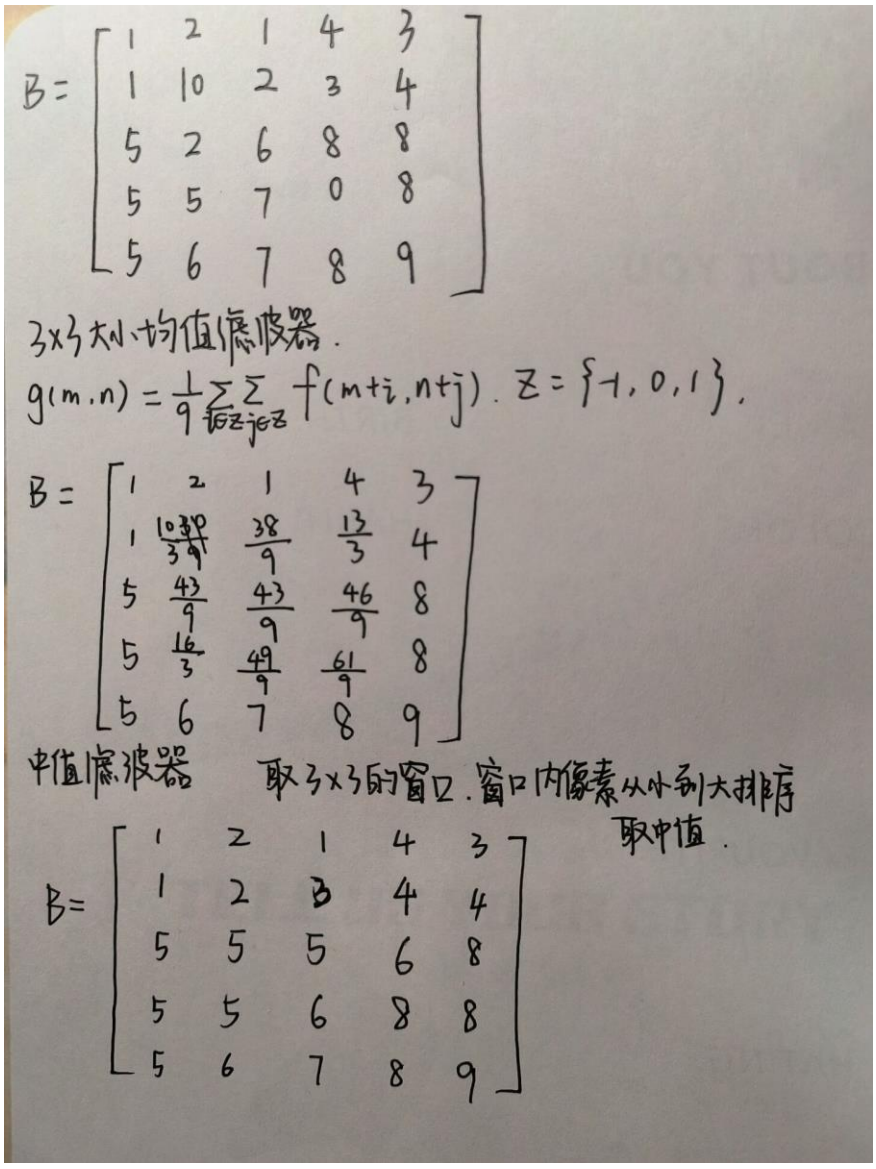
李昊泽 U202211174 人工智能 2204 班

理论作业



编程作业

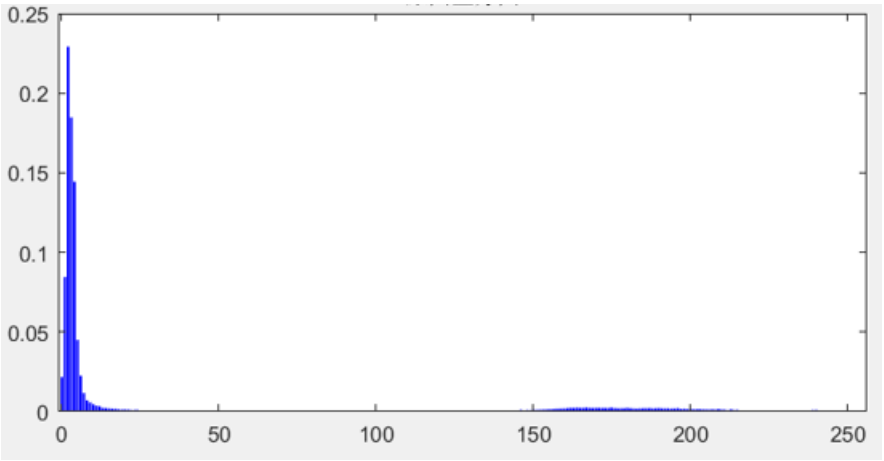
对上述低照度图像进行灰度化，计算并显示以上低照度图像的灰度直方图和离散傅里叶变换频谱幅度图

