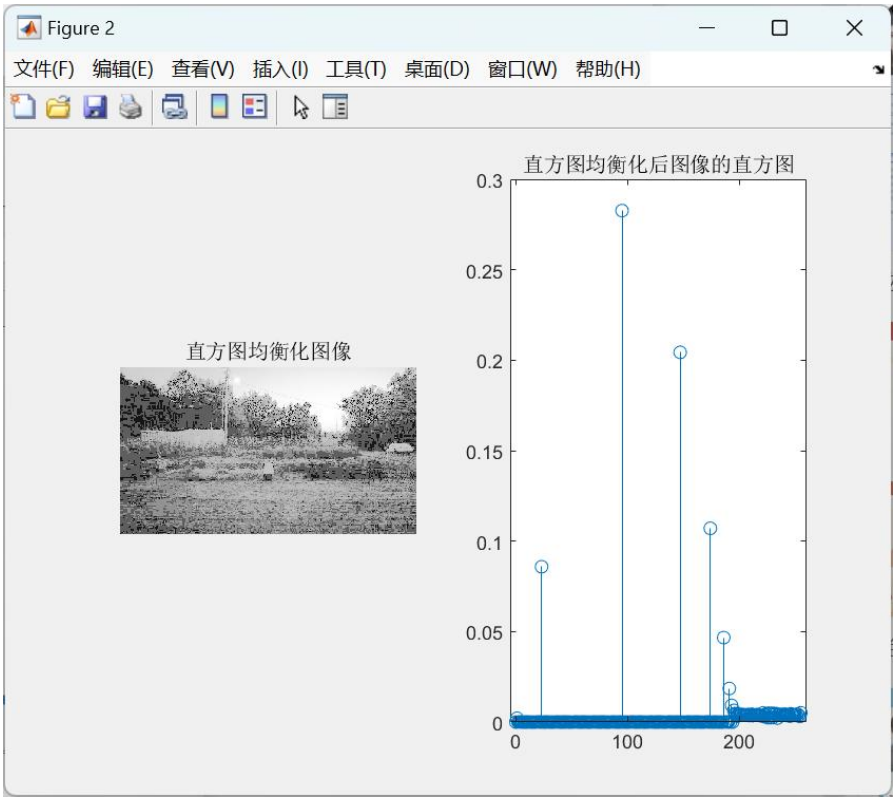
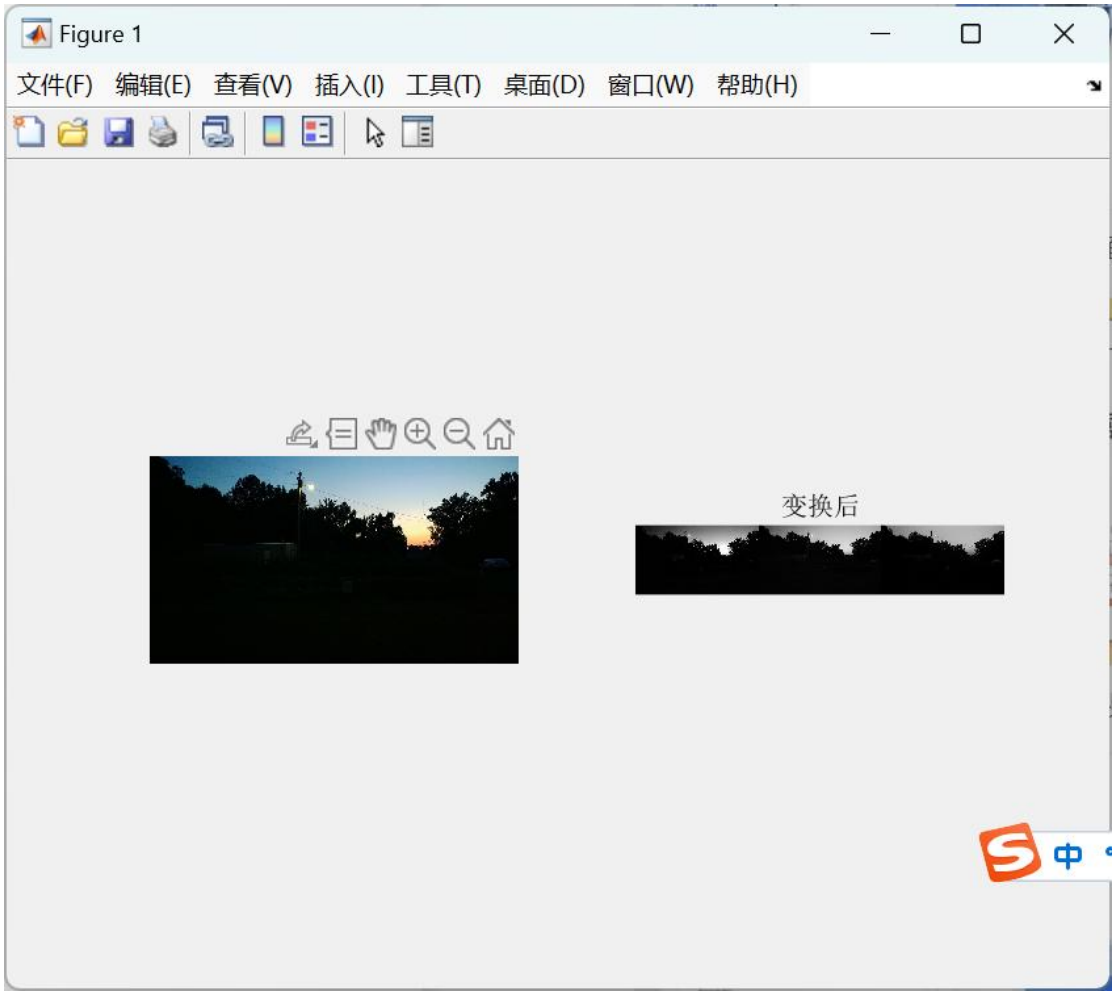


