北京大學

本科实验报告

课程名称：	机器感知实验
姓　　名：	金镇雄
学　　院：	元培
系：	智能科学系
专　　业：	智能科学与技术
年　　级：	19
学　　号：	1900094619
指导教师：	曲天书
职　　称：	副教授

2022 年　　5 月　　4 日

实验五、图像增强

一、 实验目的

- 使用 MATLAB 进行图像增强实验
- 掌握平滑滤波器的算法原理
- 掌握图像中值滤波的算法原理
- 掌握频域增强的基本原理

二、 实验要求

- 界面清晰美观
- 可显示原始图像和增强后的图像
- 实验结果分析
- 实验讨论

三、 实验原理

1. 线性平滑滤波器

   线性平滑滤波器也称为均值滤波器，这种滤波器的所有系数都是正数，对 3×3 的模板来说，最简单的是取所有系数为 1，为了保持输出图像任然在原来图像的灰度值范围内，模板与象素邻域的乘积都要除以 9。

2. 中值滤波器

   中值滤波器其滤波是把邻域中的图像的象素按灰度级进行排序，然后选择改组的中间值作为输出象素值。

3. 低通滤波

   图像的能量大部分集中在幅度谱的低频和中频部分，而图像的边缘和噪声对应于高频部分。因此能降低高频成分幅度的滤波器就能减弱噪声的影响。由卷积定理，在频域实现低通滤波的数学表达式：

$$\mathrm{G(u,v)} = \mathrm{H(u,v)F(u,v)}$$

   A. 理想低通滤波器

$$\mathrm{H(u,v)} = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

   $D(u,v)$是从点(u,v)到频率平面的原点的距离；D_0为理想低通滤波器的截止频率。

   B. 巴特沃斯低通滤波器（BLPF）

$$\mathrm{H(u,v)} = \frac{1}{1 + (\sqrt{2}-1)\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$

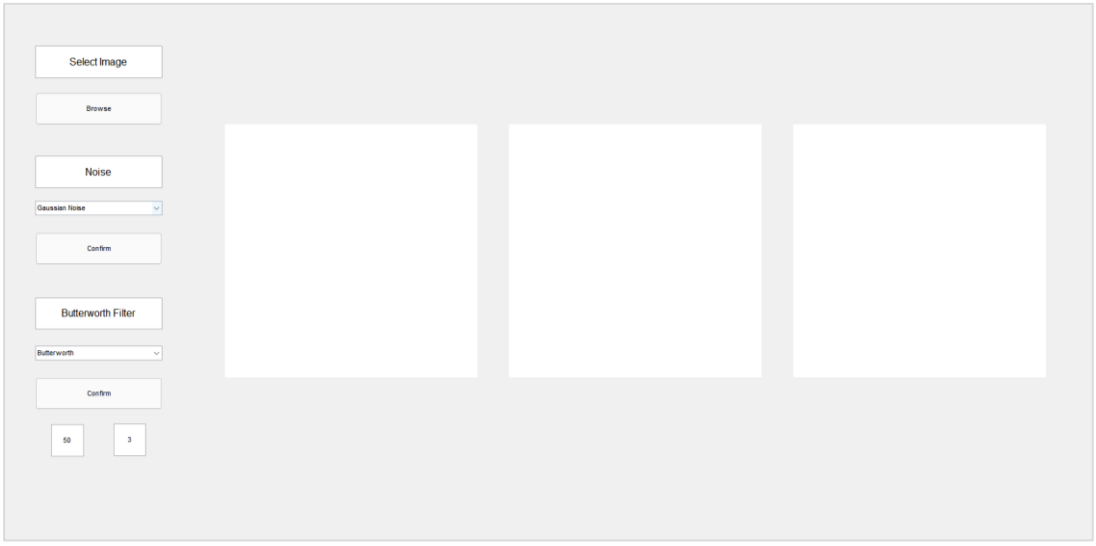
四、 实验仪器设备

PC 机、MATLAB 程序、图像文件

五、 实验内容和步骤

1. GUI 界面设计

在 GUI 界面中可选定想读入的原始图像、添加的噪声以及对含噪图像进行增强时使用的滤波器。对于理想低通滤波器可输入截止频率D_0，对于巴特沃斯低通滤波器可输入截止频率D_0和阶数 n。在界面中共有三个坐标轴控件分别可显示原始图像、含噪图像和增强图像（从左到右）。



