数字图像处理学习笔记

（MATLAB 实现）

前言

本学习笔记仅可供学习使用，不能用于商业制造盈利。

本学习笔记是个人的学习笔记，内容并不全面，仅供参考。

笔记中涉及的代码运行环境：win10　MATLAB2019a

作者：步平凡

参考资料：

[1] 数字图像处理的 MATLAB 实现（第 2 版） [美] Rafael C. Gonzalez Richard E. Woods Steven L. Eddins 著 阮秋琦 译

目录

一、MATLAB 前言

（一）图像读取与展示

1. 图像输入

```
f = imread(filename);
```

2. 图像展示

为了不覆盖已有图像，可以使用 figure 重开图像展示窗口。

```
figure, imshow(f);
```

3. 图像输出

```
imwrite(f, saveName, parameters);
```

（1）用法示例

```
imwrite(f, 'a.jpg', 'quality', q);  % 适用于 JPEG 图像，q 为 0~100 的整数，q 越大图像质量越高
imwrite(f, 'b.jpg', 'compression', 'parameter', 'resolution', [colres rowres]);     % 适用于 TIFF 图像
```

（2）参数说明

parameter: 'none'(无压缩), 'packbits'(非二值图像), 'lwz'、'deflate'、'jpeg'、'ccitt'(二值图像)、'fax3'(仅对二值图像)、'fax4'。

[colres rowres]: 表示每单位点数给出列分辨率和行分辨率，默认值为[72 72]。

（二）图像设定

```
title('图像标题');  设置标题
axis([xmin xmax ymin ymax]);                    设置图像坐标轴大小
axis ij             设置坐标原点在左上角
axis xy             设置坐标原点在左下角
axis off|on         关闭 | 打开坐标轴
grid off|on         关闭 | 打开网格(3D)
view(az, el)        观看角度，az 方位角、el 仰角(3D)
shading interp      去掉网格线，平滑曲面
colormap([r g b])   设置线框图颜色
colormap(gray)      将彩色转化为灰色
set(gca, 'xtick', start:step:end);              设置 x 轴标度（x=>y 类似）
xlabel('标题');     设置 x 轴标题（x=>y 类似）
xlim([xlim ylim] | 'auto')                      设置 x 轴坐标大小（x=>y 类似）
text(xloc, yloc, '文本', 'fontsize', size);     设置在特定位置的文本
```

（三）绘制 3D 图像

```
mesh(Z(1:k:end, 1:k:end));      绘制线框图
surf(Z(1:k:end, 1:k:end));      绘制表面图
```

（四）数据保存与加载

1. 数据保存 save

```
save(filename, variables, fmt)
save(filename, variables, '-append', '-nocompression')
```

参数说明

variables: var1, ..., varN | '-struct', structName | '-struct', structName, filed1, ..., filedN

fmt（文件类型）: '-mat' | '-ascii' | '-ascii', '-tabs' | '-ascii', 'double'

'-append': 数据追加到文件

'-nocompression': 数据存储无压缩

```
save test.mat
p = rand(1,10);
q = ones(10);
save('test.mat','p','q');              % 保存特定变量
save('test.txt','p','q','-ascii');     % 指定保存为 ASCII 文件
type('test.txt');                      % type 函数显示文件内容
r = 100;
save('test.mat','r','-append');        % 追加变量 r 到 test.mat 文件

s1.a = 12.7;
s1.b = {'abc',[4 5; 6 7]};
s1.c = 'Hello!';
save('newstruct.mat','-struct','s1'); % 保存结构体 s1
save('newstruct.mat','-struct','s1', a, b);    % 保存结构体 s1 的 a,b 两个字段
whos('-file','newstruct.mat');         % 查看文件内容
```

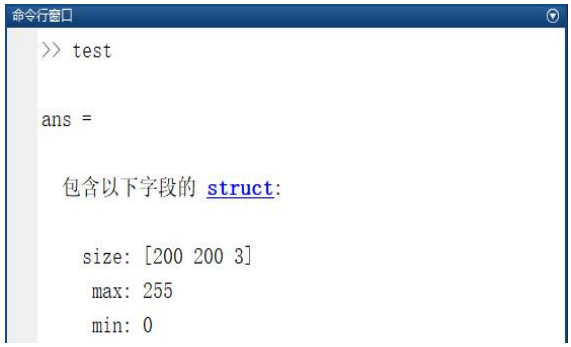
2. 数据加载 load

```
load(filename, ['-ascii' | '-mat',] variables);
    '-ascii' | '-mat': 指定文件类型
    variables: 变量列表
```

```
a = magic(4);
b = ones(2, 4) * -5.7;
c = [8 6 4 2];
save -ascii mydata.dat a b c     % 保存数据，文本格式
save mydata.mat                  % 保存数据，二进制格式
whos -file mydata.mat            % 查看数据内容
type mydata.dat                  % 查看数据内容
clear a b c
load mydata.dat -ascii           % 加载数据
load mydata.mat c                % 加载数据
```

（五）数据类型——结构体

```
% 测试结构体类型 structImage.m
function s = structImage(f)
            s.size = size(f);
            s.max = max(f(:));
            s.min = min(f(:));
        end
        % 测试结构体函数
        f = imread('Lena.jpg');
        structImage(f)
```



(六) M 函数文件

1. 文件名：与函数名相同

2. 格式：

```
function [返回参数] = 函数名(参数列表)
% H1 行
    % 帮助文本
    函数体
```

3. 函数句柄

```
函数句柄名 = @函数名;                  // 显式
函数句柄名 = @(参数列表) 函数表达式; // 隐式
```

```
f = @sin;
f(pi/4)
g = @(x) x.^2;
g([1 2 3])
```

4. 参数个数

```
in = nargin;    // 得到输入参数个数
out = nargout; // 得到输出参数个数
msg = nargchk(low, high, nuber);   // 检查参数个数
```

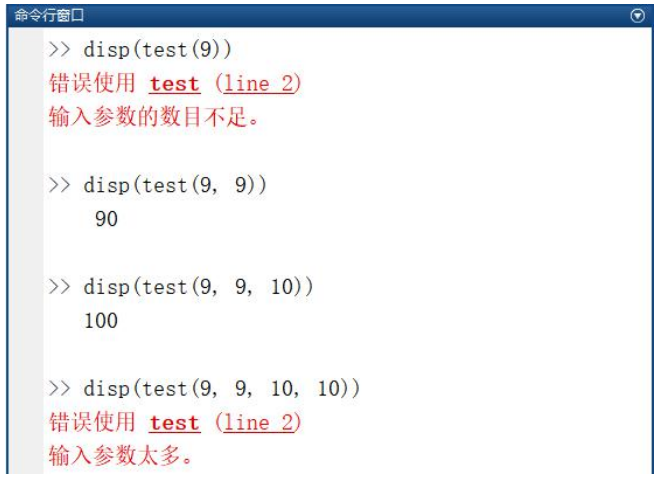
5. 可变个数参数

```
可变输入参数: varargin
可变输出参数: varargout
```

6. 错误提示

```
error(msg);     // 输出错误提示信息 msg 并终止程序
```

```
function y = test(m, varargin)
    error(nargchk(2, 3, nargin));
    if isempty(varargin)
        y = m.^2;
    elseif length(varargin) == 1
        y = m.^2 + varargin{1};
    else
        y = m.^2 + varargin{1} + varargin{2};
    end
end
```



（七）算子

1. 运算算子

运算符	名称
+	加法
-	减法
*	矩阵乘法
.*	数组乘法
/	矩阵右除
./	数组右除
\	矩阵左除
.\	数组左除
^	矩阵乘幂
.^	数组乘幂
’	矩阵转置
.’	数组转置
:	冒号

2. 关系算子

算子	名称
<	小于
<=	小于或等于
>	大于
>=	大于或等于
==	等于
~=	不等于

3. 逻辑算子

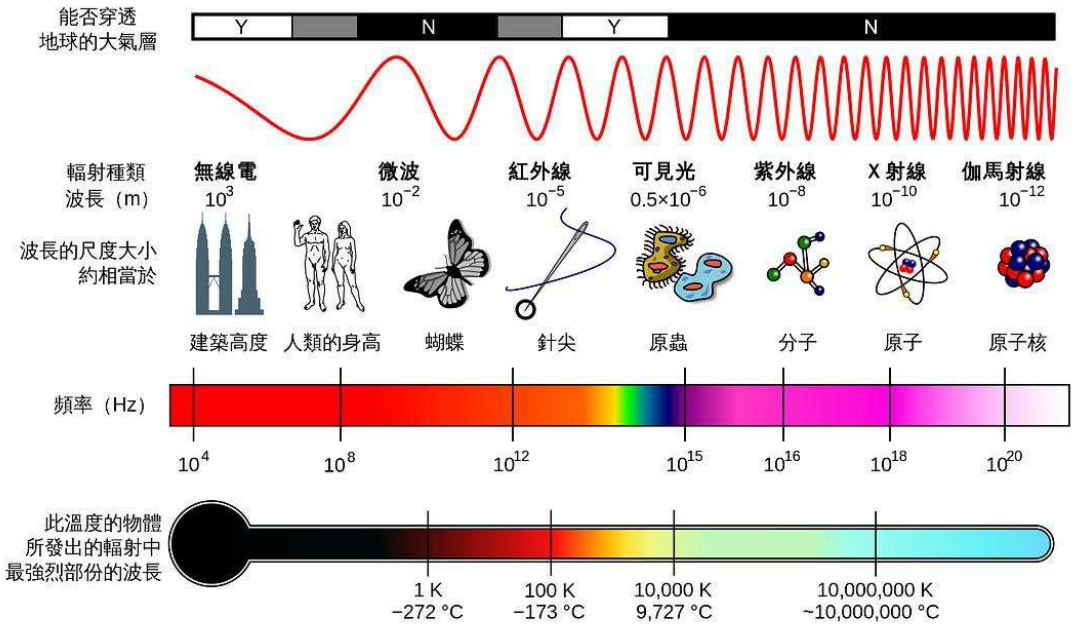
算子	名称
&	与
|	或
~	非
&&	标量与
||	标量或

4. 流程控制

算子	名称
if	逻辑条件判断，else、elseif
for	规定次数循环
while	逻辑条件循环
break	中止 for、while 循环
continue	跳过本次循环，进入下一次循环
switch	根据指定值或字符串执行语句，case、otherwise
return	结束函数并返回函数值
try...catch	检测执行错误

二、数字图像

（一）电磁波谱图



（二）数字图像

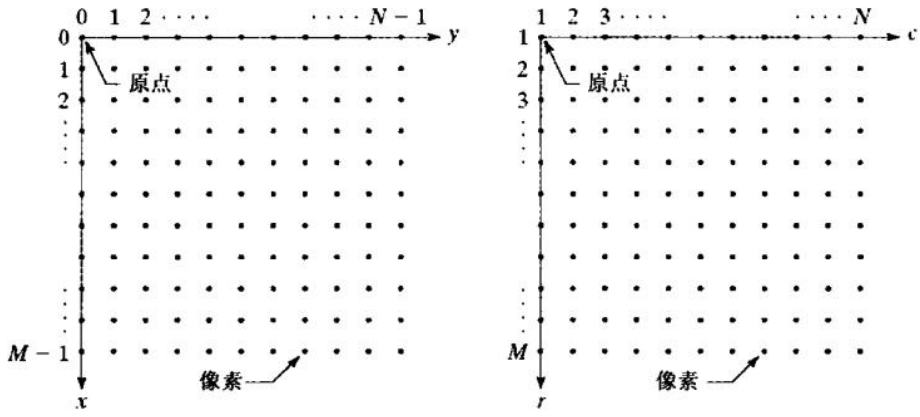
一幅图像可以定义为一个二维函数 f(x, y)，这里的 x 和 y 是空间坐标，而在任意坐标（x，y）处的幅度 f 被称为这一坐标位置图像的亮度或灰度。当 x、y 和 f 的幅值都是有限的离散值时，称图像为数字图像。

数字图像由有限数量的元素组成，每个元素都有特殊的位置和数值。这些元素称为画像元素（picture element）、图像元素（image element）和像素（pixel）。

（三）坐标约定

1. 像素坐标（x，y）　　x 表示行，y 表示列

2. 空间坐标（x，y）　　x 表示列，y 表示行



坐标约定：(a) 多数图像处理书籍采用的坐标约定；(b) 图像处理工具箱中采用的坐标约定

（四）类

类	描述
double	8 字节，双精度浮点数，范围$\pm 10^{308}$
single	4 字节，单精度浮点数，范围$\pm 10^{38}$
uint8	1 字节，无符号 8 位整数，范围[0, 255]
uint16	2 字节，无符号 16 位整数，范围[0, 65535]
uint32	4 字节，无符号 32 位整数，范围[0, 10^{31}-1]
in8	1 字节，无符号 8 位整数，范围[-128,127]
int16	2 字节，无符号 16 位整数，范围[-32768,32767]
int32	4 字节，无符号 32 位整数，范围[-214 748 364 8, 214 748 364 7]
char	2 字节，字符
logical	1 字节，值只取 0 或 1

（五）图像类型

1. 灰度图像

当图像元素是 uint8、uint16 类时，其范围分别是[0，255]、[0，6535]的整数值；当图像元素是 double、single 类时，其范围被归一化到[0，1]的浮点数。

2. 二值图像

二值图像是取值为 0、1 的逻辑数组。

3. 索引图象

4. RGB 图像

5. 图像类型转换

图像类型转换	说明
g = im2uint8(f)	g 的范围是[0, 255]
g = im2uint16(f)	g 的范围是[0, 65535]
g = im2double(f)	g 的范围是[0, 1]
g = im2single(f)	g 的范围是[0, 1]
g = mat2gray(f, [min max])	g 的范围是[0, 1], min, max 默认取 f 的最大最小值
g = tofloat(f)	g 的范围是[0, 1]
g = logical(f)	g 的范围是{0, 1}, g 为逻辑数组

注：使用 **islogical(f)**可判断数组 f 是否为逻辑数组；mat2gray 中转换公式如下：

$$g = \frac{f - \min}{\max - \min}$$

三、灰度变换

（一）gamma 变换

gamma 变换：$g = T(r) = r^{gamma}$

1．stretchlim

函数将 f 归一化到[0, 1]，然后返回图像 f 中最低端和最高端处于 10%和 90%位置的灰度值。

```
low_high = stretchlim(f);
```

2．imadjust

返回值 g 与 f 同类型，参数二三默认为[0 1]，参数四默认为 1。此函数将图像 f 中[low_in high_in]的范围映射到[low_out high_out]，即低于 low_in、高于 high_in 分别为 low_out、high_out。

```
g = imadjust(f, [low_in high_in], [low_out high_out], gamma);
```

参数 gamma 指明了由 f 映射生成图像 g 时曲线的形状。如果 gamma 的值小于 1，映射被加权至较高(较亮)的输出值，如图 2-2(a)所示。如果 gamma 的值大于 1，映射加权至较低(较暗)的输出值。如果省略函数参量，gamma 默认为 1(线性映射)。

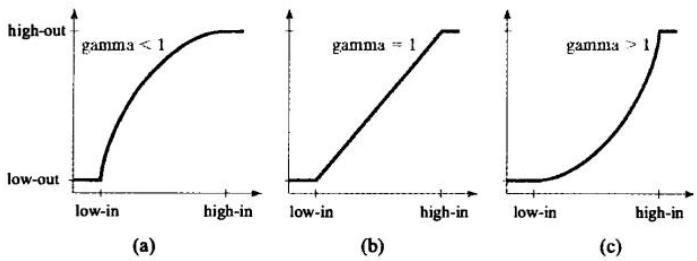
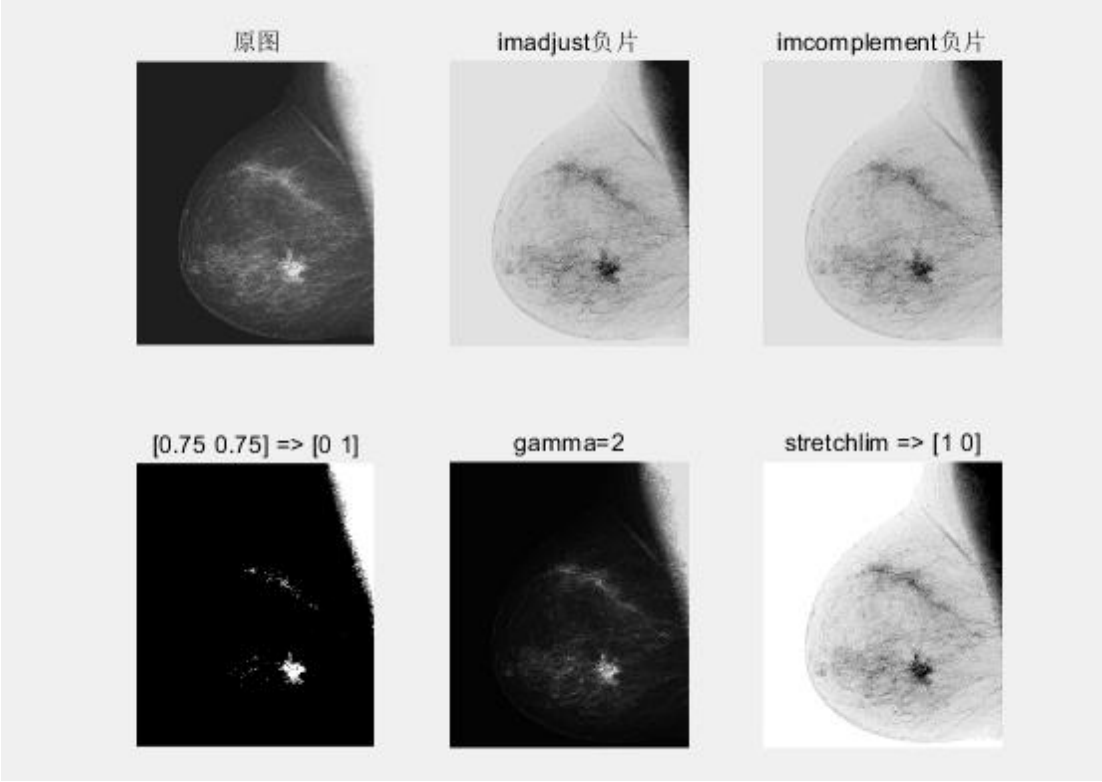