代码：

```
>> % 输入图像
A = imread('C:\Users\华为\OneDrive\图片\本机照片\作业.jpg');

%将原图转换为灰度图像
```

```
H = im2gray(A);     %从 RGB 创建灰度图
imwrite(H,'C:\Users\华为\OneDrive\图片\本机照片\作业灰度图.jpg');

[m,n]=size(H);              % 计算图像的长宽
p=zeros(1,256);         %创建数组存储像素概率
% 统计每个像素值出现的概率，  得到概率直方图
for i=0:255
    % 用 length 函数计算相同像素的个数
    p(i+1)=length(find(H==i))/(m*n);
end

%输出灰度图
subplot(2,2,1);          %用 subploy 函数将多个图像画到同一个平面中
imshow(H);                 %输出图像
title('灰度图');
%画出原图直方图
subplot(2,2,2);
bar(0:255,p,'b');
title('原图直方图');

% 求累计概率，得到累计直方图
s=zeros(1,256);
for i=1:256
        for j=1:i
            s(i)=p(j)+s(i);
        end
end
```

实验结果：

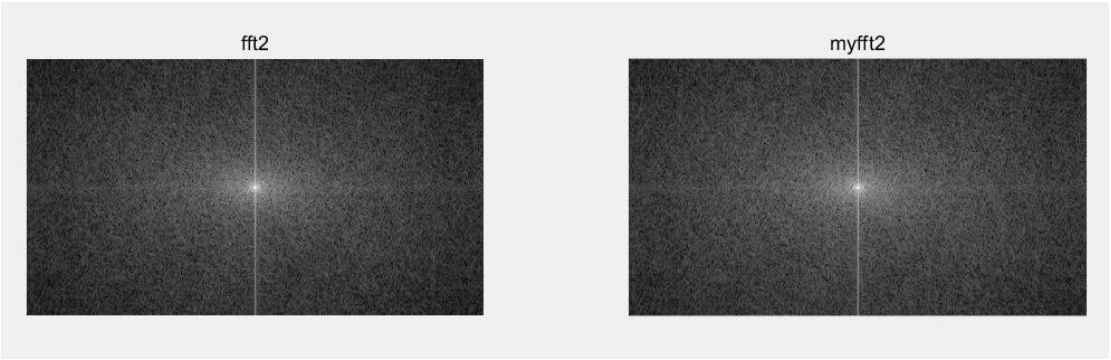


代码：

```
clc; clear;

data = imread('C:\Users\华为\OneDrive\图片\本机照片\作业.jpg');
data = im2double(data);
% data = rgb2gray(data);        % rgb 转为灰度图像
subplot(1,3,1);
imshow(data);
title('原始图像')

zidai = fft2(data);    % matlab 自带函数，来用对比
T1= abs(zidai);      %傅里叶变换的模
T1= fftshift(T1);    %对傅里叶变换后的图像进行象限转换
T1 = log(T1+1);       %数据范围压缩
subplot(1,3,2);
% imshow(real(zidai));  % 一般只要实部
imshow(T1,[]);
title('fft2');

size_data = size(data);
M = size_data(1);  % 图(原始数据矩阵)的长
N = size_data(2);  % 图(原始数据矩阵)的宽

% 下面是傅里叶正变换必备的一些矩阵:
Wm = exp(-j*2*pi/M);
Wn = exp(-j*2*pi/N); % 不同 G 中用不同的 W
Em = zeros(M);
En = zeros(N);        % E 是辅助计算矩阵
Gm = zeros(M)+Wm;
Gn = zeros(N)+Wn;   % G 是计算时要用的矩阵
F = zeros(M,N);       % F 是转换到频域的结果

% 对 Gm 的计算: 循环长度为 M
```

```
fprintf('二维离散傅里叶变换开始:\n');
for row = 0:M-1
    for col = 0:M-1
        Em(row+1,col+1) = row * col;
        Gm(row+1,col+1) = Gm(row+1,col+1)^Em(row+1,col+1);
    end
end
% 对 Gn 的计算: 循环长度为 N
for row = 0:N-1
    for col = 0:N-1
        En(row+1,col+1) = row * col;
        Gn(row+1,col+1) = Gn(row+1,col+1)^En(row+1,col+1);
    end
end

% F = real(Gm*data*Gn);  % F = Gm*f*Gn 是计算公式，一般只要实部
F = Gm*data*Gn;
subplot(1,3,3);
T=fftshift(F);    %对傅里叶变换后的图像进行象限转换
T=abs(T);            %傅里叶变换的模
T=log(T+0.7);   %数据范围压缩
% imshow(F);
imshow(T,[]);
title('myfft2');
```