2. 读入原始图像
3. 添加噪声

可添加的噪声有高斯噪声、椒盐噪声和 speckle 噪声，其特点如下：

A. 高斯噪声：概率密度函数服从高斯分布的一类噪声。

B. 椒盐噪声：椒盐噪声是由图像传感器，传输信道，解码处理等产生的黑白相间的亮暗点噪声。

C. speckle 噪声：又叫斑点噪声，图像上表现为信号相关（如在空间上相关）的小斑点。

可使用 Image Processing Toolbox（图像处理工具箱）的 imnoise 函数直接向原始图像加入相应噪声。

4. 分别用平滑滤波器、中值滤波器和低通滤波器对含噪图像进行增强

定义四种滤波器对应的函数，四个函数都以含噪图像为输入，以增强后的图像为输出。两个低通滤波器对应的函数有多个输入：理想低通滤波器的输入参数，除了含噪图像以外，还包括截止频率；巴特沃斯低通滤波器的输入包括截止频率和阶数。

5. 显示增强后的图像

六、实验结果和分析

1. 原始图像（lena.tif）

图一 原始图像

2. 添加噪声

图二 原始图像

图三 含高斯噪声

图四 含椒盐噪声

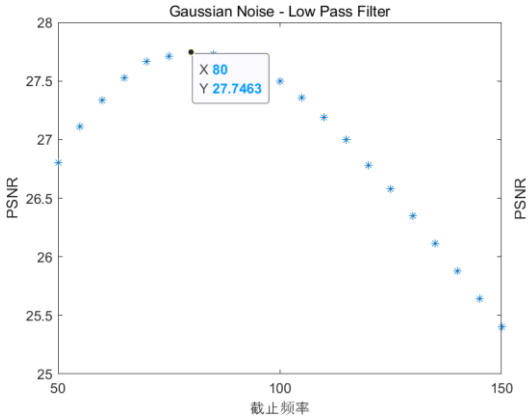
图五 含 speckle 噪声

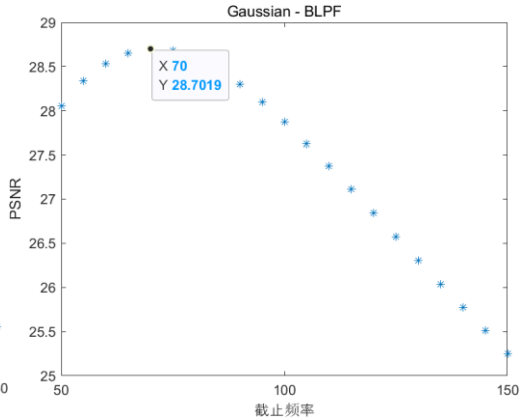
3. 低通滤波器参数设定

实验结果发现低通滤波器的截止频率影响滤波效果，截止频率过小时增强图像模糊不清而过大时与含噪图像一致，因此需要选定合适的截止频率。

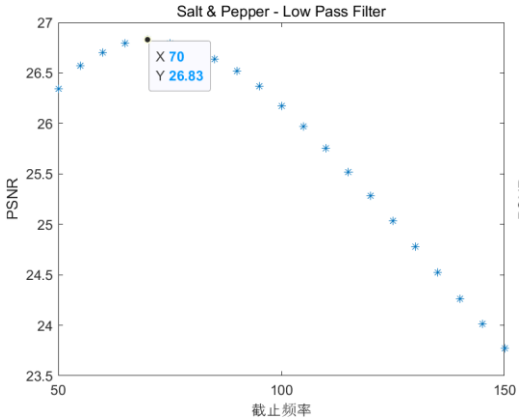
设定方法为枚举法： 一定范围内枚举截止频率，通过低通滤波器得到增强图像后计算 PSNR 值。这里假定 PSNR 最大时其滤波效果最佳，因此取 PSNR 值最大时对应的截止频率。

