云南大学

本科实验报告

课程名称：　图像理解与计算机视觉

实验名称：　实验二. 图像滤波实验

学院（系）：　信息学院

专　　业：

班　　级：

姓　　名：

学　　号：

指导教师：

成　　绩：

评　　语：

Steven

一. 实验目的

通过编程实现使学生熟悉常用的空域滤波器，能够采用空域滤波技术对图像进行平滑和锐化的处理；使学生熟悉常用的频域滤波器，并采用频域滤波技术对图像进行平滑和锐化处理。

二. 实验内容

（1）为下图加入椒盐噪声，采用中值滤波对其进行平滑操作，并将平滑结果与原图对比；

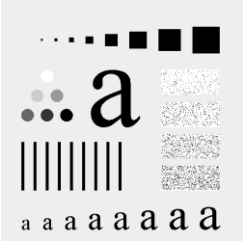
（2）采用四邻域拉普拉斯锐化法对下图进行空域锐化滤波；

（3）对下图进行傅里叶变换并显示其频谱图，接着将其傅里叶逆变换结果与原图对比；

（4）绘制高斯低通滤波器的透视图；

（5）采用高斯低通滤波器（截止半径分别为10、60、160）对下图进行频域

平滑滤波；



（6）采用频域拉普拉斯滤波法对题目（2）图像进行频域锐化滤波，并将频域锐化结果与题目（2）空域锐化结果进行对比。

三. 实验环境

Matlab软件是图像处理领域广泛使用的仿真软件之一。本实验基于Matlab 2022版本完成。

四. 实验代码（详细注释，Times New Roman/宋体 五号字体 单倍行距）

批注 [短腿1]: 和实验 1 一样，代码文件和文档中的代码有些区别，建议以源文件内容为准

```
function Exp2(func, show)
    % 读所有的图像
    img_rose = imread("rose.png");
    img_rose = rgb2gray(img_rose);
    img_lena = imread("lena.jpg");
    img_lena = rgb2gray(img_lena);
    img_moon = imread("moon.tif");
    img_letter = imread("letterA.tif");

    % 选择功能
    if func == "mediumFilter"
        %加入椒盐噪声
        img_rose_noise = imnoise(img_rose, 'salt & pepper');
        img_rose_mydenoise = mediumFilter(img_rose_noise, 5); % 第 2 个参数是卷积核大小
        img_rose_denoise = medfilt2(img_rose_noise, [5, 5]); % 第 2 个参数是卷积核大小
        if show == "show"
            subplot(2, 2, 1);
            imshow(img_rose);
            title("原图");
            subplot(2, 2, 2);
            imshow(img_rose_noise);
            title("添加椒盐噪声");
            subplot(2, 2, 3);
```

```
            imshow(img_rose_denoise);
            title("自带-中值滤波");
            subplot(2, 2, 4);
            imshow(img_rose_mydenoise);
            title("手动-中值滤波");
        end
    elseif func == "4_laplace_filter"
        alpha = input("输入锐化强度:");
        img_moon_mydenoise = four_laplace_filter(img_moon, alpha);
        img_moon_denoise = imfilter(img_moon, fspecial('laplacian', 0.2)); % 系统自带的
函数中，强度 alpha 范围为[0,1]
        if show == "show"
            subplot(1, 3, 1);
            imshow(img_moon);
            title("原图");
            subplot(1, 3, 2);
            imshow(img_moon_denoise);
            title("自带-拉普拉斯滤波");
            subplot(1, 3, 3);
            imshow(img_moon_mydenoise);
            title("手动-拉普拉斯滤波");
        end
    elseif func == "fourier"
        fourier_(img_lena);
    elseif func == "disp_GLPF_func"
        disp_GLPF_func()
    elseif func == "GLPF"
        letter_1 = GLPF(img_letter, 10);
        letter_2 = GLPF(img_letter, 60);
        letter_3 = GLPF(img_letter, 160);
        if show == "show"
            subplot(2, 2, 1);
            imshow(img_letter);
            title("原图");
            subplot(2, 2, 2);
            imshow(letter_1);
            title("高斯低通滤波-阈值 10");
            subplot(2, 2, 3);
            imshow(letter_2);
            title("高斯低通滤波-阈值 60");
            subplot(2, 2, 4);
            imshow(letter_3);
            title("高斯低通滤波-阈值 160");
        end
```

```
    elseif func == "freq_laplace_filter"
        img1=four_laplace_filter(img_moon, 0.1);
        img2=freq_laplace_filter(img_moon);
        if show == "show"
            subplot(1, 3, 1);
            imshow(img_moon);
            title("原图");
            subplot(1, 3, 2);
            imshow(img1);
            title("（空域）四邻域拉普拉斯滤波");
            subplot(1, 3, 3);
            imshow(img2);
            title("频域拉普拉斯滤波");
        end
    end
end

function img_new = mediumFilter(img, kernel_size)
    img_new = img;
    [row_length, col_length] = size(img);
    % 遍历
    for row_start = 1:row_length - kernel_size + 1
        for col_start = 1:col_length - kernel_size + 1
            % 取子块
            kernel = img(row_start:row_start + kernel_size - 1, col_start:col_start +
kernel_size - 1);
            kernel = sort(kernel);
            key_gray = kernel(round(kernel_size * kernel_size / 2)); % 取中间值
            img_new(row_start + fix(kernel_size / 2), col_start + fix(kernel_size / 2)) =
key_gray; % 填色
        end
    end
end

function img_new = four_laplace_filter(img, alpha)
    img_new = img;
    [row_length, col_length] = size(img);
    % 遍历
    for row_start = 1:row_length - 3
        for col_start = 1:col_length - 3
            % 计算中间点的位置
            r = row_start + 1;
            c = col_start + 1;
```

```
                img_new(r, c) = (1 + 4 * alpha) * img(r, c) - alpha * (img(r - 1, c) + img(r + 1, c)
+ img(r, c - 1) + img(r, c + 1)); %  按公式填色
            end
        end
end

function img_new = fourier_(img)
        %  对图像进行傅里叶变换并显示频谱图
        f = fft2(img);
        f1 = log(abs(f) + 1); %原始频谱图
        f2 = fftshift(f);
        f3 = log(abs(f2) + 1); %中心化后的频谱图
        %  频域逆变换到空间域
        img_new = real(ifft2(ifftshift(f2)));
        img_new = im2uint8(mat2gray(img_new));

        subplot(2, 2, 1);
        imshow(img);
        title('原图');
        subplot(2, 2, 2);
        imshow(f1, []);
        title('原始频谱图');
        subplot(2, 2, 3);
        imshow(f3, []);
        title('移动至频谱图中心');
        subplot(2, 2, 4);
        imshow(img_new);
        title('逆傅里叶变换');
end

%  返回高斯低通滤波的透视图
function disp_GLPF_func()
        %  生成高斯低通滤波器
        [x, y] = meshgrid(-127:128, -127:128);
        D0 = 50;
        H = exp(- (x.^2 + y.^2) / (2 * D0^2));
        %  生成透视图
        figure;
        mesh(x, y, H);
        title('高斯低通滤波器的透视图');
end

function [image_result] = GLPF(image_2zhi, D0)
        image_fft = fft2(image_2zhi); %用傅里叶变换将图象从空间域转换为频率域
```