图 2-2　在函数 imadjust 中各种可用的映射

3．例子

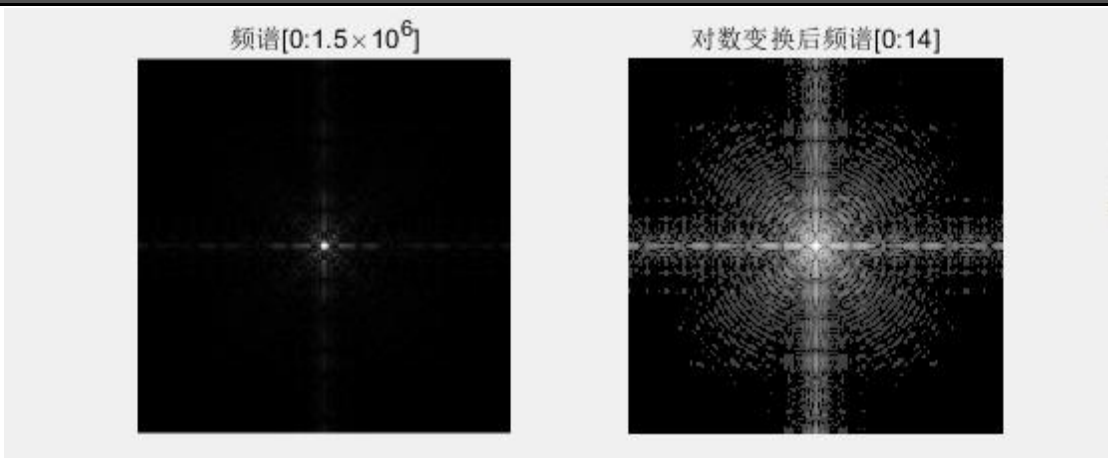
```
% imadjust 灰度变换
f = imread('Fig0203(a).tif');
g1 = imadjust(f, [0 1], [1 0]);      % 负片
g2 = imcompliment(f);                % 负片
g3 = imadjust(f, [0.5 0.75], []);
g4 = imadjust(f, [], [], 2);         % gamma>1 效果会变暗
g5 = imadjust(f, stretchlim(f), [1 0]);
figure,
subplot(231),imshow(f), title('原图');
subplot(232),imshow(g1), title('imadjust 负片');
subplot(233),imshow(g2), title('imcompliment 负片');
subplot(234),imshow(g3), title('[0.75 0.75] => [0 1]');
subplot(235),imshow(g4), title('gamma=2');
subplot(236),imshow(g5), title('stretchlim => [1 0]');
```



（二）对数变换

对数变换：$g = c * \log(1 + f)$，对数变换的作用在于压缩动态范围。

```
f = imread('Fig0205(a).tif');
g = im2uint8(mat2gray(log(1+double(f))));
figure;
subplot(121),imshow(f),title('频谱[0:1.5\times10^6]');
8subplot(122),imshow(g),title('对数变换后频谱[0:14]');
```



（三）对比度拉伸变换

对比度拉伸变换：$s = T(r) = \dfrac{1}{1 + (\dfrac{m}{r})^E}$

s 为输出图像的灰度值[0, 1]范围内，r 表示输入图像的灰度，m 表示输入图像的均值，E 控制函数的斜度。

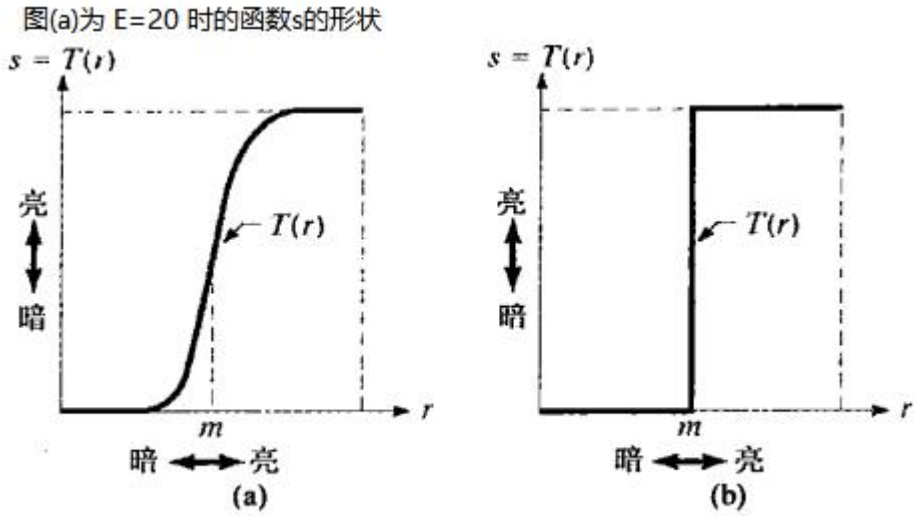


图 2-4 (a) 对比度拉伸变换；(b) 阈值变换

（四）任意灰度变换——线性插值

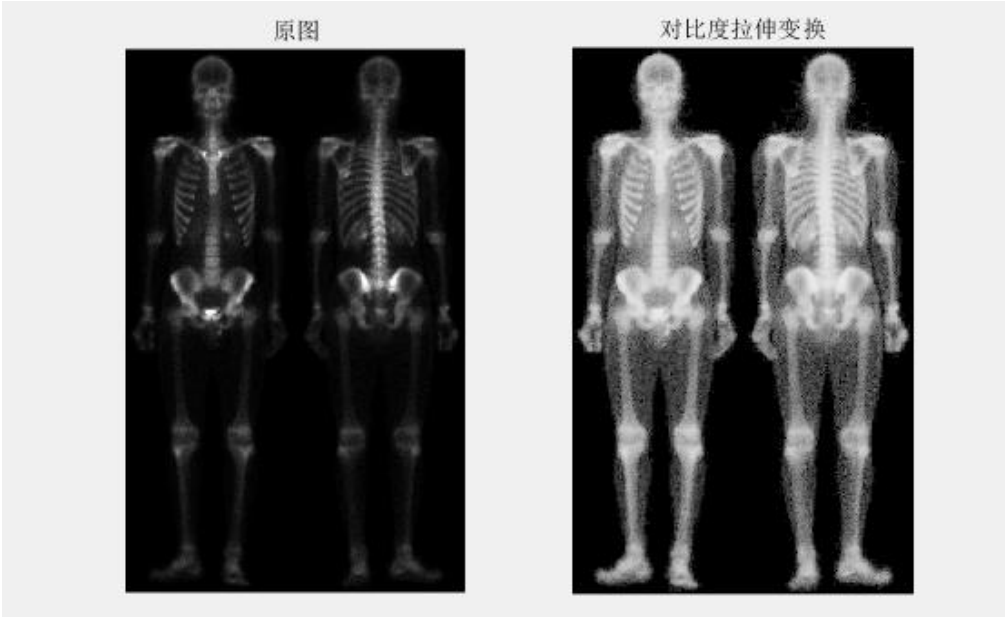
```
内插函数: g = interp1(z, T, f);
```

g、f 为输出、输入图像，都在[0, 1]范围内；已知在 z 处的灰度值为 T，z、T 都为列向量。

（五）intrans 函数

```
g = intrans(f, method, varargin);
g = intrans(f, 'neg');                  // 负片
g = intrans(f, 'log', c, 'uint8|uint16'); // 对数变换
g = intrans(f, 'gamma', gam);           // gamma 变换
g = intrans(f, 'stretch', m, E);        // 对比度拉伸变换
g = intrans(f, 'specified', txfun);        // 任意灰度变换
```

```
f = imread('Fig0206(a).tif');
g = intrans(f, 'stretch', mean2(tofloat(f)), 0.9);
figure;
subplot(121),imshow(f),title('原图');
subplot(122),imshow(g),title('对比度拉伸变换');
```



（五）映射图像范围

```
g = gscale(f, method, low, high); // g 与 f 同类型
g = gscale(f, 'full8');                 // 输出 g 映射到[0, 255]
g = gscale(f, 'full16');              // 输出 g 映射到[0, 65535]
g = gscale(f, 'minmax', low, high);// low 与 high 都在[0, 1]内，相当于 g 映射到[low*Fmin, high*Fmax]
```

四、直方图

（一）直方图

1．定义

对于直方图，在[0, G]范围内定义下面的离散函数，可以理解为各个灰度值出现的次数：

$$h(r_k) = n_k$$

其中，r_k 表示[0, G]内的第 k 个灰度，n_k 表示这种灰度的像素数。对于 uint8，G 的值是 255；对于 uint16，G 的值是 65535；对于浮点型，G 的值是 1.0。

对于归一化直方图，可以理解为各个灰度值出现的概率：

$$\mathrm{p}(r_k) = \frac{h(n_k)}{n} = \frac{n_k}{n}$$

其中，n 表示图像中的像素总数。

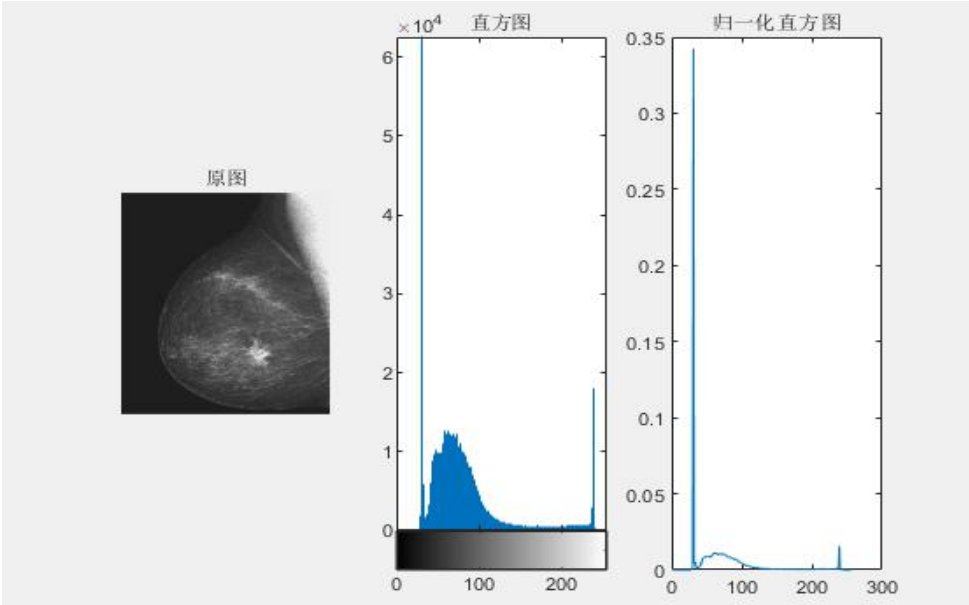
2．绘制

```
h = imhist(f, b);          // 直方图
p = imhist(f, b) / numel(f);   // 归一化直方图
```

其中，b 表示直方图中灰度的个数，即上方的 G，b 默认值为 256。

3．例子

```
f = imread('Fig0203(a).tif');
p = imhist(f)/numel(f);
figure;
subplot(131),imshow(f),title('原图');
subplot(132),imhist(f);title('直方图');
subplot(133),plot(p),title('归一化直方图');
```



（二）其他图形

1．条形图

```
bar(horz, z, width);
```

其中，z 为包含被绘制点的行向量；horz 为水平标度之的向量；width 表示介于 0、1 的条形图宽度的浮点数，默认值为 0.8。

2．杆状图

```
stem(horz, z, 'LinSpec', 'fill');
```

其中，z 为包含被绘制点的行向量；horz 为水平标度之的向量；LinSpec 为设置线条颜色形状；fill 表示填充形状。

3．折线图

```
plot(horz, z, 'LinSpec');
```

其中，z 为包含被绘制点的行向量；horz 为水平标度之的向量；LinSpec 为设置线条颜色形状。

4．函数画图

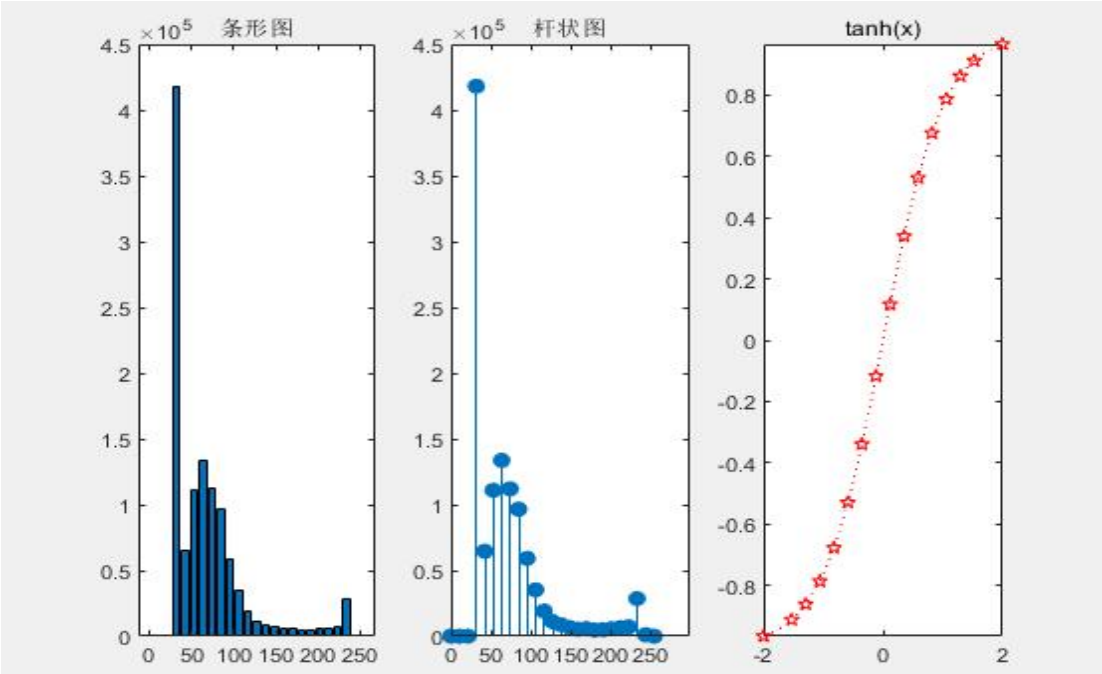
```
fplot(fhandle, limits, 'LinSpec', n);
```

其中，fhandle 表示函数句柄；limits 为[xmin xmax]的向量；LinSpec 为设置线条颜色形状；n 设置函数绘制最小点数。

颜色规定		线条规定		标记点规定	
符合	颜色	符合	线型	符合	标记点
k	黑	-	实线	+	加号
w	白	--	虚线	o	圆
r	红	:	点线	*	星号
g	绿	-.	点划线	.	点
b	蓝			x	叉
c	青			s	方块
y	黄			q	菱形
m	洋红			^	向上三角形
				v	向下三角形
				<	向左三角形
				>	向右三角形
				p	五角星
				h	六角星

5. 例子

```
f = imread('Fig0203(a).tif');
h = imhist(f, 25);
horz = linspace(0, 255, 25);
fhandle = @tanh;

figure;
subplot(131),bar(horz, h),title('条形图');
set(gca, 'xtick', 0:50:255);
subplot(132),stem(horz, h, 'fill'),title('杆状图');
set(gca, 'xtick', 0:50:255);
subplot(133),fplot(fhandle, [-2, 2], 'r:p'), title('tanh(x)');
```