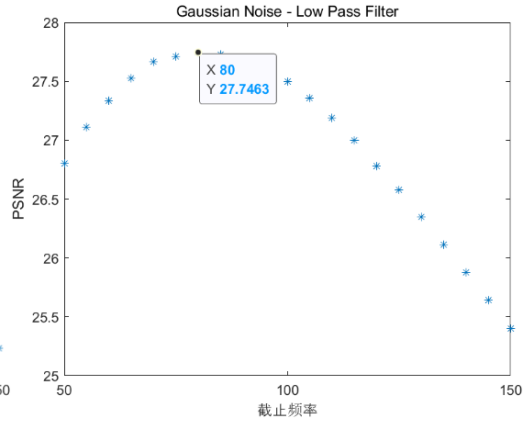
A. 噪声为高斯噪声时，截止频率与增强图像的 PSNR 值间的关系如下：



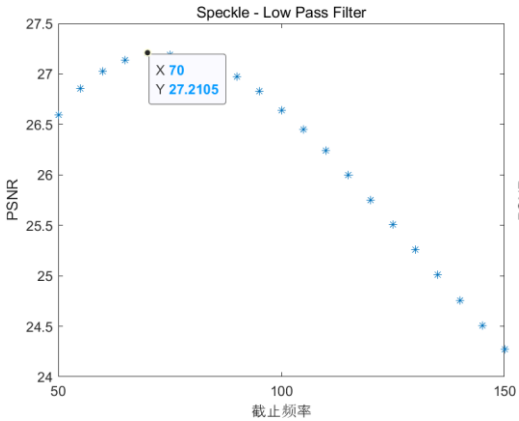


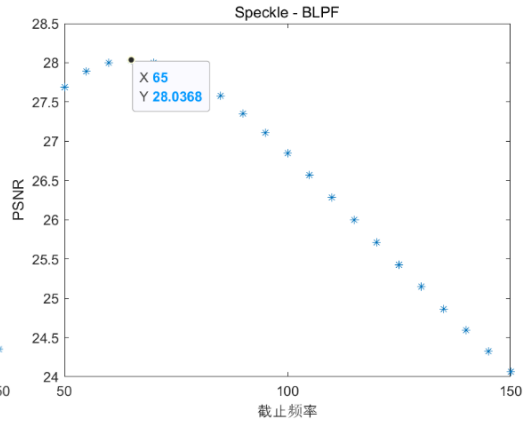
B. 噪声为椒盐噪声时，截止频率与增强图像的 PSNR 值间的关系如下：





C. 噪声为 Speckle 噪声时，截止频率与增强图像的 PSNR 值间的关系如下：



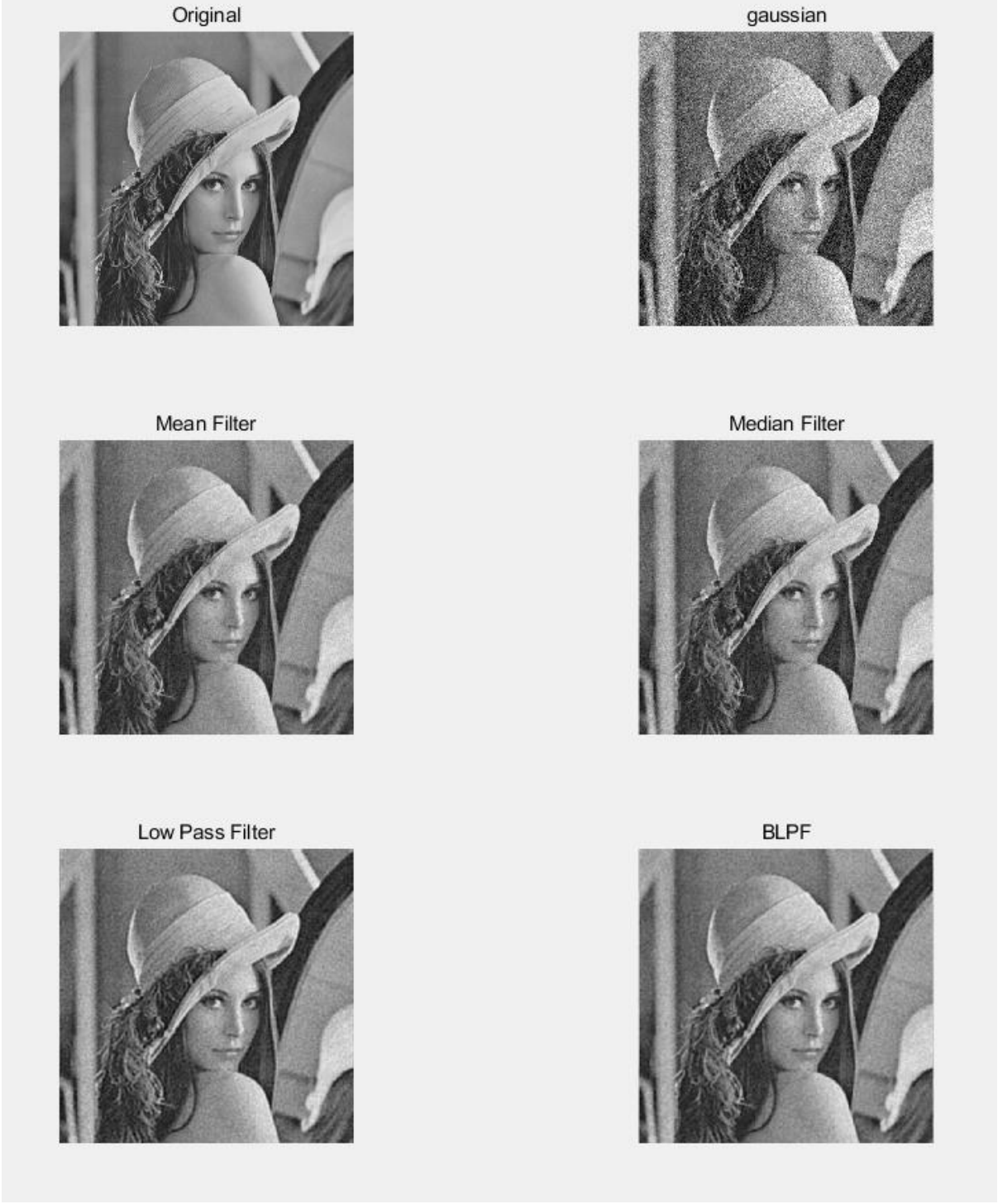


4. 以下是 5% 高斯噪声的去除效果：

理想低通滤波器的截止频率为 80。

巴特沃斯低通滤波器的截止频率和阶数分别为 70 和 3。

可见，四种滤波器都有明显的去噪效果，且均值滤波和低通滤波的效果优于中值滤波的效果。