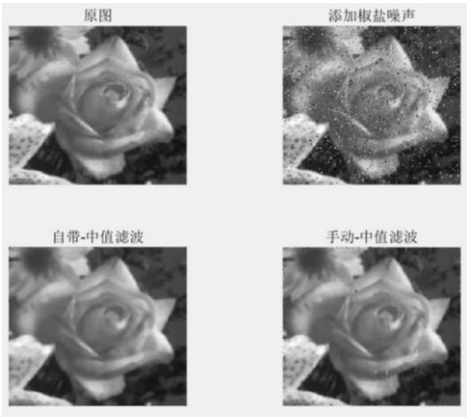
```
    image_fftshift = fftshift(image_fft);
    %将零频率成分（坐标原点）变换到傅里叶频谱图中心
    [width, high] = size(image_2zhi);
    D = zeros(width, high);
    %创建一个 width 行，high 列数组，用于保存各像素点到傅里叶变换中心的距离
    for i = 1:width

        for j = 1:high
            D(i, j) = sqrt((i - width / 2)^2 + (j - high / 2)^2);
            %像素点（i,j）到傅里叶变换中心的距离
            H = exp(-1/2 * (D(i, j).^2) / (D0 * D0));
            %高斯低通滤波函数
            image_fftshift(i, j) = H * image_fftshift(i, j);
            %将滤波器处理后的像素点保存到对应矩阵
        end

    end

    image_result = ifftshift(image_fftshift); %将原点反变换回原始位置
    image_result = uint8(real(ifft2(image_result)));
end

%  频域拉普拉斯滤波
function new_img=freq_laplace_filter(img)
    %读入图像，并转换为 double 型
    %获得图像的高度和宽度
    img=im2double(img);
    [M, N] = size(img);
    %图像中心点
    M0 = M / 2;
    N0 = N / 2;
    J = fft2(img);
    J_shift = fftshift(J);
    %%%%%%%%================高频提升（拉普拉斯算子）
============================
    %参数 A>=1,当其等于 1 时，为普通的高通滤波器
    A = 2;
    for x = 1:M
        for y = 1:N
            %计算频率域拉普拉斯算子
            h_hp = 1 + 4 * ((x - M0)^2 + (y - N0)^2) / (M0 * N0);
            h_bp = (A - 1) + h_hp;
            J_shift(x, y) = J_shift(x, y) * h_bp;
        end
```

```
    end
    J = ifftshift(J_shift);
    new_img = ifft2(J);
end
```