（三）直方图均衡

直方图均衡可以增强局部对比度而不影响全局对比度，常用于背景或前景太亮或太暗的图像。本质是归一化直方图的累计分布函数（CDF）：

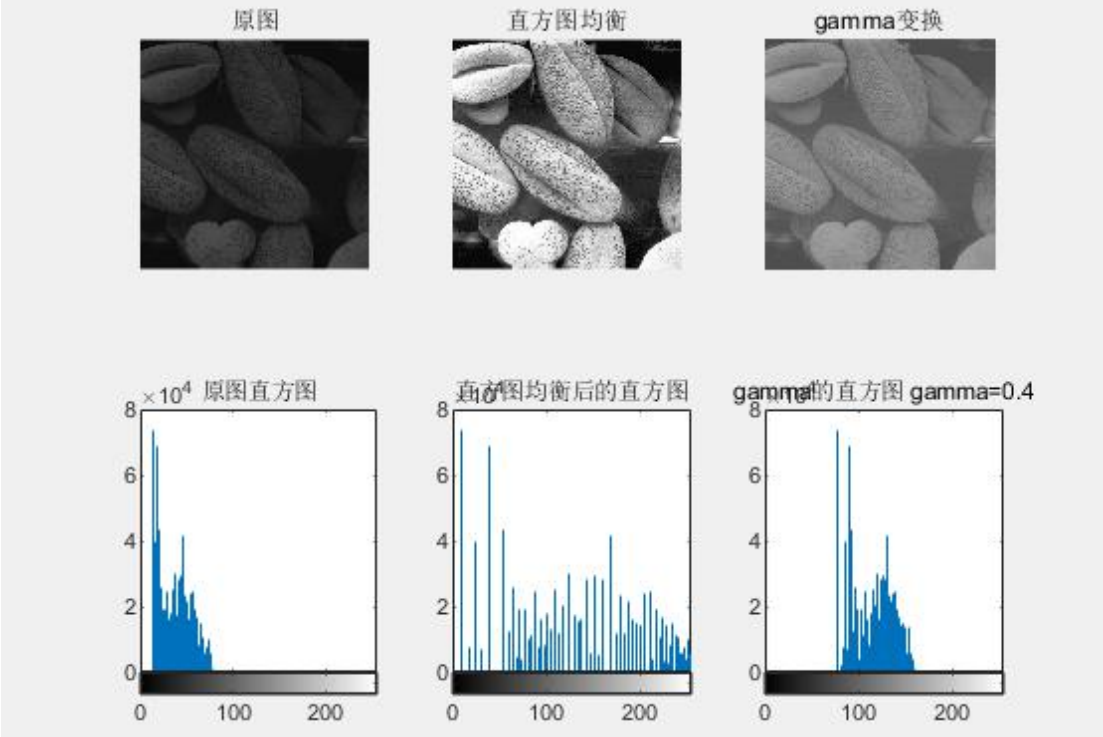
$$s = T(r) = \int_0^r p_r(w)dw$$

```
g = histeq(f, nlev);
```

其中，f 为输入图像；nlev 为输出图像的灰度级，默认值为 64。

```
f = imread('Fig0208(a).tif');
g = histeq(f, 256);              % 直方图均衡
gam = intrans(f, 'gamma', 0.4);

figure;
subplot(231),imshow(f),title('原图');
subplot(234),imhist(f);title('原图直方图'),ylim('auto');
```

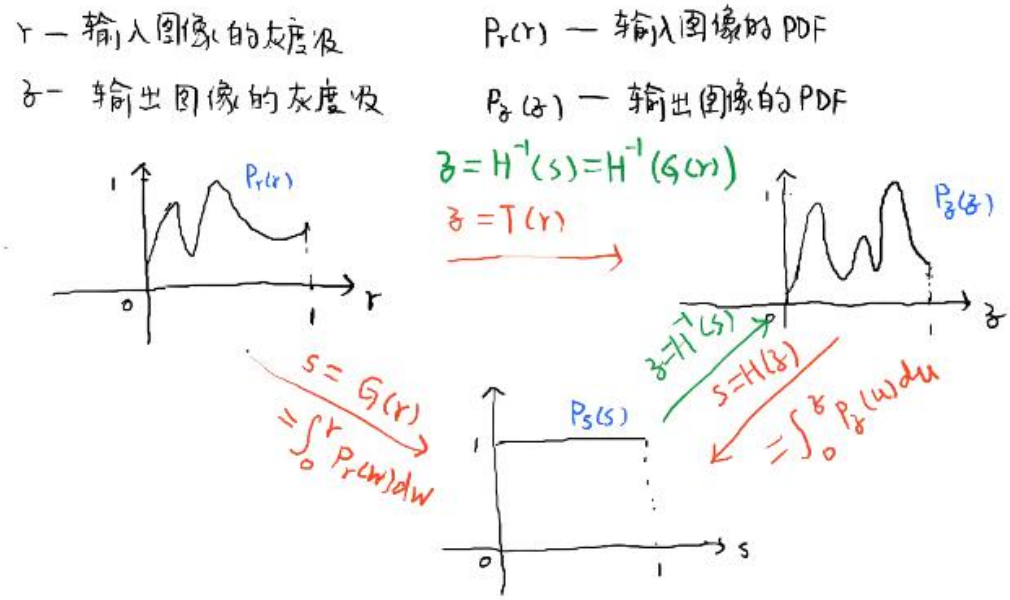
```
subplot(232),imshow(g),title('直方图均衡');
subplot(235),imhist(g);title("直方图均衡后的直方图"),ylim('auto');
subplot(233),imshow(gam),title('gamma 变换');
subplot(236),imhist(gam);title("gamma 的直方图 gamma=0.4"), ylim('auto');
```



（四）直方图匹配（规定化）

1. 原理

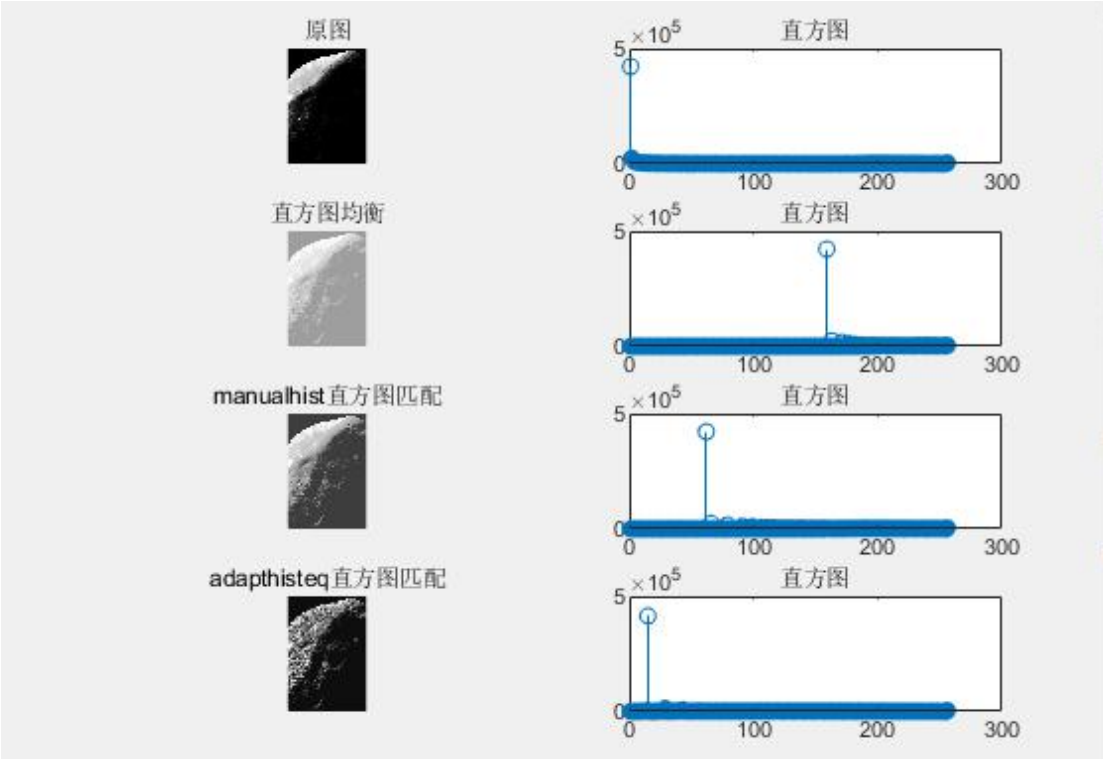
由 r 归一化直方图到 s，z 也可归一化直方图到 s，则 r 到 z 可通过 r 归一化直方图在 s 到 z 的归一化直方图反函数得到，如下图：



2. 绘图函数

冈萨雷斯自定义函数：manulhist
工具箱函数：adapthisteq

```
f = imread('Fig0210(a).tif');
h = histeq(f, 256);      % 直方图均衡
p = manualhist;
g1 = histeq(f, p);       % 自定义函数直方图匹配
% 工具箱函数直方图匹配
g2 = adapthisteq(f, 'NumTiles', [25 25], 'ClipLimit', 0.05);

figure;
subplot(421),imshow(f),title('原图');
subplot(422),stem(imhist(f));title('直方图'),ylim('auto');
subplot(423),imshow(h),title('直方图均衡');
subplot(424),stem(imhist(h));title('直方图'),ylim('auto');
subplot(425),imshow(g1),title('manualhist 直方图匹配');
subplot(426),stem(imhist(g1));title('直方图'),ylim('auto');
subplot(427),imshow(g2),title('adapthisteq 直方图匹配');
subplot(428),stem(imhist(g2));title('直方图'),ylim('auto');
```



五、空间滤波(空间卷积)

（一）线性空间滤波

1．滤波器

也叫模板(mask|template)、滤波模板、核、卷积模板、卷积核等，指的是对邻域内每个像素所乘的系数组成的矩阵。

2．滤波器大小

若邻域大小为 m*n，则需要 mn 个系数，滤波器大小一般取奇数，即 m=2*a+1, n=2*b+1。

3．操作分类：相关、卷积。

卷积与相关的唯一区别在于，执行卷积操作时，将滤波器 w 旋转 180°。

4．一维相关、卷积

图 2-14　一维相关和卷积的操作说明

5．二维相关、卷积

相关操作：$\omega(x,y)\star f(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)f(x+s,y+t)$

卷积操作：$\omega(x,y)\star f(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)f(x-s,y-t)$

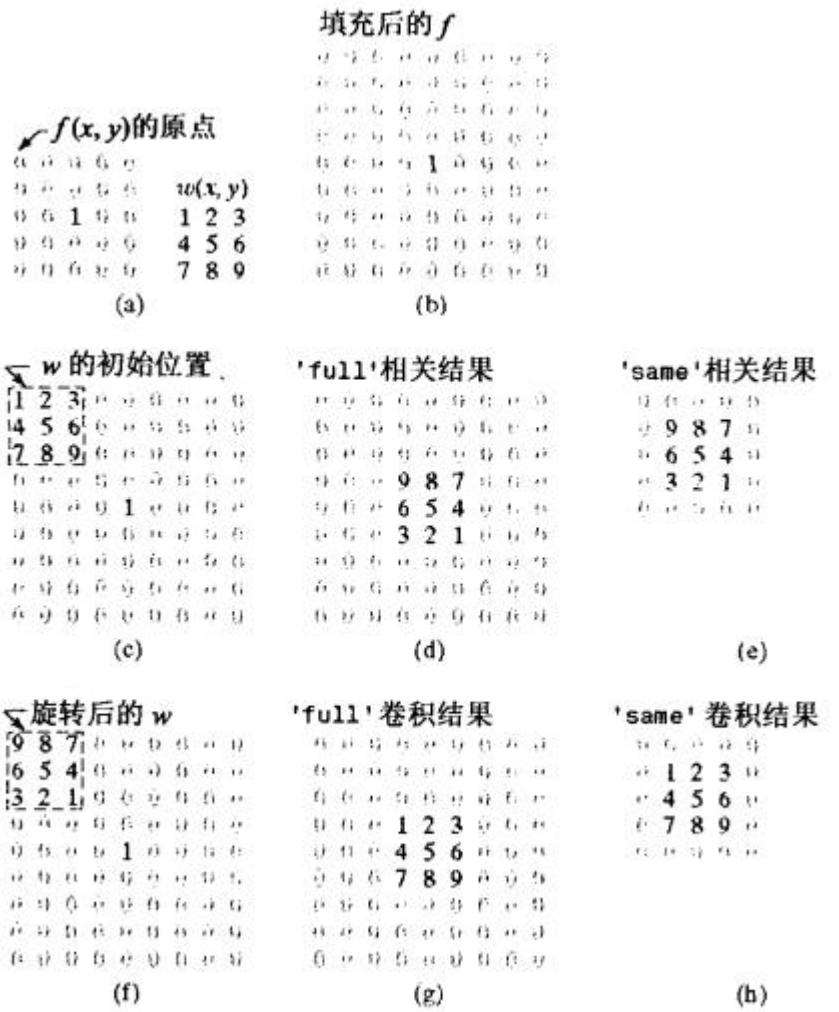


图 2-15　二维相关和卷积的操作示例。为便于查看，0 显示为灰色

6．MATLAB 函数

```
g = imfilter(f, w, filtering_mode, boundary_options, size_options);
```

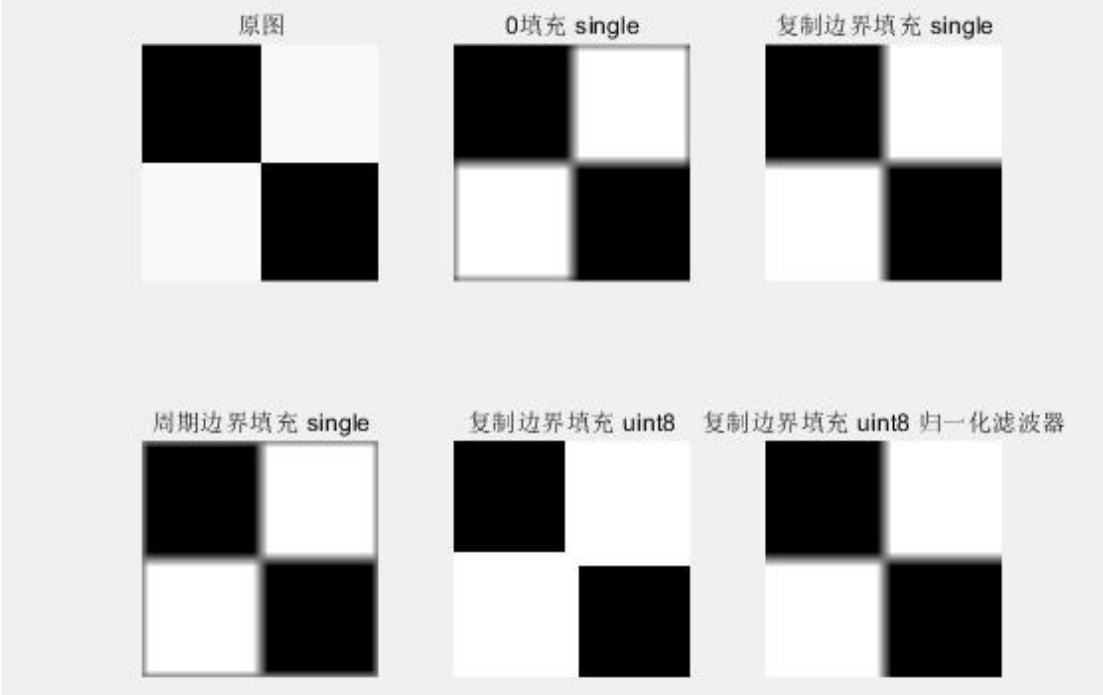
执行前，f 最好转换为浮点型。

其中，f 为输入图像；输出图像 g 与 f 同类型；w 为滤波模板；filtering_mode 为滤波模式，分为'corr'(默认)、'conv'；boundary_options 为边界填充选项，分为 p(一个值,默认为 0)、'replicate'(复制边界)、'symmetric'(自身边界镜像扩展)、'circular'(周期循环)；size_options 为输出图像大小，分为'same'(默认)、'full'。

7．例子

```
f = imread('Fig0216(a).tif');
f = im2single(f);
w = ones(31);
g1 = imfilter(f, w);                % 0 填充
g2 = imfilter(f, w, 'replicate');   % 复制边界填充
g3 = imfilter(f, w, 'circular');    % 周期边界填充
f = im2uint8(f);
g4 = imfilter(f, w, 'replicate');   % 不转浮点型出现截断
w = 1 / 31^2 .* w;                  % 滤波器归一化
g5 = imfilter(f, w, 'replicate');   % 归一化滤波器不会截断
figure;
subplot(231),imshow(f),title('原图');
```

```
subplot(232),imshow(g1, []),title('0 填充 single');
subplot(233),imshow(g2, []),title('复制边界填充 single');
subplot(234),imshow(g3, []),title('周期边界填充 single');
subplot(235),imshow(g4, []),title('复制边界填充 uint8');
subplot(236),imshow(g5, []),title('复制边界填充 uint8 归一化滤波器');
```



（二）非线性空间滤波

1．填充图像

```
fp = padarray(f, [r c], method, direction);
```

其中，f 为输入图像；[r c]为填充 f 的行数和列数；method 为填充方法，分为'symmetric'(边界镜像填充)、'replicate'(复制边界填充)、'circular'(周期填充)；direction 为移动方向，分为'pre'(在每一维的第一个元素前填充)、'post'(在每一维的最后一个元素后填充)、'both'(包含前面两种，为默认值)。