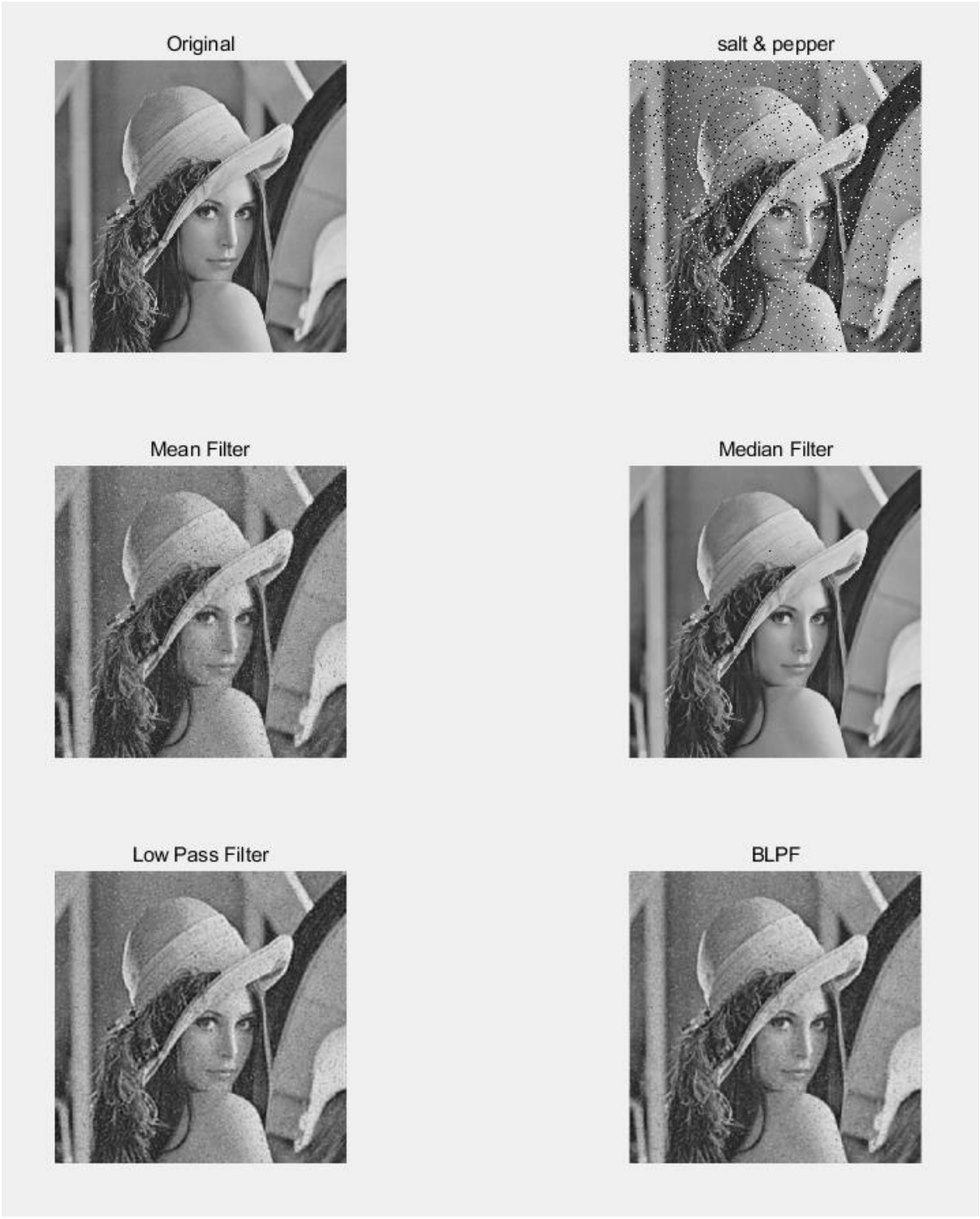


5. 以下是 5% 椒盐噪声的去除效果：

理想低通滤波器的截止频率为 70。

巴特沃斯低通滤波器的截止频率和阶数分别为 65 和 3。

可见，中值滤波的去噪效果最优，其失真可以察觉但可以接受。



6. 以下是 Speckle 噪声的去除效果：

理想低通滤波器的截止频率为 80。

巴特沃斯低通滤波器的截止频率和阶数分别为 70 和 3。

与去除高斯噪声的结果类似，四种滤波器都有去噪效果，且均值滤波和低通滤波的效果优于中值滤波的效果。

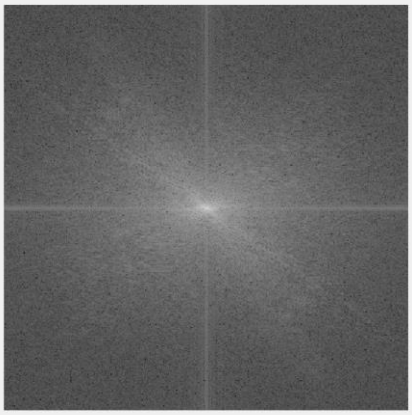
7. 低通滤波器频域分析

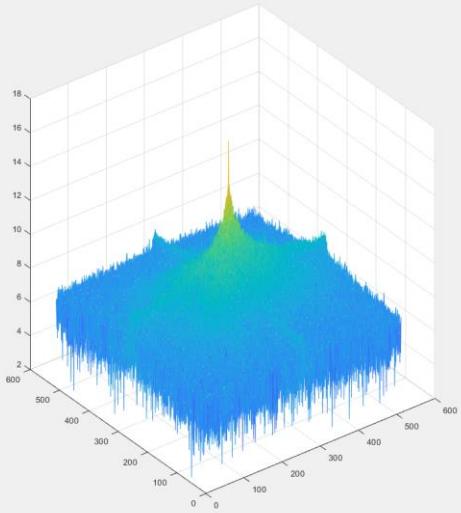
图像二维频谱图通过对输入图像进行水平和竖直两个方向的所有扫描线的一维傅立叶变换进行叠加得到，用来表示输入图像的频率分布。