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
直方图均衡化图像
直方图均衡化后图像的直方图
0.3
0.25
0.2
0.15
0.1
0.05
0
0
100
200

同态滤波

```
%参数声明
rH = 1;
rL = 0.1;
c = 0.2;%介于rH和rL之间
D0 = 0.2;

image = imread('p.jpg');
[M, N] = size(image);
%取对数
img_log = log(double(image) + 1);

%平移到中心，判断语句代替指数计算
img_py = zeros(M, N);
for i = 1:M
   for j= 1:N
       if mod(i+j, 2) == 0
           img_py(i,j) = img_log(i, j);
       else
           img_py(i,j) = -1 * img_log(i, j);
       end
   end
end

% 对填充后的图像进行傅里叶变换
img_py_fft = fft2(img_py);

%同态滤波函数
img_tt = zeros(M, N);
deta_r = rH - rL;
D = D0^2;
m_mid=floor(M/2);%中心点坐标
n_mid=floor(N/2);

for i = 1:M
   for j =1:N
        dis = ((i-m_mid)^2+(j-n_mid)^2);
        img_tt(i, j) = deta_r * (1-exp((-c)*(dis/D))) + rL;
   end
end

%滤波
img_temp =   img_py_fft.*img_tt;

%反变换,取实部，绝对值
img_temp = abs(real(ifft2(img_temp)));

%指数化
img_temp = exp(img_temp) - 1;


%归一化处理
max_num = max(img_temp(:));
min_num = min(img_temp(:));
range = max_num - min_num;
img_after = zeros(M,N,'uint8');
for i = 1 : M
    for j = 1 : N
        img_after(i,j) = uint8(255 * (img_temp(i, j)-min_num) / range);
    end
end

subplot(1,2,1), imshow(image), title('原图像');
subplot(1,2,2), imshow(img_after), title('变换后');
```



刘恒宇 人工智能 2204 班 U202214932