2．非线性空间滤波函数

colfilt 函数会生成一个 mn*MN 的矩阵，所以比较占内存，但相对快速，所以比较常用。

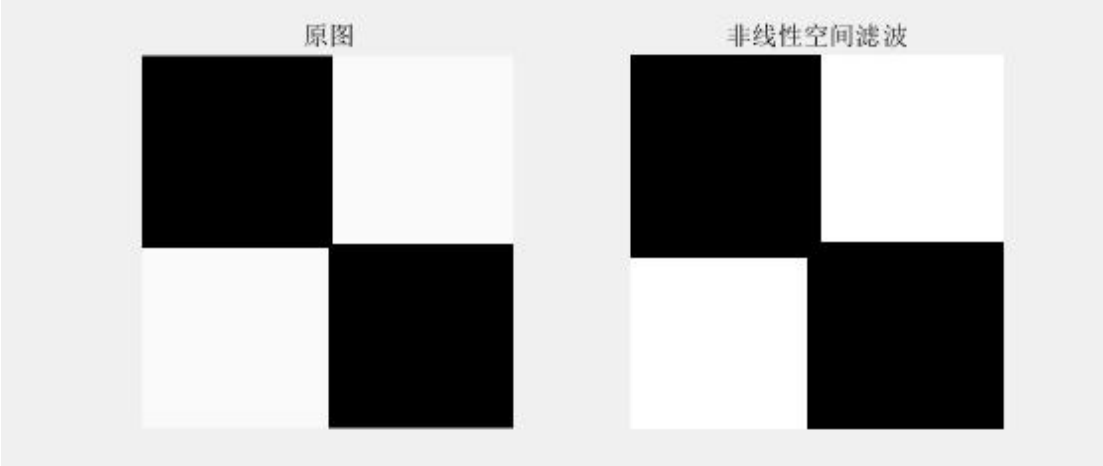
```
g = colfilt(f, [m n], 'sliding', fun);
```

执行前，f 一般先进行边界填充。

其中，f 为输入图像；[m n]为滤波器大小；'sliding'表示滤波器在 f 中逐像素滑动；fun 为函数句柄。

3．例子

```
f = imread('Fig0216(a).tif');
fp = padarray(f, [3 3], 'replicate');    % 边界填充
gmean = @(A) prod(A, 1).^(1/size(A, 1));% 求几何平均
g = colfilt(fp, [3 3], 'sliding', gmean);%非线性空间滤波
[M, N] = size(f);
g = g((1:M)+3, (1:N)+3);                 % 去掉填充部分
figure;
subplot(121),imshow(f),title('原图');
subplot(122),imshow(g),title('非线性空间滤波');
```



（三）线性空间滤波器

```
w = fspecial('type', parameters);
```

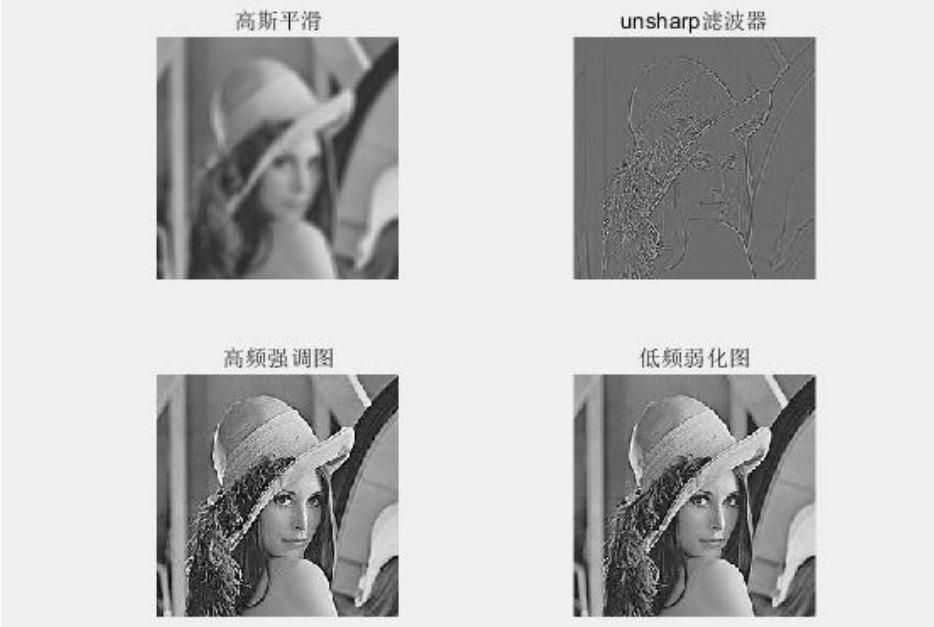
其中，type 为滤波器类型；parameters 进一步定义指定的滤波器。

图像类型转换	说明
fspecial('average', [r c])	矩阵平均滤波器，[r c]为滤波器大小{[3 3]}
fspecial('disk', r)	圆形平均滤波器，r 为半径{5}
fspecial('gaussian', [r c], sig)	高斯低通滤波器，[r c]为滤波器大小{[3 3]}；sig 为偏移量{0.5}
fspecial('laplacian', alpha)	3*3 的拉普拉斯滤波器，alpha 为[0, 1]的数{0.2}
fspecial('log', [r c], sig)	高斯-拉普拉斯滤波器；[r c]为滤波器大小{[5 5]}；sig 为偏移量{0.5}
fspecial('motion', len, theta)	len 为滤波器大小，像素为单位{9}；theta 为运动方向，度数为单位{0}
fspecial('prewitt')	3*3 的 Prewitt 滤波器
special('sobel')	3*3 的 Sobel 滤波器
fspecial('unsharp', alpha)	3*3 的非尖锐滤波器，alpha 为[0, 1]的控制形状的数{0.2}

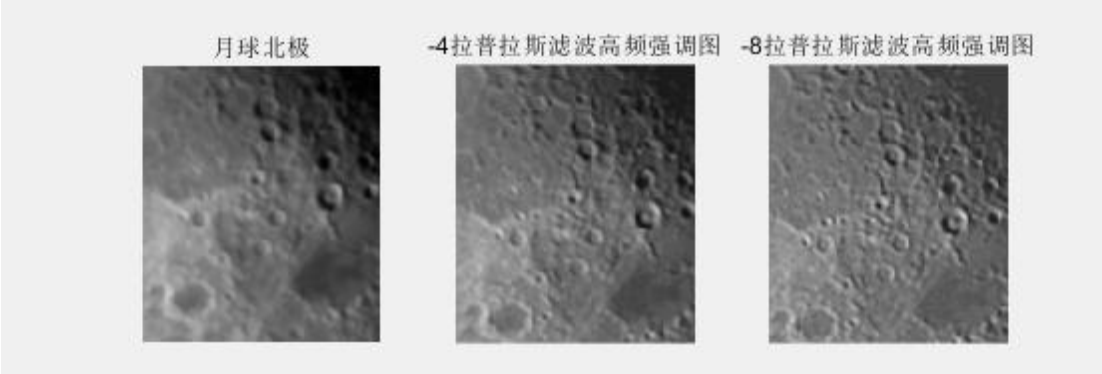
```
% 文件 LaplacianMask.m
function f = LaplacianMask(n)
    % 生成任意奇数大小的拉普拉斯模板
    f = ones(n);
    f((n+1)/2, (n+1)/2) = (-1)*n*n + 1;
end
```

```
f = imread('Lena.jpg');
f = im2single(rgb2gray(f));          % 转为浮点型
w1 = fspecial('average', 3);
w2 = fspecial('average', 5);
w3 = fspecial('average', 7);
g1 = imfilter(f, w1, 'replicate');
g2 = imfilter(f, w2, 'replicate');
g3 = imfilter(f, w3, 'replicate');

w4 = fspecial('gaussian', [3 5], 0.5);% 高斯滤波器
```

```
g4 = imfilter(f, w4, 'replicate');     % 高斯平滑滤波后
fd = f - g4;          % unsharp 滤波器
g5 = f + 1.5*fd;      % 高频强调图  详细化
g6 = f + 0.7*fd;      % 低频弱化图  模糊化
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(g1),title('均值平滑 3*3 卷积核');
subplot(223),imshow(g2),title('均值平滑 5*5 卷积核');
subplot(224),imshow(g3),title('均值平滑 7*7 卷积核');
figure;
subplot(221),imshow(g3),title('高斯平滑');
subplot(222),imshow(fd, []),title('unsharp 滤波器');
subplot(223),imshow(g5),title('高频强调图');
subplot(224),imshow(g6),title('低频弱化图');
```



```
f = imread('Fig0217(a).tif');
f = tofloat(f);
w1 = fspecial('laplacian', 0);
w2 = [1 1 1; 1 -8 1; 1 1 1];
g1 = imfilter(f, w1, 'replicate');  % 拉普拉斯滤波后
g3 = f - g1;                        % 高频强调图
g2 = imfilter(f, w2, 'replicate');  % 拉普拉斯滤波后
g4 = f - g2;                        % 高频强调图
figure;
subplot(131),imshow(f),title('月球北极');
subplot(132),imshow(g3, []),title('-4 拉普拉斯滤波高频强调图')
subplot(133),imshow(g4, []),title('-8 拉普拉斯滤波高频强调图')
```



（四）非线性空间滤波器

```
g = ordfilt2(f, order, domain);
```

函数表示邻域排序集合中的第 order 个元素去代替 f 中的每个元素，然后得到图像 g。domain 可当作逻辑模板，对应值为 1 的参与计算，为 0 则不参与。

最小滤波器：g = ordfilt2(f, 1, ones(m, n));

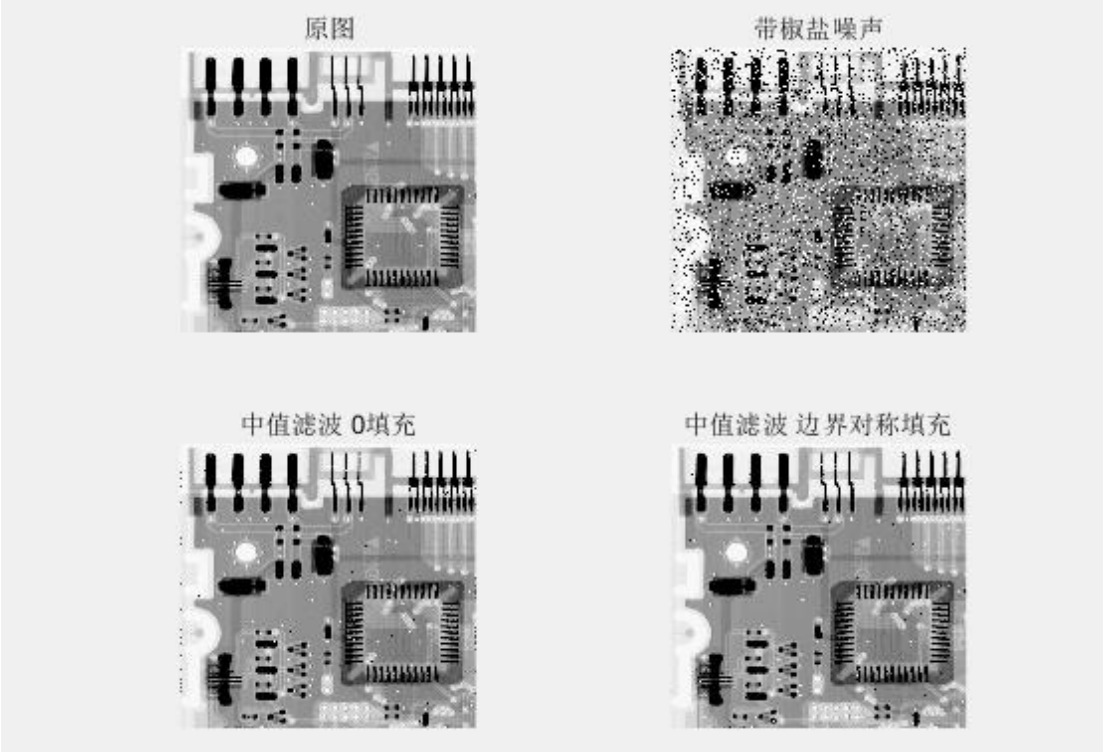
最大滤波器：g = ordfilt2(f, m*n, ones(m, n));

中值滤波器：g = ordfilt2(f, (m*n+1)/2, ones(m, n));

```
g = medfilt2(f, [m, n], padopt);
```

函数为中值滤波器，其中，[m, n]为滤波器大小；padopt 为边界填充方式，分为'zeros'(默认)、'symmetric'(边界复制)、'indexed'(f 为 double 类时填充 1，否则填充 0)。

```
f = imread('Fig0219(a).tif');
fn = imnoise(f, 'salt & pepper', 0.2);  % 添加椒盐噪声
g1 = medfilt2(fn);                      % 中值滤波 0 填充
g2 = medfilt2(fn, 'symmetric');         % 中值滤波 边界对称填充

figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(fn),title('带椒盐噪声');
subplot(223),imshow(g1),title('中值滤波 0 填充');
subplot(224),imshow(g2),title('中值滤波 边界对称填充');
```



六、模糊技术

七、频域处理

（一）傅里叶变换

1．二维离散傅里叶变换（DFT）

$$\mathrm{F}(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-j(\frac{2\pi u}{M}x+\frac{2\pi v}{N}y)}$$

其中，u=0, 1,...,M-1；v=0, 1,...,N-1；j 为虚数单位；f(x, y)为 M*N 像素的数字图像；所得到的 F(x,y)的频率中心在左上角处。

2．二维离散傅里叶反变换（IDFT）

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j(\frac{2\pi x}{M}u+\frac{2\pi y}{N}v)}$$

其中，x=0, 1,...,M-1；y=0, 1,...,N-1；j 为虚数单位；F(x, y)为数字图像的傅里叶变换；所得到的 f(x, y)属于复数域。

3．傅里叶谱

频谱，也就是 F(u, v)的幅度，它是一个实函数，令 R(u, v)、I(u, v)表示 F(u, v)的实部和虚部。

$$\mathrm{F}(u,v)=R(u,v)+j*I(u,v)$$

$$|F(u,v)|=[R^2(u,v)+I^2(u,v)]^{\frac{1}{2}}$$

4．相角

$$\phi(u,v)=\arctan[\frac{I(u,v)}{R(u,v)}]$$

5．功率谱

功率谱为幅度的平方，表示如下：

$$P(u,v)=|F(u,v)|^2=R^2(u,v)+I^2(u,v)$$

6．周期性

由 DFT 和 IDFT 得到的图像都是无穷周期的。

$$F(u+k_1M,v+k_2N)=F(u,v)$$
$$f(u+k_1M,v+k_2N)=f(u,v)$$

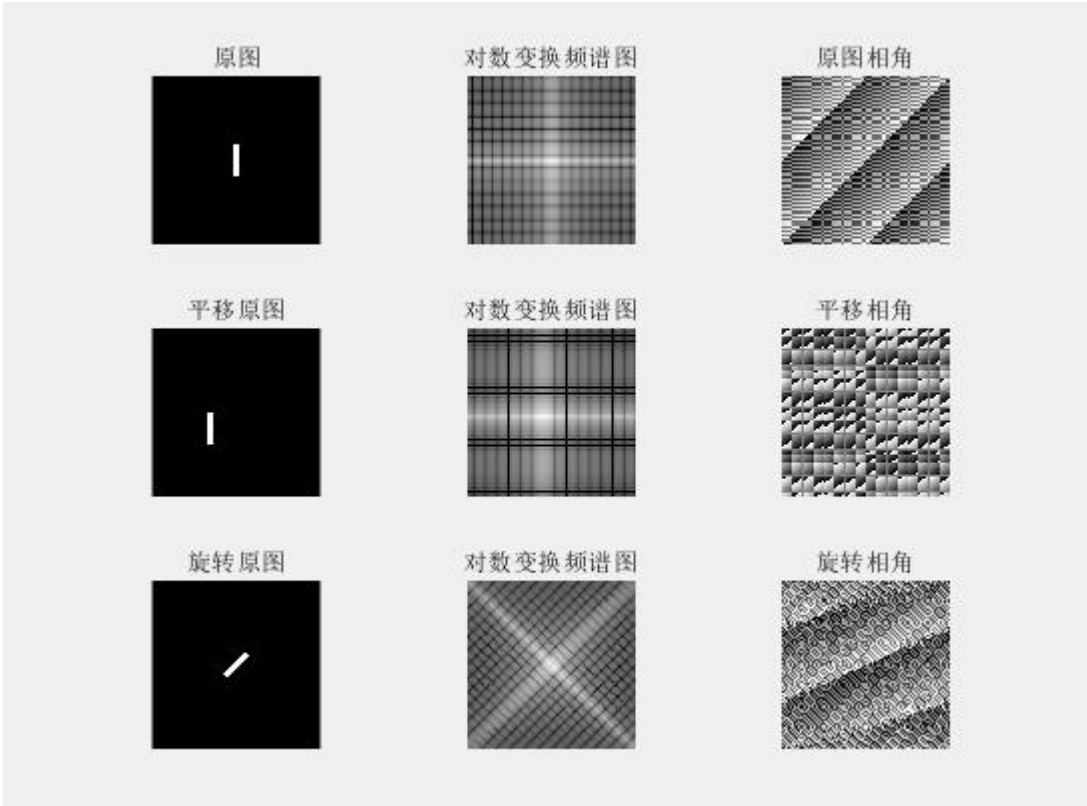
7．MATLAB 函数

```
F = fft2(f[, P, Q]);   % 计算 f 的 DFT 变换；填充 0 直至大小为 P x Q
f = real(ifft2(F));% 计算 F 的 IDFT 变换，使用 real()函数选取实数部分
Fc = fftshift(F);      % 将频率中心移到中心位置，坐标为[floor(M/2)+1, floor(N/2)+1]
F = ifftshift(Fc); % 将频率中心移到左上角处
S1 = abs(Fc);          % 计算 F 的频谱，使用 imshow(S1, [])输出图像
S2 = abs(1+log(Fc));   % 计算 F 的对数频谱
phi = atan2(imag(F), real(F)); % 计算 F 的相角
phi = angle(F);                % 计算 F 的相角
F = S.*exp(i*phi); % 使用频谱和相角计算 DFT，依据 F = |F| e^{jφ}
```