频谱图以图像的中心为圆心，圆的相位对应原图中频率分量的相位，半径对应频率高低。低频半径小，高频半径大，中心为直流分量，某点的灰度值对应该频率的能量高低。

A. 原始图像的频谱图和其三维透视图如下：

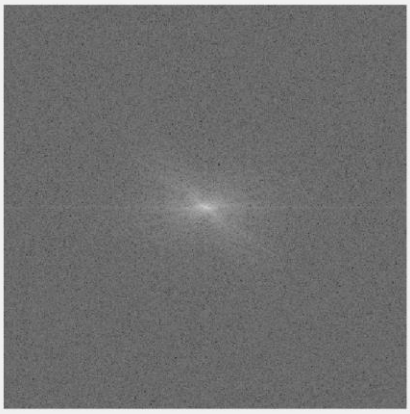
原始图像的能量主要分布于中心处，即低频部分。

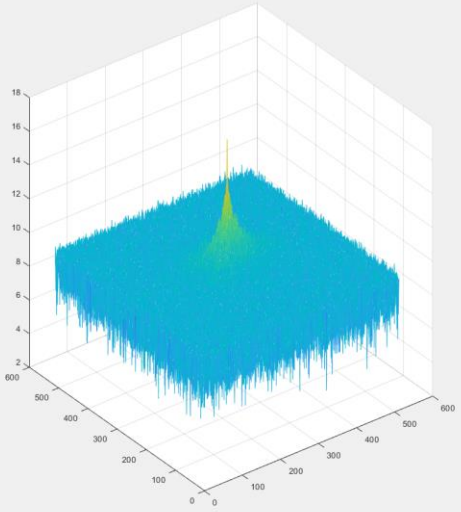




B. 含噪图像的频谱图和其三维透视图如下：

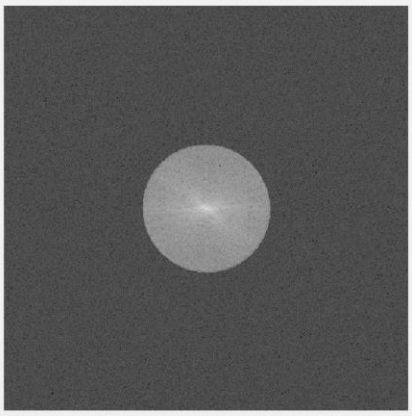
加入高斯噪声后可发现低频处几乎与原始图像一致，而高频处的能量有所增加。

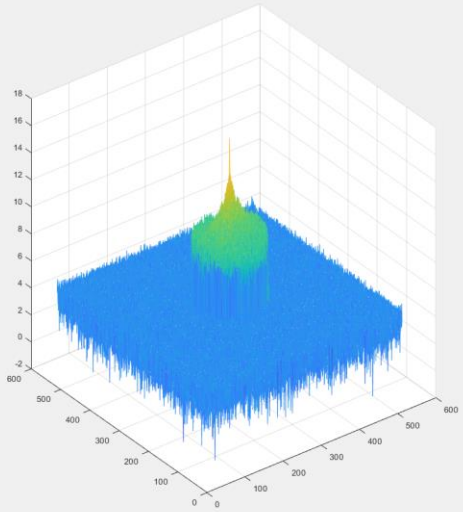




C. 理想低通滤波器去噪图像的频谱图和其三维透视图如下：

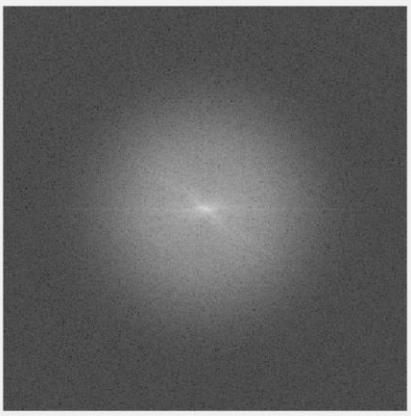
可见，理想低通滤波器在以原点为圆心，截止频率D_0为半径的圆内，通过所有的频率，而在圆外截断所有的频率。由于加入到的噪声占高频，低通滤波后可去除噪声。

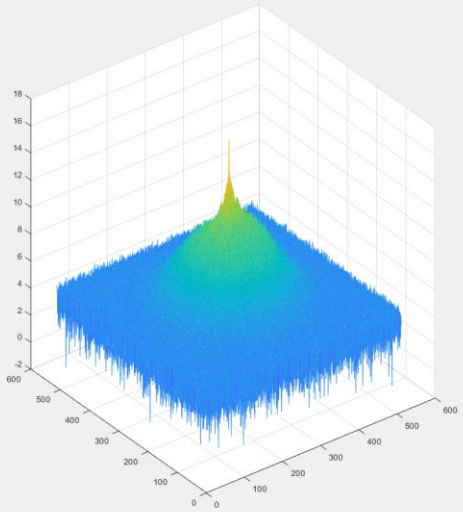




D. 巴特沃斯低通滤波器去噪图像的频谱图和其三维透视图如下：

可见，巴特沃斯低通滤波器去除了含噪图像的高频部分，而其过渡没有理想低通滤波器那么剧烈。





七、讨论

1. PSNR

PSNR(Peak Signal to Noise Ratio)，峰值信噪比，即峰值信号的能量与噪声的平均能量之比，通常表示的时候取 log 变成分贝（dB）。其定义式如下：

$$\text{PSNR} = 10 \times log_{10}\left(\frac{MAXI^2}{MSE}\right) = 10 \times log_{10}\left(\frac{(2^n-1)^2}{MSE}\right)$$

其中 $MAXI$ 表示图像点颜色的最大数值，如果每个采样点用 8 位表示，那么就是 255；MSE 为均方误差，表达两幅图在每一个位置上的像素值的差异的平均，即两个图像中每一个相同位置的像素值相减，平方，求和，再求平均。两个 $m \times n$ 单色图像 I 和 K，若一个为另外一个的噪声近似，则它们的均方误差定义为：