附件（.m文件）：Exp2.m

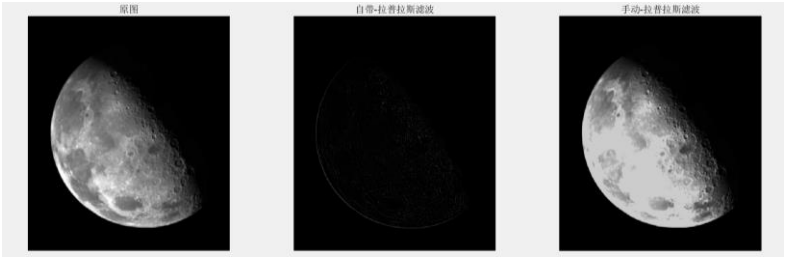
五. 实验结果及分析

（1）原图、加入椒盐噪声的图像、中值滤波结果



（手写的中值滤波去噪效果没有库函数的效果好，大体上还算说得过去）

（2）原图、四邻域拉普拉斯锐化结果

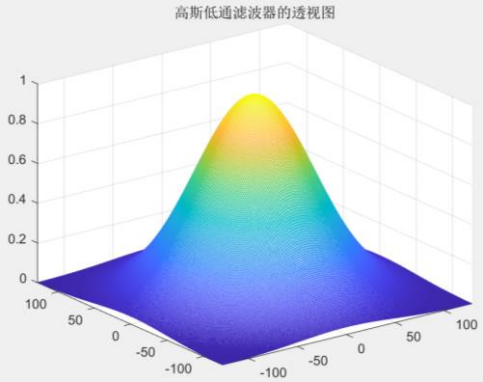


（此时拉普拉斯滤波效果比较强，只能显示一些比较明显的特征，两者的α均为0.2）

（3）原图、傅里叶变换频谱图、傅里叶逆变换图像

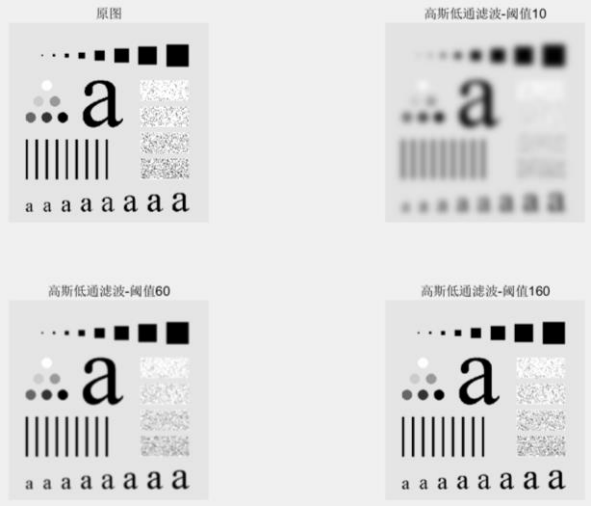
（两图并无差别，因为只是傅里叶变换然后再逆变换，并未进行滤波等操作）

（4）高斯低通滤波器的透视图



（低通滤波的传递函数如图所示，对于中间的低频部分给予较高的权重，而高频部分较低的权重，以实现低通滤波）

（5）原图、高斯低通滤波结果（截止半径分别为10、60、160）



（不同阈值下的平滑滤波，其效果是不同的。对于低通滤波，其阈值越低，增强的频率范围越小，抑制的频率范围越大，则平滑效果越明显）

（6）原图、频域拉普拉斯锐化结果、空域拉普拉斯锐化结果

（空域拉普拉斯滤波相对来说作用没那么明显，但频域拉普拉斯滤波作用效果太过了使得丢失了不少细节信息，也称不上效果多好）

六. 实验体会

本次实验主要验证、应用了滤波技术，包括空域和频域滤波两种方式的多种方法。

这次实验中最难理解的是傅里叶变换，以及变换后的频域内的概念，因为之前对图像的认识全部建立在二维矩阵和离散像素点的基础上，从没考虑过图像中还有频率、能量的概念，一时间感觉非常割裂。课下花了不少时间才逐渐理解，频率是描述图像整体属性的一种属性，不过也因此对图像有了另一角度的理解。