8．例子

```
% ==============================原图
f = imread('Fig0303(a).tif');
% f = im2single(f);
F = fft2(f);            % DFT
f_ = ifft2(F);          % IDFT
f_ = real(f_);          % 除去误差
Fc = fftshift(F);       % 移动频率中心
SF = abs(Fc);           % 原图频谱图
SF2 = log(1 + abs(Fc)); % 对数变换频谱图
fphi = angle(Fc);       % 相角
fphi = atan2(imag(F), real(F));
[M, N] = size(f);
fprintf("频率中心：(%d, %d)\n", floor(M/2)+1, floor(N/2)+1);
% ==============================平移
g = zeros(512);
g(209+50:305+50, 249-80:265-80) = f(209:305, 249:265);
% g = im2single(g);
G = fft2(g);            % DFT
Gc = fftshift(G);       % 移动频率中心
SG = abs(Gc);           % 原图频谱图
SG2 = log(1 + abs(Gc)); % 对数变换频谱图
gphi = angle(Gc);       % 相角
% ==============================旋转
```

```
h = imrotate(f, -45, 'bilinear', 'crop');
H = fft2(h);              % DFT
Hc = fftshift(H);         % 移动频率中心
SH = abs(Hc);             % 原图频谱图
SH2 = log(1 + abs(Hc)); % 对数变换频谱图
hphi = angle(Hc);         % 相角
% ============================画图
figure;
subplot(331), imshow(f), title('原图');
subplot(332), imshow(SF2, []), title('对数变换频谱图');
subplot(333), imshow(fphi, []), title('原图相角');
subplot(334), imshow(g), title('平移原图');
subplot(335), imshow(SG2, []), title('对数变换频谱图');
subplot(336), imshow(gphi, []), title('平移相角');
subplot(337), imshow(h), title('旋转原图');
subplot(338), imshow(SH2, []), title('对数变换频谱图');
subplot(339), imshow(hphi, []), title('旋转相角');
```



（二）频域滤波

1．0 填充图像

（1）原因

空间域和频率域的线性滤波的基础都是卷积定理，如下所示，等号左侧为空间域滤波，右侧为频率域滤波。

$$DFT[f(x,y)\bigstar h(x,y)] = DFT[f(x,y)].*DFT[h(x,y)]$$

我们知道，当工作在 DFT 时，图像及其变换都是周期性的。不难发现，如果关于函数非零部分的延续周期很靠近，卷积周期函数会引起相邻周期的重叠，这种影响称为折叠误差。避免这种误差可通过对图像进行填充 0 来解决。

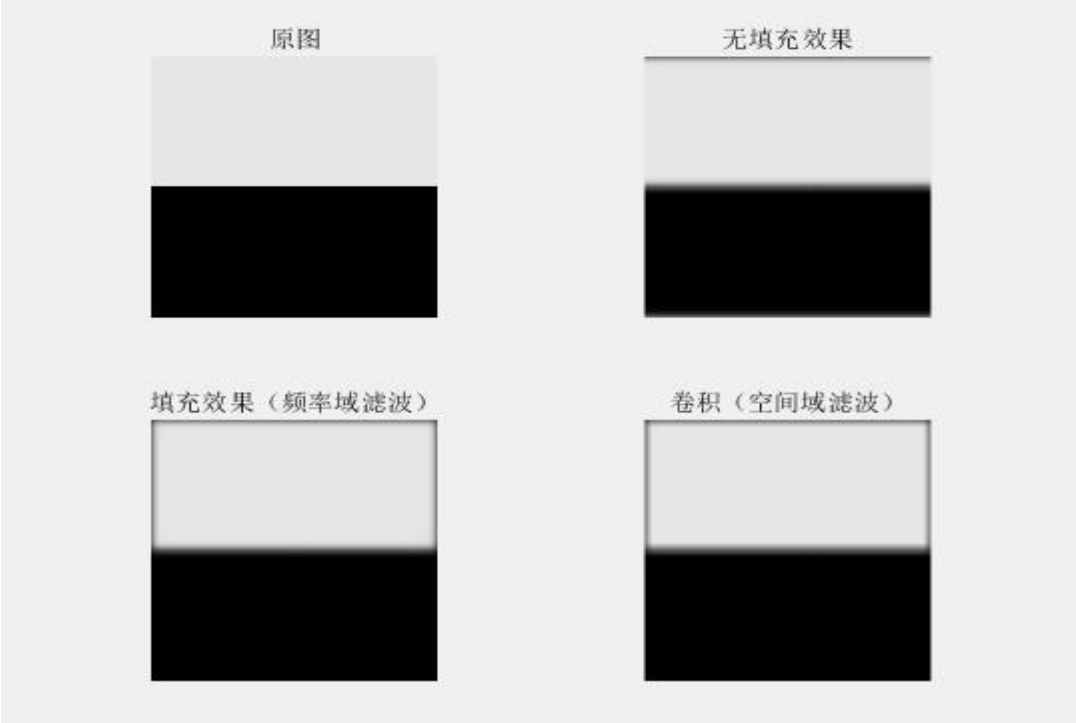
（2）方法

使用 paddedsize 函数，默认返回值为最小偶数值；添加'PWR2'参数返回值为最小 2 次幂值。

```
PQ = paddedsize([A, B], 'PWR2');          // PQ [>=A+B-1]
PQ = paddedsize([A, B], [C, D], 'PWR2');  // PQ [ >=A+C-1, >=B+D-1]
```

（3）例子

```
f = imread('Fig0305(a).tif');        % 读取图像
figure,
subplot(221),imshow(f),title('原图');

% 无填充效果
f = imread('Fig0305(a).tif');        % 读取图像
[M, N] = size(f);                    % 图像大小
[f, revertclass] = tofloat(f);       % 转为 float 类型
F = fft2(f);                         % DFT
H = lpfilter('gaussian', M, N, 10); % 脉冲响应的傅里叶变换(传递函数) 10 表示标准差
G = H.*F;                            % 频率域滤波
g = ifft2(G);                        % DFT 反变换
gc = revertclass(g);                 % 转回 unit8 类型
subplot(222),imshow(gc),title('无填充效果');

% 填充效果 点乘方法
f = imread('Fig0305(a).tif');        % 读取图像
[f, revertclass] = tofloat(f);       % 转为 float 类型
PQ = paddedsize(size(f));            % 计算填充大小
F = fft2(f, PQ(1), PQ(2));           % DFT
H = lpfilter('gaussian', PQ(1), PQ(2), 2*10);
G = H.*F;                            % 频率域滤波
g = ifft2(G);                        % DFT 反变换
gc = g(1:size(f,1), 1:size(f,2));    % 截取大小
gc = revertclass(gc);                % 转回 unit8 类型
subplot(223),imshow(gc),title('填充效果（频率域滤波）');

% 卷积方法(与填充效果一致)
f = imread('Fig0305(a).tif');        % 读取图像
[f, revertclass] = tofloat(f);       % 转为 float 类型
h = fspecial('gaussian', 15, 7);     % 生成高斯模板
g = imfilter(f, h);                  % 滤波器
g = revertclass(g);                  % 转回 unit8 类型
subplot(224), imshow(g), title('卷积（空间域滤波）');
```



2．DFT 滤波步骤

（1）将图像转化为浮点型

```
[f, revertclass] = tofloat(f);
```

（2）获得填充参数

```
PQ = paddedsize(size(f)););
```

（3）DFT 变换

```
F = fft2(f, PQ(1), PQ(2));
```

（4）生成非居中格式的滤波函数 H

```
H = [l|h]pfilter(type, PQ(1), PQ(2), D0, n);
```

（5）频率域滤波

```
G = H.*F;
```

（6）IDFT 变换

```
g = ifft2(G);
```

（7）获得原始大小图像

```
gc = g(1:size(f,1), 1:size(f,2));
```

（8）转回原型

```
gc = revertclass(gc);
```

3．频域滤波函数

函数 dftfilt 集合上述步骤于一身，可简化代码。

```
g = dftfilt(f, H, classout);
```

其中，classout 为输出图像类型，'original'代表保持原类型，'filtpoint'代表输出图像为浮点型。

（三）从空间域滤波器获得频率域滤波器

1．空间域滤波器转为频率域滤波器

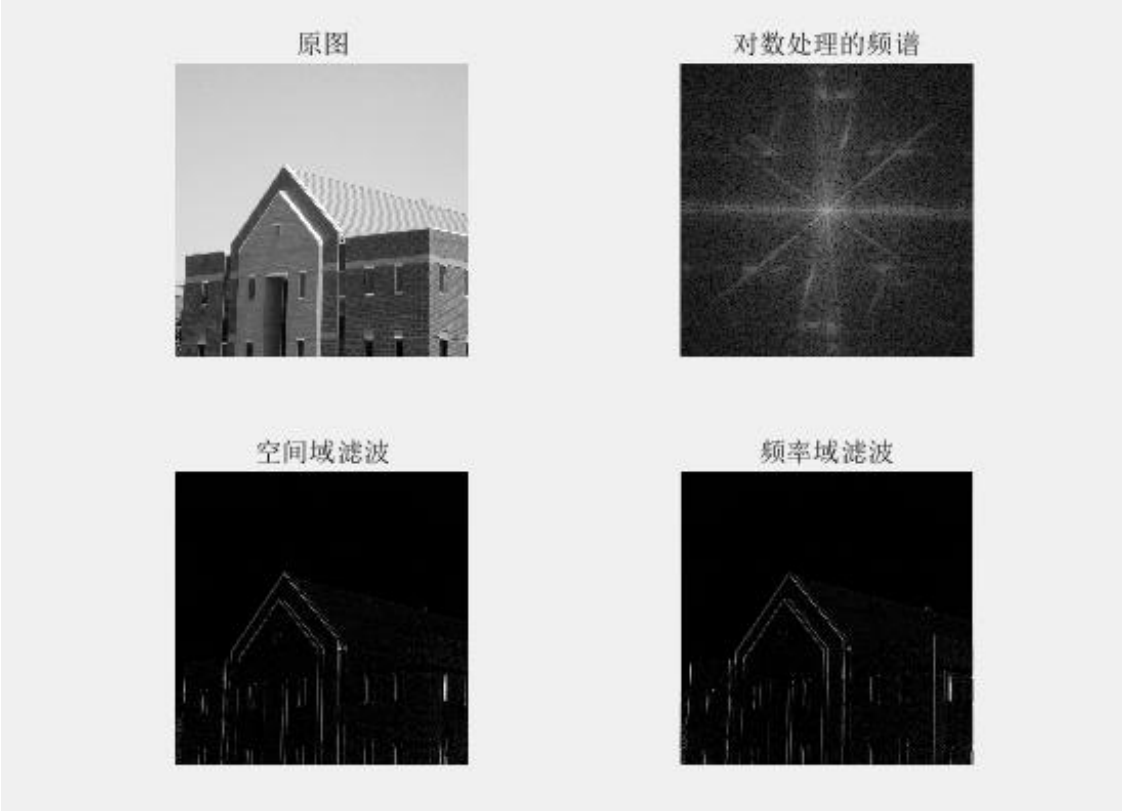
```
H = freqz2(h, R, C);
```

其中，h 为二维空间域滤波器，输出 H 为相应的频率域滤波器且 H 的频率中心在中间，R、C 为输出 H 的行、列。

freqz2(h, R, C)　　% 缺省输出值时显示为三位透视图

2．例子

```
f = imread('Fig0309(a).tif');
f = tofloat(f);
F = fft2(f);                        % DFT
Fc = fftshift(log(1+abs(F)));       % 频率中心转到左上角
PQ = paddedsize(size(f));           % 计算需要填充的数值
h = fspecial('sobel')';             % 已旋转 180 度
H = freqz2(h, PQ(1), PQ(2));        % DFT
H = ifftshift(H);                   % 频率中心移到左上角
gs = imfilter(f, h);                % 空间域滤波
gf = dftfilt(f, H);                 % 频率域滤波

figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(Fc, []),title('对数处理的频谱');
subplot(223),imshow(abs(gs), []),title('空间域滤波');
subplot(224),imshow(abs(gf), []),title('频率域滤波');
```



（四）直接生成滤波器

1．建立网格数组

```
[U, V] = dftuv(M, N);
```

返回网格坐标，经平方和计算后可得到各网格点到中心的距离，其中，M、N 为网格大小的行数、列数。

2．计算平方和

```
D = hypot(U, V);
```

返回 U.^2+V.^2 的值。

```
[U, V] = dftuv(8, 5);
D = hypot(U, V);
Dc = fftshift(D)
```

```
Dc =

  8×5 single 矩阵

   4.4721   4.1231   4.0000   4.1231   4.4721
   3.6056   3.1623   3.0000   3.1623   3.6056
   2.8284   2.2361   2.0000   2.2361   2.8284
   2.2361   1.4142   1.0000   1.4142   2.2361
   2.0000   1.0000        0   1.0000   2.0000
   2.2361   1.4142   1.0000   1.4142   2.2361
   2.8284   2.2361   2.0000   2.2361   2.8284
   3.6056   3.1623   3.0000   3.1623   3.6056
```

3．低通(平滑)频域滤波器

（1）理想滤波器（ILPF）

理想低通滤波器切断的是圆外的分量，保留圆上、圆内的点不变。其传递函数如下：

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

其中，D_0 为正数，D(u,v)为点(u,v)到滤波器中心的距离。

（2）巴特沃斯滤波器(BLPF)

巴特沃斯滤波器与理想低通滤波器相同，截断圆外的分量；但不同的是，巴特沃斯版的传递函数比较连续。其传递函数如下：

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

其中，D_0 为正数，n 为阶数，D(u,v)为点(u,v)到滤波器中心的距离。

（3）高斯滤波器（GLPF）

$$H(u,v)=e^{-\frac{D^2(u,v)}{2\sigma^2}}$$

其中，σ 为标准差，代表 D_0；D(u,v)为点(u,v)到滤波器中心的距离。

（4）自定义函数

自定义函数 lpfilter 函数集成了定义以上三种滤波器，具体使用如下：

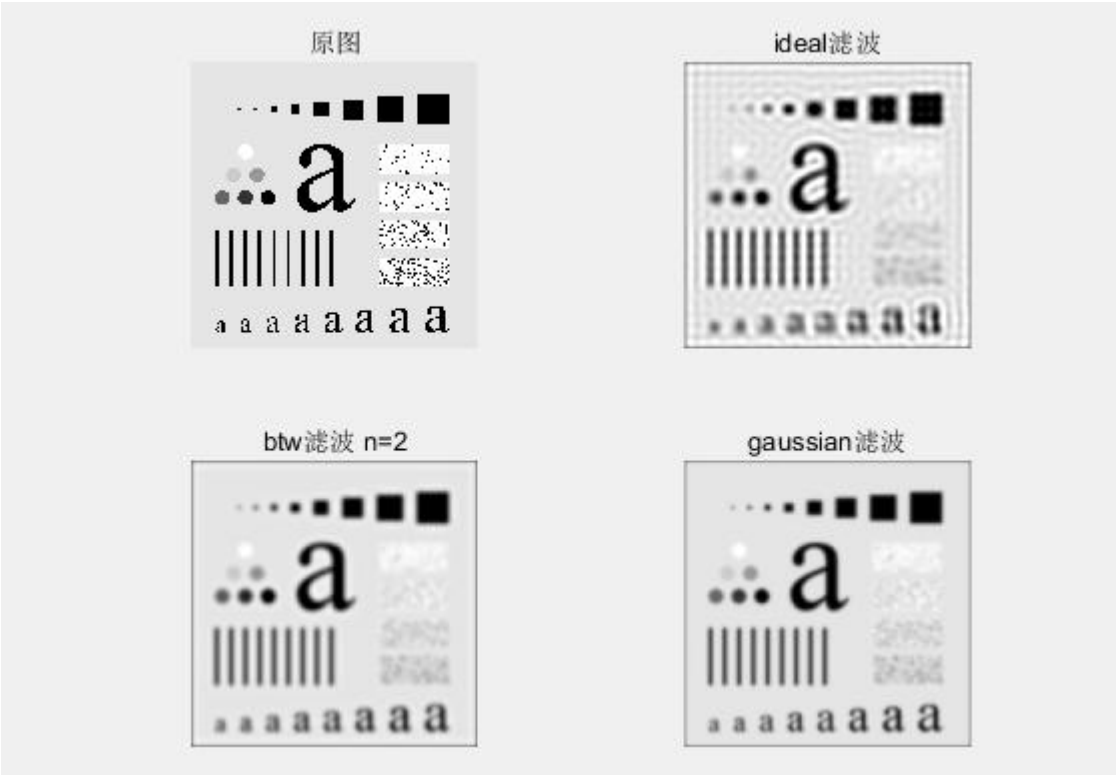
```
H = lpfilter(type, M, N, D0, n);
```