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left||I(i,j) - K(i,j)\right||^2$$

从其定义可见，PSNR 值越大，就代表失真越少，含噪图像越接近原始图像。PSNR 值高于 40dB 说明图像质量极好（非常接近原始图像）；高于 30dB 低于 40dB 说明失真可以察觉但可以接受；高于 20dB 低于 30dB 说明图像质量差；低于 20dB 说明图像不可接受。

需要注意，PSNR 具有局限性。PSNR 是最普遍和使用最为广泛的一种图像客观评价指标，然而它是基于对应像素点间的误差，即基于误差敏感的图像质量评价。由于并未考虑到人眼的视觉特性，因而会出现评价结果与人的主观感觉不一致的情况。

2. 以下是实验结果中加入高斯噪声、椒盐噪声和 Speckle 噪声后的含噪图像以及分别进行各种滤波后得到的增强图像对应的 PSNR 值：

A. 加入高斯噪声时，含噪图像和四种增强图像的 PSNR 如下：

Image	Gaussian Noise	Mean Filter	Median Filter	Low Pass Filter	Butterworth Low Pass Filter
PSNR	20.0684	28.1513	26.7674	27.7386	28.6810

B. 加入椒盐噪声时，含噪图像和四种增强图像的 PSNR 如下：

Image	Salt & Pepper Noise	Mean Filter	Median Filter	Low Pass Filter	Butterworth Low Pass Filter
PSNR	18.4450	26.7183	34.8156	26.8125	27.5684

C. 加入 Speckle 噪声时，含噪图像和四种增强图像的 PSNR 如下：

Image	Speckle Noise	Mean Filter	Median Filter	Low Pass Filter	Butterworth Low Pass Filter
PSNR	18.8592	27.2492	24.0352	27.1669	28.0200

可见，与实验结果一致，四种滤波器都有去噪效果，都一定程度上提高了 PSNR 值。图像含高斯噪声和 Speckle 噪声时，均值滤波和低通滤波的 PSNR 值高于中值滤波，有更优的滤波效果；图像含椒盐噪声时，中值滤波的 PSNR 值明显高于其余滤波的 PSNR 值。