实验结果：



对以上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

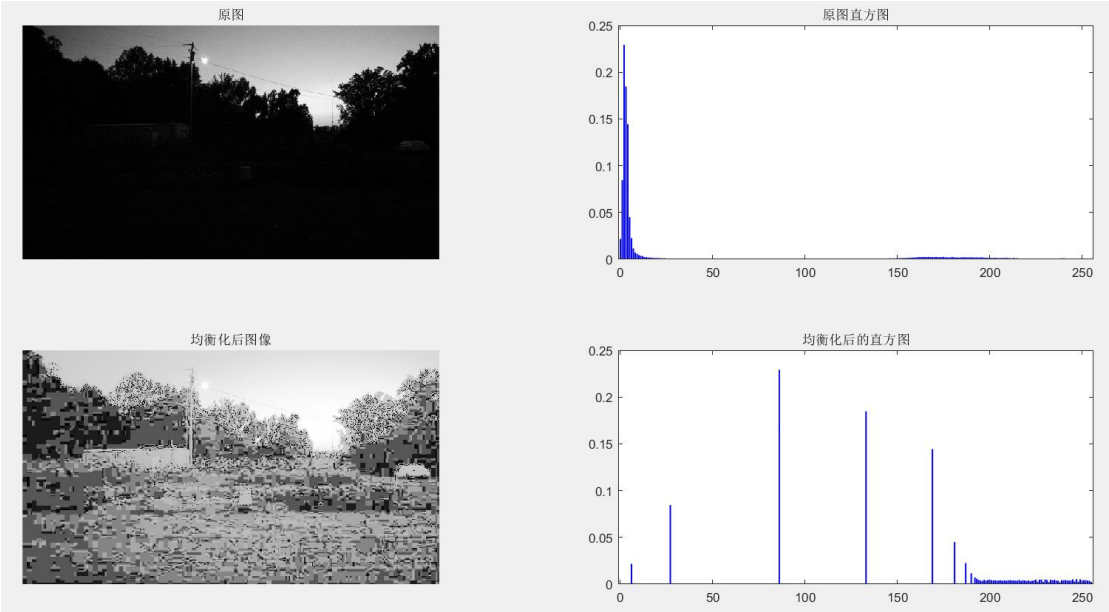
代码：

```
>> % 输入图像
A = imread('C:\Users\华为\OneDrive\图片\本机照片\作业.jpg');

%将原图转换为灰度图像
H = im2gray(A);    %从 RGB 创建灰度图
imwrite(H,'C:\Users\华为\OneDrive\图片\本机照片\作业灰度图.jpg');
```

```
[m,n]=size(H);              % 计算图像的长宽
p=zeros(1,256);         %创建数组存储像素概率
% 统计每个像素值出现的概率，  得到概率直方图
for i=0:255
    % 用 length 函数计算相同像素的个数
    p(i+1)=length(find(H==i))/(m*n);
end

%输出原图
subplot(2,2,1);          %用 subploy 函数将多个图像画到同一个平面中
imshow(H);                 %输出图像
title('原图');
%画出原图直方图
subplot(2,2,2);
bar(0:255,p,'b');
title('原图直方图');

% 求累计概率，得到累计直方图
s=zeros(1,256);
for i=1:256
        for j=1:i
            s(i)=p(j)+s(i);
        end
end

%完成每个像素点的映射
a=round(s*255);
b=H;
for i=0:255
        b(H==i)=a(i+1);
end

%输出均衡化后的图像
subplot(2,2,3);
imshow(b)
title('均衡化后图像');

for i=0:255
      GPeq(i+1)=sum(p(a==i));
end
%画出均衡化后的直方图
subplot(2,2,4);
bar(0:255,GPeq,'b');
title('均衡化后的直方图');
```

>>

实验结果：



代码：

```
function new_img = Expand( img )
    [height, width] = size(img);
    max_pixel = max(max(img));
    min_pixel = min(min(img));
    new_img=zeros(height,width);
    for i = 1 : height
        for j = 1 : width
            new_img(i, j) = 255 * (img(i, j) - min_pixel) / (max_pixel - min_pixel);
        end
    end
    new_img = uint8(new_img);
end

function H = HomomorphicFiltering( gamma_H, gamma_L, c, D0, height, width )
    for i = 1 : height
        x = i - (height / 2);
        for j = 1 : width
            y = j - (width / 2);
            H(i, j) = (gamma_H - gamma_L) * (1 - exp(-c * ((x ^ 2 + y ^ 2) / D0 ^ 2))) +
gamma_L;
        end
    end
end

%读入图片
img = imread('C:\Users\华为\OneDrive\图片\本机照片\作业.jpg');
```

```
img = im2gray(img);
figure(1);
subplot(2, 1, 1);
imshow(img);
title('Raw Image');
gamma_H = 2;
gamma_L = 0.25;
c = 0.25;
D0 =200;
f = double(img);
f = log(f + 1);%取指数
F = fft2(f);%傅里叶变换
F=fftshift(F);%频谱搬移
[height, width] = size(F);
%设计一个同态滤波器
H = HomomorphicFiltering(gamma_H, gamma_L, c, D0, height, width);
g=H.*F;%同态滤波
g = ifft2(ifftshift(g));%频谱搬移,傅里叶逆变换
g = exp(g)-1;
g = real(g);
%拉伸像素值
new_img = Expand(g);
subplot(2,1,2);
imshow(new_img);
title('Homomorphic Filtered Image');
```

比较：

对原图进行直方图均衡化，虽然能通过重新分布图像的灰度值使图像更明亮，但也明显放大了噪声

同态滤波增加了图像的亮度，也在一定程度上减少了噪声，但也没有完全，但因其增加了低灰度的比例，增强效果仍然不明显。