其中，type 为滤波器类型，分为'ideal'(理想)、'btw'(巴特沃斯)、'gaussian'(高斯)；M、N 为滤波器大小；D0 为截至频率，表示到滤波器中心的距离；n 为巴特沃斯阶数。

（5）例子

```
f = imread('Fig0313(a).tif');
```

```
PQ = paddedsize(size(f));
D0 = 0.05*PQ(2);
H1 = lpfilter('ideal', PQ(1), PQ(2), D0);
H2 = lpfilter('btw', PQ(1), PQ(2), D0, 2);
H3 = lpfilter('gaussian', PQ(1), PQ(2), D0);
g1 = dftfilt(f, H1);
g2 = dftfilt(f, H2);
g3 = dftfilt(f, H3);

figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(g1),title('ideal 滤波');
subplot(223),imshow(g2),title('btw 滤波 n=2');
subplot(224),imshow(g3),title('gaussian 滤波');
```



4．高通(锐化)频域滤波器

高通频域滤波与低通频域滤波相反，高通滤波通过削弱傅里叶变换的低频以及保持高频相对不变从而锐化图像。若低通滤波的传递函数为 $H_{LP}(u,v)$，则相应的高通滤波传递函数为：

$$H_{HP}(u,v)=1-H_{LP}(u,v)$$

（1）理想滤波器

理想低通滤波器切断的是圆外的分量，保留圆上、圆内的点不变。其传递函数如下：

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

其中，D_0 为正数，D(u,v)为点(u,v)到滤波器中心的距离。

（2）巴特沃斯滤波器

巴特沃斯滤波器与理想低通滤波器相同，截断圆外的分量；但不同的是，巴特沃斯版的传递函数比较连续。其传递函数如下：

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

其中，D_0 为正数，n 为阶数，D(u,v)为点(u,v)到滤波器中心的距离。

（3）高斯滤波器

$$H(u,v)=1-e^{-\frac{D^2(u,v)}{2\sigma^2}}$$

其中，σ 为标准差，代表 D_0；D(u,v)为点(u,v)到滤波器中心的距离。

（4）自定义函数

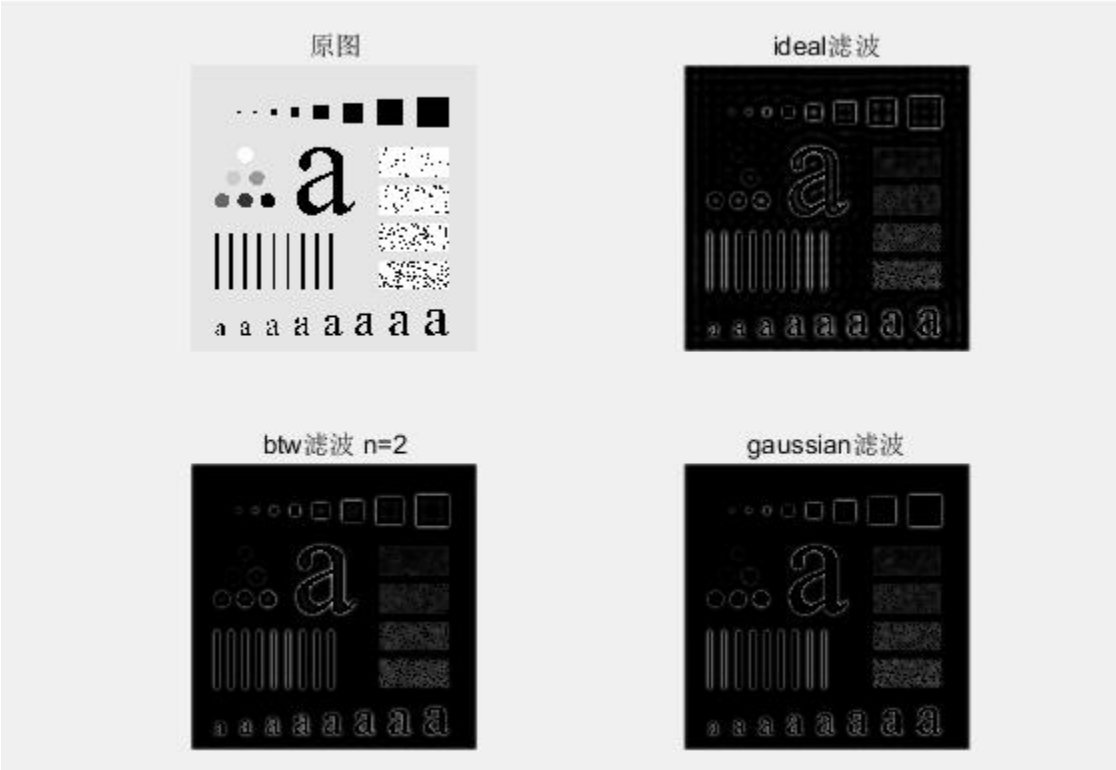
自定义函数 hpfilter 函数集成了定义以上三种滤波器，具体使用如下：

```
H = hpfilter(type, M, N, D0, n);
```

其中，type 为滤波器类型，分为'ideal'(理想)、'btw'(巴特沃斯)、'gaussian'(高斯)；M、N 为滤波器大小；D0 为截至频率，表示到滤波器中心的距离；n 为巴特沃斯阶数。

（5）例子

```
f = imread('Fig0313(a).tif');
PQ = paddedsize(size(f));
D0 = 0.05*PQ(2);
H1 = hpfilter('ideal', PQ(1), PQ(2), D0);
H2 = hpfilter('btw', PQ(1), PQ(2), D0, 2);
H3 = hpfilter('gaussian', PQ(1), PQ(2), D0);
g1 = dftfilt(f, H1);
g2 = dftfilt(f, H2);
g3 = dftfilt(f, H3);

figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(g1),title('ideal 滤波');
subplot(223),imshow(g2),title('btw 滤波 n=2');
subplot(224),imshow(g3),title('gaussian 滤波');
```



5．高频强调滤波

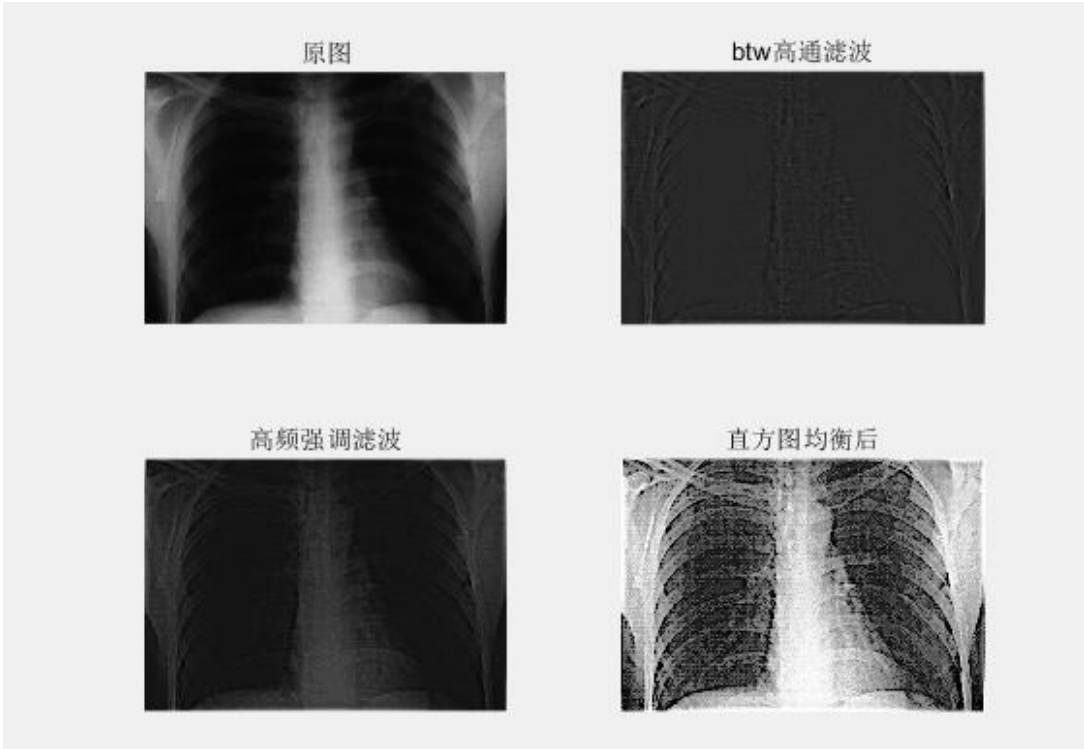
（1）高频强调滤波的传递函数：

$$H_{HFE}(u,v) = a + bH_{HP}(u,v)$$

其中，a 为偏移量，b 为乘数，$H_{HP}(u,v)$为高通滤波器的传递函数。

（2）例子

```
f = imread('Fig0319(a).tif');
PQ = paddedsize(size(f));
D0 = 0.05*PQ(1);

H1 = hpfilter('btw', PQ(1), PQ(2), D0, 2);  % 高通滤波器
H2 = 0.5 + 2.0 * H1;                        % 高频强调滤波器
g1 = dftfilt(f, H1, 'fltpoint');          % 频率域滤波
g1 = gscale(g1);                          % 范围拉伸
g2 = dftfilt(f, H2, 'fltpoint');          % 频率域滤波
g2 = gscale(g2);                          % 范围拉伸
g3 = histeq(g2, 256);                     % 直方图均衡
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(g1),title('btw 高通滤波');
subplot(223),imshow(g2),title('高频强调滤波');
subplot(224),imshow(g3),title('直方图均衡后');
```



（五）选择性滤波

1．带阻带通滤波器

带通滤波器保留的是环状上及环状内的分量。设带阻滤波器为 $H_{BR}(u,v)$，则相应的带通滤波器为

$$H_{BP}(u,v)=1-H_{BR}(u,v)$$

（1）理想带阻滤波器

理想低通滤波器切断的是圆外的分量，保留圆上、圆内的点不变。其传递函数如下：

$$H(u,v)=\begin{cases}1 & D_0-\dfrac{W}{2}\le D(u,v)\le D_0\\ 0 & \text{其他}\end{cases}$$

其中，D_0 为正数，D(u,v)为点(u,v)到滤波器中心的距离。

（2）巴特沃斯带阻滤波器

巴特沃斯滤波器与理想低通滤波器相同，截断圆外的分量；但不同的是，巴特沃斯版的传递函数比较连续。其传递函数如下：

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

其中，D_0 为正数，n 为阶数，D(u,v)为点(u,v)到滤波器中心的距离。

（3）高斯带阻滤波器

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

其中，σ 为标准差，代表 D_0；D(u,v)为点(u,v)到滤波器中心的距离。

（4）自定义函数

自定义函数 bandfilter 函数集成了定义以上滤波器，具体使用如下：

```
H = bandfilter(type, band, M, N, D0, W, n);
```

其中，type 为滤波器类型，分为'ideal'(理想)、'btw'(巴特沃斯)、'gaussian'(高斯)；band 为滤波器种类，分为'reject'(带阻)、'pass'(带通)；M、N 为滤波器大小；D0 为截至频率，表示到滤波器中心的距离；n 为巴特沃斯阶数。

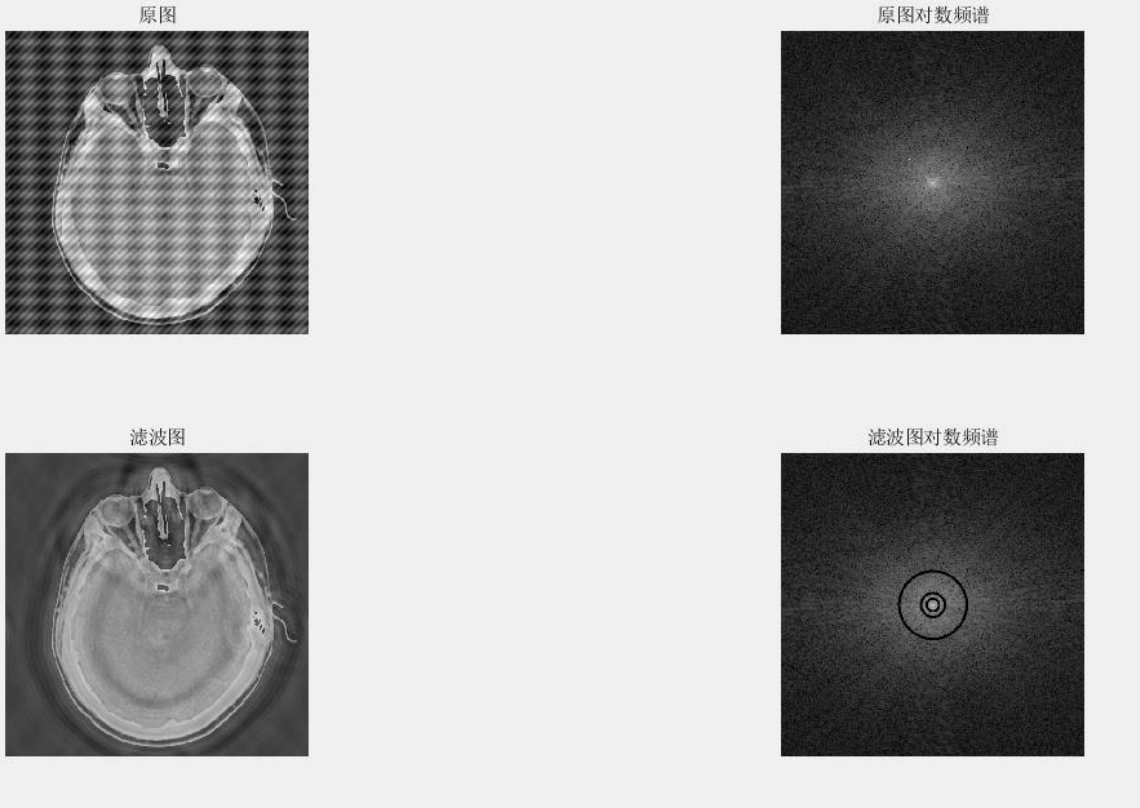
（5）函数 bandfilter2

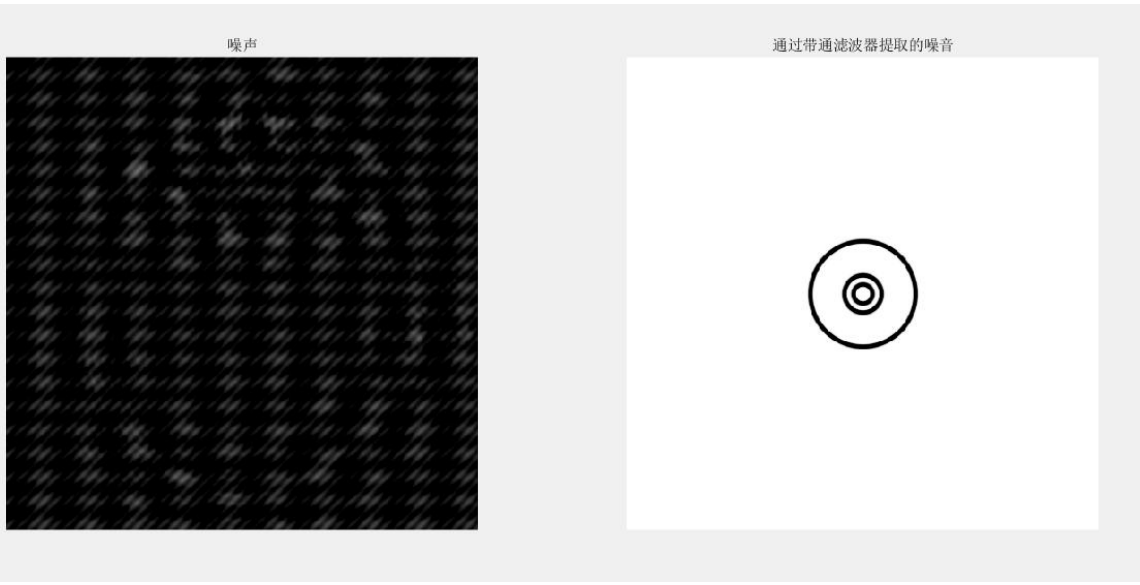
```
function H = bandfilter2(type, band, M, N, D0, W, n)
    [U, V] = dftuv(M, N);
    D = hypot(U, V);          % 生成频率网格点
    H = zeros(M, N);          % 距离
    k = numel(D0);            % 频带个数

    if numel(W) == 1
        W = W * ones(1, k);
    end
    if numel(n) == 1
        n = n * ones(1, k);
    end

    switch lower(type)
    case 'ideal'
        fun = @idealReject;
    case 'btw'
        fun = @btwReject;
    case 'gaussian'
        fun = @gaussianReject;
    end

    for i=1:k
        p = fun(D, D0(i), W(i), n(i));
        H = H + p;
    end

    H = H - min(H(:));
    H = H / max(H(:));
    if strcmp(band, 'pass')
        H = 1 - H;
    end
end

function H = btwReject(D, D0, W, n)
% 带阻滤波器
    H = 1 ./ (1 + (((D*W)./(D.^2-D0^2 + eps)).^(2*n)));
end

function H = gaussianReject(D, D0, W, n)
% 高斯带阻滤波器
    H = 1 - exp(-((D.^2 - D0^2)./(D.*W + eps)).^2);
end

function H = idealReject(D, D0, W, n)
```

```
    RI = D <= D0 - (W/2);   % Points of region inside the inner
                            % boundary of the reject band are labeled 1.
                            % All other points are labeled 0.

    RO = D >= D0 + (W/2);   % Points of region outside the outer
                            % boundary of the reject band are labeled 1.
                            % All other points are labeled 0.

    H = tofloat(RO | RI);   % Ideal bandreject filter.
end
```