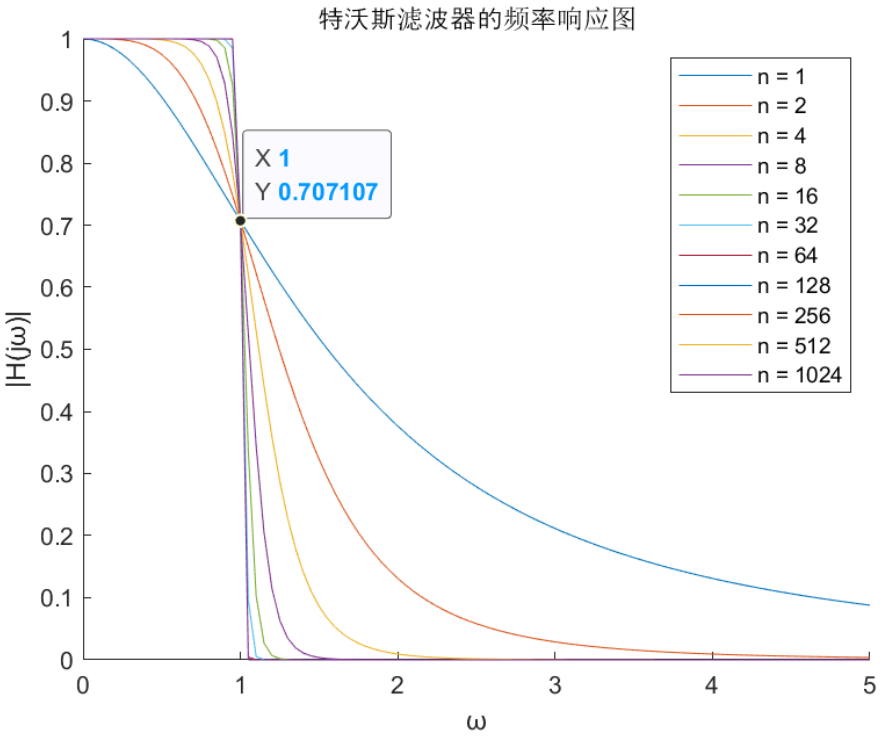
3. 巴特沃斯低通滤波器与理想低通滤波器

从巴特沃斯低通滤波器的滤波函数

$$H(u,v) = \frac{1}{1 + (\sqrt{2} - 1)\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$

可知其阶数 n趋于无穷大时，若频率 $D(u,v)$大于截止频率D_0，则函数值趋于 0；若 $D(u,v)$小于D_0，则函数值趋于 1。这是理想低通滤波器的函数特性，可见，当巴特瓦斯低通滤波器的阶数趋于无穷大时，几乎接近于理想低通滤波器。

下图为巴特沃斯滤波器的频率响应图：



上图中曲线的交点处的横坐标表示截止频率，纵坐标表示其增益。此外，也可发现当巴特瓦斯低通滤波器的阶数较高时，接近于理想低通滤波器。

4. 中值滤波去除椒盐效果最优的原因

由于椒盐噪声是阶跃脉冲噪声（取值 0 或 255 且小概率出现），中值滤波是去中位数，不会被阶跃值影响，所以差点儿能全然过滤掉阶跃脉冲噪声。而均值滤波处理阶跃值时分配权重不会变化。那么求平均值时受阶跃值影响而产生的误差就较大。因此效果不理想。

5. 均值滤波和低通滤波去除高斯噪声效果更优的原因

高斯噪声是图像中每一个像素点都从原灰度值依据高斯分布做随机噪声。那么选取中值的代表意义并不大，由于各个像素都是独立同分布的。中值滤波相当于在模版内再选出了一个经过高斯噪声变换后的灰度值，滤波效果和噪声图像没有明显改善，所以比较之下均值滤波较优。