（6）例子

```
f = imread('FigP0405.tif');
f = tofloat(f);
F = fft2(f);                    % DFT
Fc = fftshift(log(1+abs(F)));   % 对数频谱
% 带阻滤波器
H = bandfilter2('btw', 'reject', 512, 512, [10 20 57], 5, 5);
g = dftfilt(f, H);              % 频域滤波
G = fft2(g);                    % DFT
Gc = fftshift(log(1+abs(G)));   % 对数频谱

noise = f - g;
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(Fc, []),title('原图对数频谱');
subplot(223),imshow(g);title('滤波图');
subplot(224),imshow(Gc, []),title('滤波图对数频谱');
figure;
subplot(121),imshow(noise),title('噪声');
subplot(122),imshow(fftshift(H)),title('通过带通滤波器提取的噪音');
```





2．陷波带阻带通滤波器

（1）陷波带阻带通滤波器

陷波带通滤波器是通过事先定义的关于频率矩形中心领域内的频率。零相移滤波器必须关于原点对称，因此，以频率(u,v)为中心的开槽必须有相应的位于(-u,-v)处的开槽。设陷波带阻滤波器为 $H_{NR}(u,v)$，则相应的陷波带通滤波器为

$$H_{NP}(u,v)=1-H_{NR}(u,v)$$

一对开槽的一般形式：

$$H_{NR}(u,v)=\prod_{k=1}^{Q}H_k(u,v)H_{-k}(u,v)$$

其中，$H_k(u,v)$、$H_{-k}(u,v)$是高通滤波器，其中心分别为(u_k,v_k)、$(-u_k,-v_k)$，这些中心是以频率矩形的中心(M/2, N/2)来确定的。每个滤波器计算如下：

$$D_k(u,v)=[(\frac{u-M}{2-u_k})^2+(\frac{v-N}{2-v_k})^2]^{\frac{1}{2}}$$

$$D_{-k}(u,v)=[(\frac{u-M}{2+u_k})^2+(\frac{v-N}{2+v_k})^2]^{\frac{1}{2}}$$

（2）陷波带阻带通滤波器函数

- 函数 cnotch 可以生成陷波带阻带通滤波器，具体使用如下：

```
H = cnotch(type, notch, M, N, C, D0, n);
```

其中，type 为滤波器类型，分为'ideal'(理想)、'btw'(巴特沃斯)、'gaussian'(高斯)；notch 为滤波器种类，分为'reject'(陷波带阻)、'pass'(陷波带通)；M、N 为滤波器大小；C 为频率中心位置[u, v]；D0 为截至频率，表示到滤波器中心的距离；n 为巴特沃斯阶数。

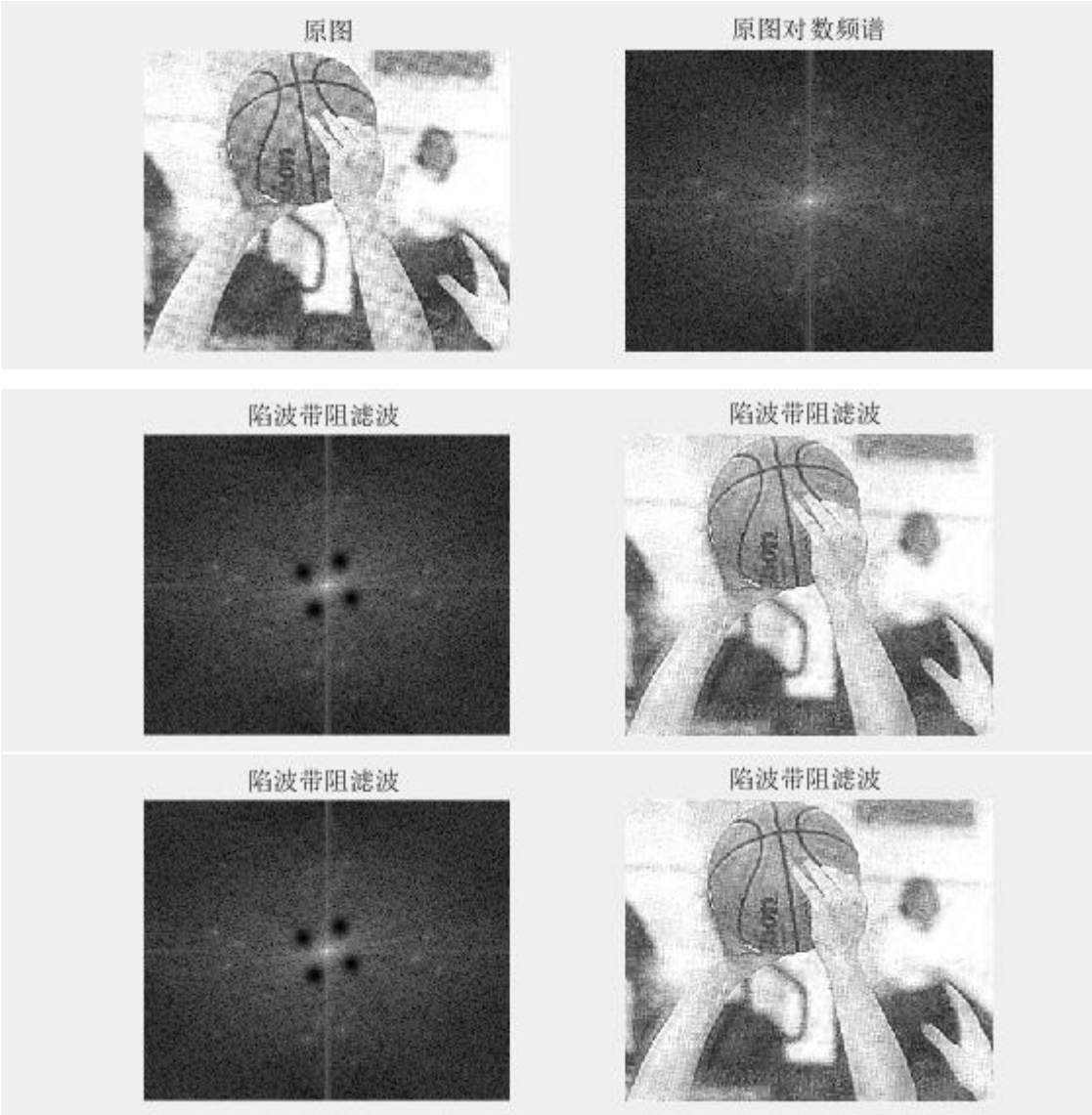
- 函数 recnotch 可以生成陷波带阻带通滤波器，新增包含沿着 DFT 的轴取值范围进行滤波的特殊情况，具体使用如下：

```
H = recnotch(notch, mode, M, N, W, SV, SH);
```

其中，notch 为滤波器种类，分为'reject'(陷波带阻)、'pass'(陷波带通)；mode 为滤波模式，分为'both'、'horizontal'、'vertical'；M、N 为滤波器大小；W 为滤波器宽度，默认值为 1；SV、SH 表示频率矩形的位置，即垂直方向在[-SV, SV]，水平方向在[-SH, SH]，默认值都为 1。

（3）例子(notch)

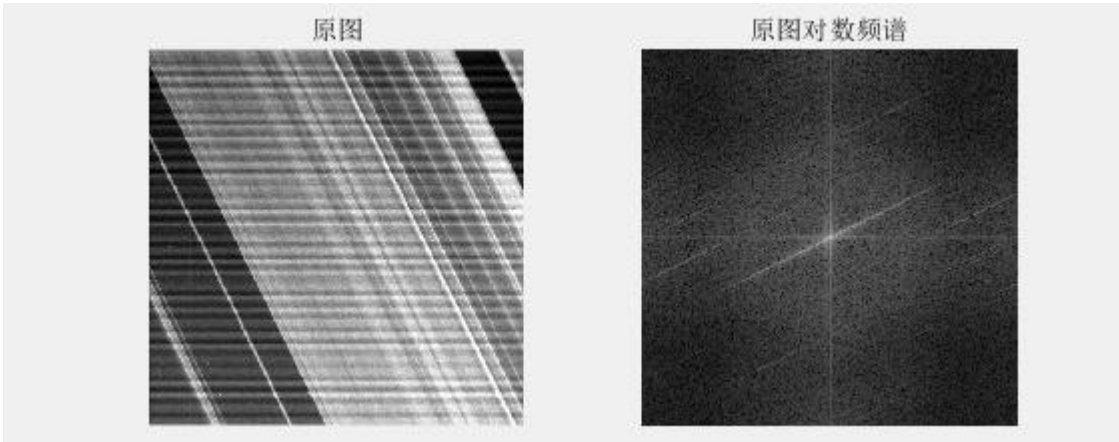
```
f = imread('Fig0321(a).tif');
[f, revertclass] = tofloat(f);
[M, N] = size(f);
F = fft2(f);                          % DFT
Fc = log(1 + abs(fftshift(F)));                          % 对数频谱
S = gscale(Fc);
C1 = [99 154; 128 163];
H1 = cnotch('gaussian', 'reject', M, N, C1, 5);      % 陷波带阻滤波器
p1 = gscale(fftshift(H1).*(tofloat(S)));
g1 = dftfilt(f, H1);                                     % 陷波带阻 C1 频域滤波
g1 = revertclass(g1);
figure;
subplot(121),imshow(f),title('原图');
subplot(122),imshow(S),title('原图对数频谱');
figure;
subplot(121),imshow(p1),title('陷波带阻滤波 ');
subplot(122),imshow(g1),title('陷波带阻滤波 ');

C2 = [99 154; 128 163; 49 160; 133 233; 55 132; 108 225; 112 74];
H2 = cnotch('gaussian', 'reject', M, N, C2, 5);      % 陷波带阻滤波器
p2 = gscale(fftshift(H2).*(tofloat(S)));
g2 = dftfilt(f, H2);                                     % 陷波带阻 C2 频域滤波
g2 = revertclass(g2);
figure;
subplot(121),imshow(p1),title('陷波带阻滤波 ');
subplot(122),imshow(g1),title('陷波带阻滤波 ');
```

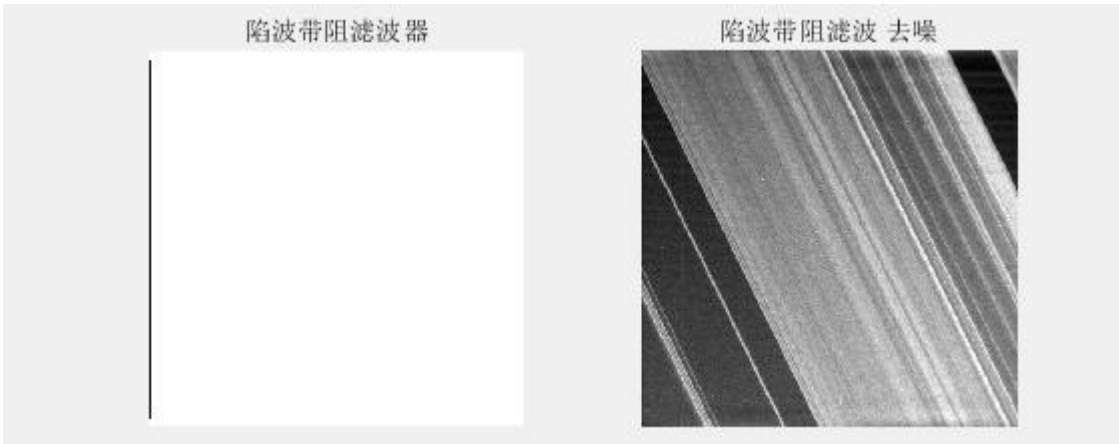


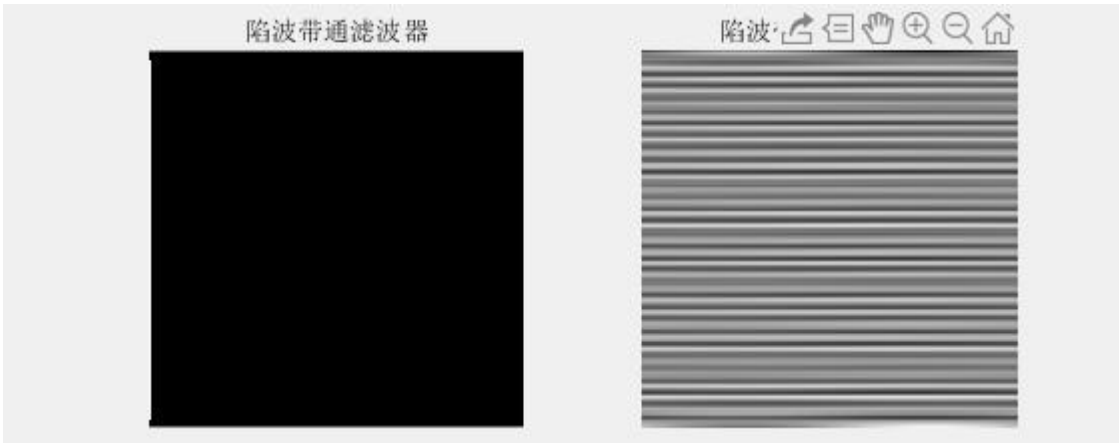
（4）例子(recnotch)

```
f = imread('Fig0322(a).tif');
[f, revertclass] = tofloat(f);
[M, N] = size(f);
F = fft2(f);
Fc = log(1 + abs(fftshift(F)));
S = gscale(Fc);
HR = recnotch('reject', 'vertical', M, N, 3, 15, 15);     % 陷波带阻滤波器
g1 = dftfilt(f, HR);              % 频域滤波 去掉空间干扰 即去噪
g1 = revertclass(g1);
HP = recnotch('pass', 'vertical', M, N, 3, 15, 15);       % 陷波带通滤波器
g2 = dftfilt(f, HP);              % 频域滤波 得到空间干扰 即噪音
g2 = gscale(g2);

figure;
subplot(121),imshow(f),title('原图');
subplot(122),imshow(S),title('原图对数频谱');
figure;
```

```
subplot(121),imshow(HR),title('陷波带阻滤波器');
subplot(122),imshow(g1),title('陷波带阻滤波 去噪');
figure;
subplot(121),imshow(HP),title('陷波带通滤波器');
subplot(122),imshow(g2),title('陷波带通滤波 噪音');
```