八、代码

```
function varargout = lab5(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @lab5_OpeningFcn, ...
                   'gui_OutputFcn',  @lab5_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function lab5_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
init_bg();
global res;
axes(handles.axes1);
imshow(res);
axes(handles.axes2);
imshow(res);
axes(handles.axes3);
imshow(res);

function varargout = lab5_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function imgtxt_Callback(hObject, eventdata, handles)
```

```
function imgtxt_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in select_img.
function select_img_Callback(hObject, eventdata, handles)
global img
[file,path] = uigetfile('*.tif');
str = path + "/" + file;
img = imread(str);
set(handles.imgtxt,'String',file);
axes(handles.axes1)
imshow(img)

function param1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function param1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'Visible','off')



function param2_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function param2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'Visible','off')



function noisetxt_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
```

```
function noisetxt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in noise_confirm.
function noise_confirm_Callback(hObject, eventdata, handles)
global noise
global img
m = get(handles.noise_menu,'Value');
if m == 1
    noise = imnoise(img,'gaussian');
elseif m == 2
    noise = imnoise(img,'salt & pepper');
else
    noise = imnoise(img,'speckle');
end
axes(handles.axes1)
imshow(img)
axes(handles.axes2)
imshow(noise)

function filtertxt_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function filtertxt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in filter_confirm.
function filter_confirm_Callback(hObject, eventdata, handles)
global noise
global img
global filtered
m = get(handles.filter_menu,'Value');
if m == 1
    filtered = meanFilter(noise);
elseif m == 2
    filtered = medianFilter(noise);
elseif m == 3
    x = str2double(get(handles.param1,'String'));
    filtered = lowpassFilter(noise,x);
else
```

```
    x = str2double(get(handles.param1,'String'));
    y = round(get(handles.param2,'String'));
    filtered = blpfFilter(noise,x,y);
end
axes(handles.axes1)
imshow(img)
axes(handles.axes2)
imshow(noise)
axes(handles.axes3)
imshow(filtered)

% --- Executes on selection change in noise_menu.
function noise_menu_Callback(hObject, eventdata, handles)
m = get(hObject,'Value');
if m == 1
    set(handles.noisetxt,'String','Gaussian Noise')
elseif m == 2
    set(handles.noisetxt,'String','Salt & Pepper Noise')
else
    set(handles.noisetxt,'String','Speckle Noise')
end

% --- Executes during object creation, after setting all properties.
function noise_menu_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in filter_menu.
function filter_menu_Callback(hObject, eventdata, handles)
m = get(hObject,'Value');

if m == 3           % low pass filter
    set(handles.param1,'Visible','on')
    set(handles.param2,'Visible','off')
    set(handles.param1,'String','70');
    set(handles.filtertxt,'String','Low Pass Filter');
elseif m == 4       % blpf filter
    set(handles.param1,'Visible','on')
    set(handles.param2,'Visible','on')
    set(handles.param1,'String','50');
    set(handles.param2,'String','3');
    set(handles.filtertxt,'String','Butterworth Filter');
else
    set(handles.param1,'Visible','off')
    set(handles.param2,'Visible','off')
    if m == 1
        set(handles.filtertxt,'String','Mean Filter');
```

```
    else
        set(handles.filtertxt,'String','Median Filter');
    end
end



% --- Executes during object creation, after setting all properties.
function filter_menu_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function init_bg()
global res;
res = uint8(zeros(500, 500, 3));
for i = 1:500
    for j = 1:500
        res(i,j,1:3)=[255,255,255];
    end
end

 function img_out = meanFilter(A)
img_out = A;
[R,C] = size(A);
for i = 1:R
    for j = 1:C
        u = max(i-1,1);
        d = min(i+1,R);
        l = max(j-1,1);
        r = min(j+1,C);
        img_out(i,j) = mean(mean(A(u:d,l:r)));
    end
end

function img_out = medianFilter(A)
img_out = A;
[R,C] = size(A);
for i = 1:R
    for j = 1:C
        u = max(i-1,1);
        d = min(i+1,R);
        l = max(j-1,1);
        r = min(j+1,C);
        a = A(u:d,l:r);
        a = a(:);
        img_out(i,j) = median(a);
    end
end
```

```
function img_out = lowpassFilter(A,d0)
imgFFT = fftshift(fft2(double(A)));
[R,C] = size(A);
r0 = round(R/2);
c0 = round(C/2);
img_out = zeros(R,C);
d0 = d0^2;
for i = 1:R
    tmp = zeros(1,C);
    for j = 1:C
        d = (i-r0)^2+(j-c0)^2;
        if d <= d0
            h = 1;
        else
            h = 0;
        end
        tmp(j) = h*imgFFT(i,j);
    end
    img_out(i,:) = tmp;
end

img_out = ifftshift(img_out);
img_out = uint8(real(ifft2(img_out)));

function img_out = blpfFilter(A,d0,n)
imgFFT = fftshift(fft2(double(A)));
[R,C] = size(A);
r0 = round(R/2);
c0 = round(C/2);
img_out = zeros(R,C);
d0 = d0^(2*n);
a = sqrt(2)-1;
for i = 1:R
    for j = 1:C
        d = (i-r0)^2+(j-c0)^2;
        h = 1 / (1+a*(d^n)/d0);
        img_out(i,j) = h*imgFFT(i,j);
    end
end

img_out =ifftshift(img_out);
img_out = uint8(real(ifft2(img_out)));



function psnr = PSNR(img,noise)
[n,m] = size(img);
img1 = double(img);
img2 = double(noise);
MAXI = 255;
MSE = sum(sum((img1-img2).^2))/(m*n);
psnr = 20*log10(MAXI/sqrt(MSE));
```