八、图像复原

（一）图像退化、复原模型

当噪声作用在输入图像 f(x,y)时，会产生一幅退化的图像：

$$g(x,y)=H[f(x,y)]+\eta(x,y)$$

而图像复原的目的就是得到原始图像的估计 $\hat{f}(x,y)$，并使估计图像接近原始图像。

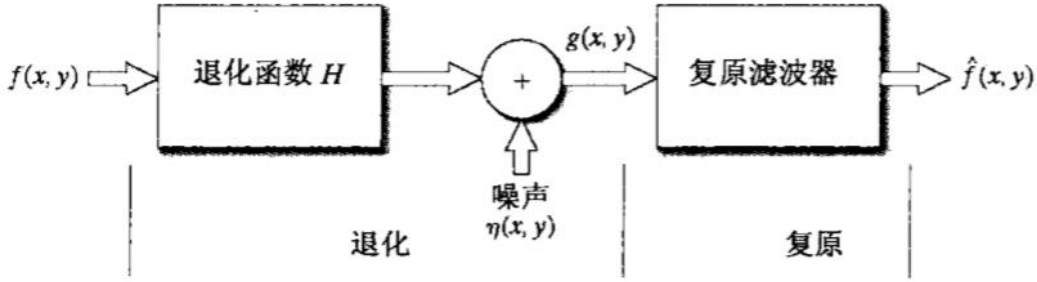


图 4-1　图像退化/复原处理的模型

上述模型的对应运算如下。

其中，H(u,v)也称为光传递函数(OTF)，h(x,y)称为点扩撒函数(PSF)。工具箱提供了 otf2psf、psf2otf 实现两者间的转换。

$$g(x,y)=h(x,y)\bigstar f(x,y)+\eta(x,y) \qquad \text{空间域滤波}$$

$$G(u,v)=H(x,y).*F(u,v)+N(u,v) \qquad \text{频率域滤波}$$

（二）噪声模型

模拟噪声的行为和效果是图像复原的核心。两种基本噪声模型：空间域噪声、频域噪声。

表 4-1　随机变量以及它们的 PDF、CDF 和随机数产生器

名　称	PDF	均值和方差	CDF	产生器
均匀	$p(z)=\begin{cases}\frac{1}{b-a} & 0\le z\le b\\ 0 & \text{其他}\end{cases}$	$m=\frac{a+b}{2},\ \sigma^2=\frac{(b-a)^2}{12}$	$F(z)=\begin{cases}0 & z<a\\ \frac{z-a}{b-a} & a\le z\le b\\ 1 & z>b\end{cases}$	MATLAB 函数 rand
高斯	$p(z)=\frac{1}{\sqrt{2\pi}b}e^{-(z-a)^2/2b^2}$　$-\infty<z<\infty$	$m=a,\ \sigma^2=b^2$	$F(z)=\int_{-\infty}^{z}p(v)dv$	MATLAB 函数 randn
对数正态	$p(z)=\frac{1}{\sqrt{2\pi}bz}e^{-[\ln(z)-a]^2/2b^2}$　$z>0$	$m=e^{a+(b^2/2)},\ \sigma^2=[e^{b^2}-1]e^{2a+b^2}$	$F(z)=\int_{0}^{z}p(v)dv$	$z=e^{bN(0,1)+a}$
瑞利	$p(z)=\begin{cases}\frac{2}{b}(z-a)e^{-(z-a)^2/b} & z\ge a\\ 0 & z<a\end{cases}$	$m=a+\sqrt{\pi b/4},\ \sigma^2=\frac{b(4-\pi)}{4}$	$F(z)=\begin{cases}1-e^{-(z-a)^2/b} & z\ge a\\ 0 & z<a\end{cases}$	$z=a+\sqrt{-b\ln[1-U(0,1)]}$
指数	$p(z)=\begin{cases}ae^{-az} & z\ge 0\\ 0 & z<0\end{cases}$	$m=\frac{1}{a},\ \sigma^2=\frac{1}{a^2}$	$F(z)=\begin{cases}1-e^{-az} & z\ge 0\\ 0 & z<0\end{cases}$	$z=-\frac{1}{a}\ln[1-U(0,1)]$
厄兰	$p(z)=\frac{a^b z^{b-1}}{(b-1)!}e^{-az}$　$z\ge 0$	$m=\frac{b}{a},\ \sigma^2=\frac{b}{a^2}$	$F(z)=\left[1-e^{-az}\sum_{n=0}^{b-1}\frac{(az)^n}{n!}\right]$　$z\ge 0$	$z=E_1+E_2+\cdots+E_b$ (E_i是带有参数 a 的指数随机数)
椒盐	$p(z)=\begin{cases}P_p & z=0\ \text{(pepper)}\\ P_s & z=2^n-1\ \text{(salt)}\\ 1-(P_p+P_s) & z=k\\ & (0<k<2^n-1)\end{cases}$	$m=(0)P_p+k(1-P_p-P_s)+(2^n-1)P_s$ $\sigma^2=(0-m)^2P_p+(k-m)^2(1-P_p-P_s)+(2^n-1-m)^2P_s$	$F(z)=\begin{cases}0 & z<0\\ P_p & 0\le z<k\\ 1-P_s & k\le z<2^n-1\\ 1 & 2^n-1\le z\end{cases}$	具体附加逻辑的 MATLAB 函数 rand

1．空间随机噪声—imnoise 函数

（1）函数

```
g = imnoise(f, type, parameters);
```

其中，f 是输入图像，输出图像 g 为带噪的退化图像。

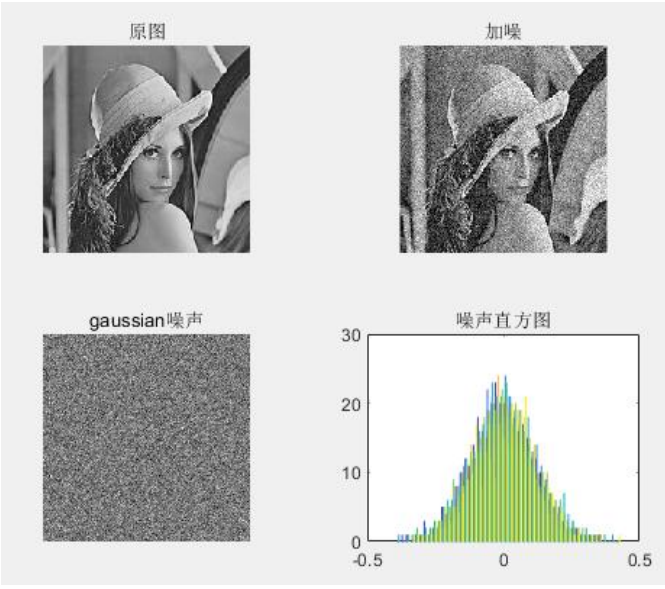
```
g = imnoise(f, 'gaussian', m=0, var=0.01);均值为 0、方差为 0.01 的高斯噪声加到 f
```

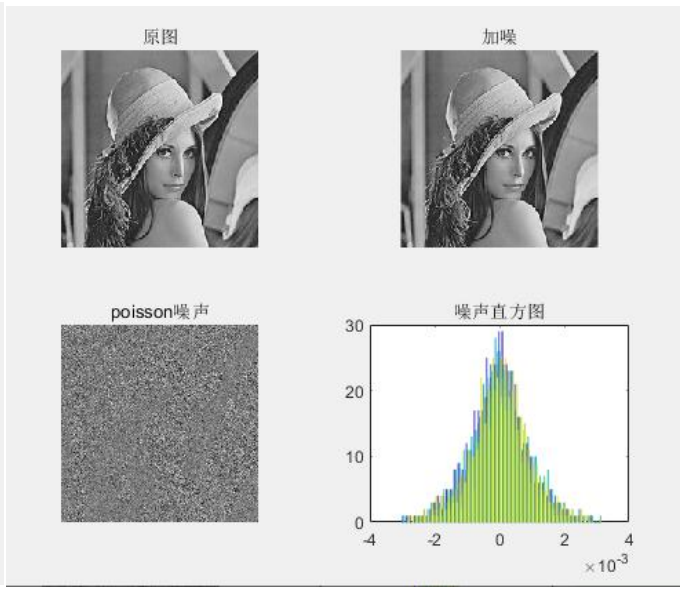
```
g = imnoise(f, localvar, V); 均值为 0、局部方差为 V 的高斯噪声加到 f, lovalvar 是 f 每个点的理想方差值
g = imnoise(f, 'localvar', image_intensity, var); 均值为 0、局部方差为 V 的高斯噪声加到 f, var 是图像的灰度值函数, image_intensity 包含归一化的灰度值
g = imnoise(f, 'salt & pepper', d); 椒盐噪声加到 f, d 为噪声密度
g = imnoise(f, 'speckle', var=0.04); 用 g=f+n.*f 这种乘性噪声加到 f, n 是均值为 0、方差为 0 的均匀分布的随机噪声
g = imnoise(f, 'poisson'); 将泊松噪声加到 f
```

（2）例子

```
function fig_plot(f, g, type)
    n = g - f;
    figure;
    subplot(221),imshow(f),title('原图');
    subplot(222),imshow(g),title('加噪');
    subplot(223),imshow(n, []),title(type+"噪声");
    subplot(224),hist(n, 50),title('噪声直方图');
end
```

```
f = imread("pic_Lena.jpg");
f = rgb2gray(f);
f = tofloat(f);
g1 = imnoise(f, 'gaussian', 0, 0.01);   % [均值, 方差]
g2 = imnoise(f, 'poisson');

fig_plot(f, g1, "gaussian");
fig_plot(f, g2, "poisson");
```





2．空间随机噪声—imnoise2 函数

（1）函数

```
r = imnoise2(type, parameters);
```

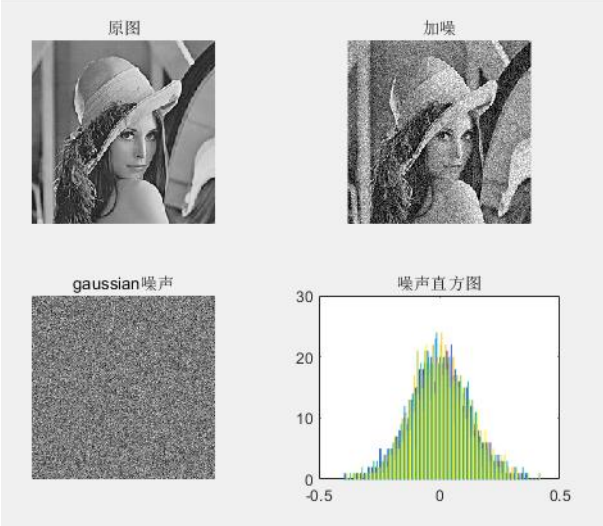
其中，type 是噪声类型，r 为噪声模型。

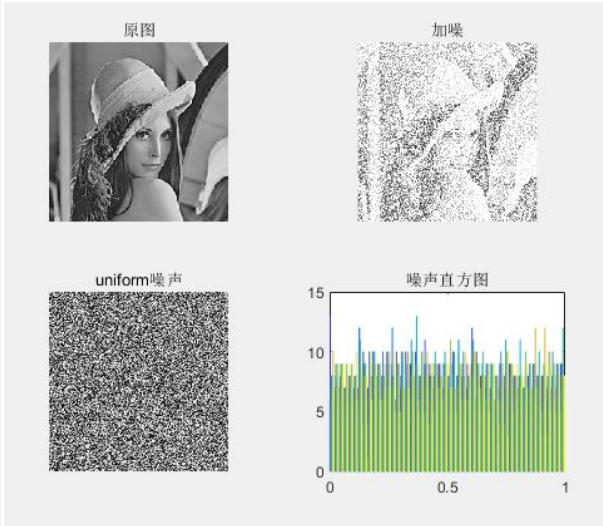
```
r = imnoise2('uniform', M, N, a, b); 均匀随机噪声
r = imnoise2('gaussian', M, N, a, b); 高斯随机噪声
```

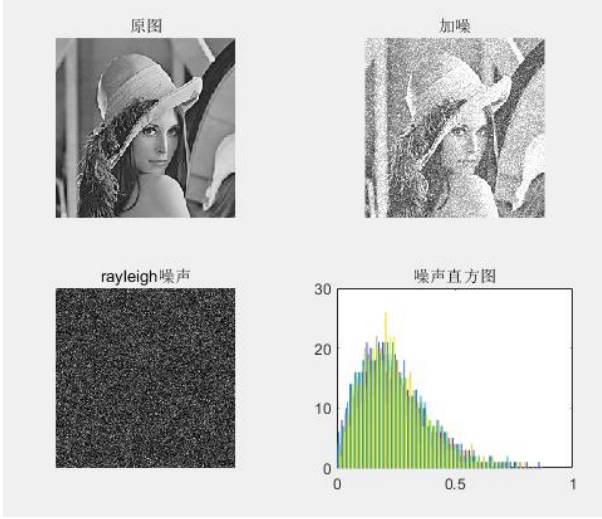
```
r = imnoise2('salt & pepper', M, N, a=0.05, b=0.05); 椒盐随机噪声
r = imnoise2('lognormal', M, N, a=1, b=0.25); 对数正态随机噪声
r = imnoise2('rayleigh', M, N, a, b); 瑞利随机噪声
r = imnoise2('exponential', M, N, a=1); 指数随机噪声
r = imnoise2('erlang', M, N, a=2, b=5); 厄兰随机噪声
```

（2）例子

```
f = imread("pic_Lena.jpg");
f = rgb2gray(f);
f = tofloat(f);
[M, N] = size(f);
noise1 = imnoise2('gaussian', M, N, 0, 0.1); % [均值, 标准差]
g1 = noise1 + f;
noise2 = imnoise2('uniform', M, N);
g2 = noise2 + f;
noise3 = imnoise2('rayleigh', M, N);
g3 = 0.25*noise3 + f;

fig_plot(f, g1, "gaussian");
fig_plot(f, g2, "uniform");
fig_plot(f, g3, "rayleigh");
```







3．周期噪声—imnoise3 函数

（1）周期噪声模型

其中 A 为振幅，Bx、By 为关于原点发相移，u0、v0 为关于 x 轴 y 轴的正弦频率。

$$r(x,y)=A\sin[\frac{2\pi u_0}{M}(x+B_x)+\frac{2\pi v_0}{N}(y+B_y)]$$

$$R(u,v)=j\frac{AMN}{2}[e^{-j2\pi(\frac{u_0B_x}{M}+\frac{v_0B_y}{N})}\delta(u+u_0,v+v_0)-e^{j2\pi(\frac{u_0B_x}{M}+\frac{v_0B_y}{N})}\delta(u-u_0,v-v_0)]$$

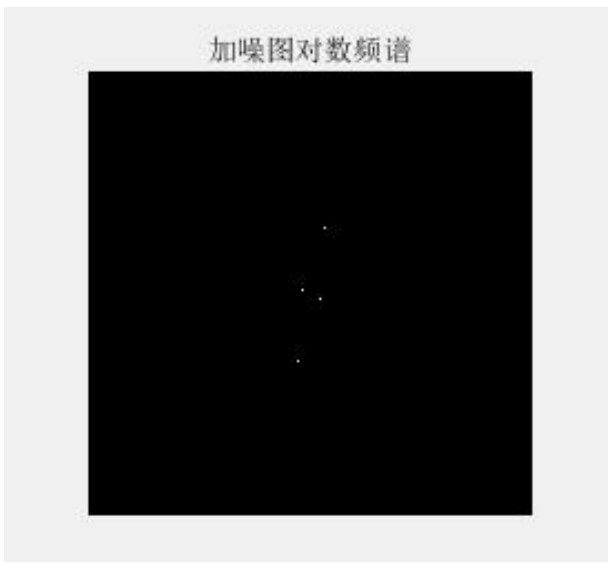
（2）函数

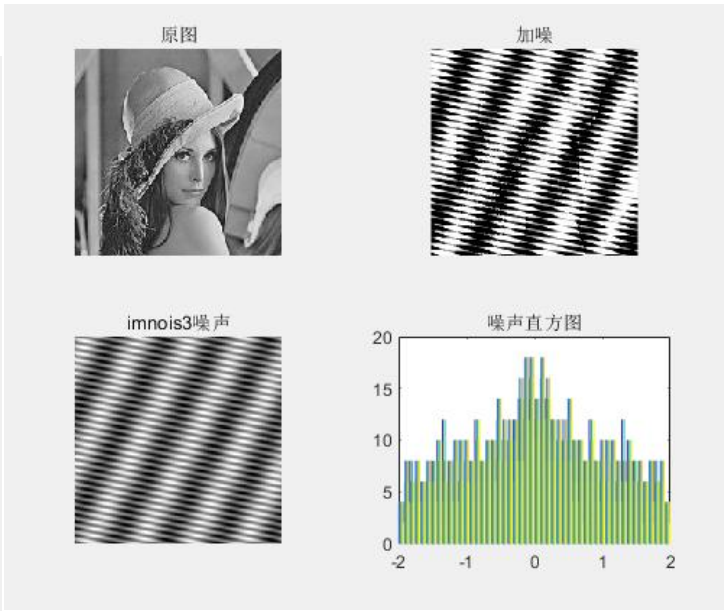
```
[r, R, S] = imnoise3(M, N, C, A, B);
```

其中，M、N 为噪声大小；C 为[k, 2]的脉冲坐标；A 为振幅；B 为初始相角；r 为噪声模型，R 为 r 的 DFT 变化，S 为 r 的频谱。

（3）例子

```
f = imread("pic_Lena.jpg");
f = rgb2gray(f);
f = tofloat(f);
[M, N] = size(f);
C = [2 4; 30 -6];
[n, N, S] = imnoise3(M, N, C);
g = n + f;
figure;imshow(S, []),title("加噪图对数频谱")
[x, y] = find(S>0.5);
fig_plot(f, g, "imnois3");
```





（三）估计噪声参数

1．方法

对于周期噪声，可以分析退化图像的傅立叶频谱，通过目测可找到频率尖峰，之后通过带阻、陷波带阻等方法去噪。

对于空间域噪声，通过选取退化图像的感兴趣区域(ROI 区域)来估计均值和方差，即 a、b 的值。

2．步骤

（1）选取 ROI 区域

此函数会弹出图像窗口来选择区域，选择完成后双击区域完成选定。

```
[B, c, r] = roipoly(f);
```

其中，f 为输入图像；B 为对应选定 ROI 区域的逻辑数组；c、r 为图像 ROI 区域边界上点的列坐标、行坐标。

（2）计算 ROI 区域的直方图

```
[h, npix] = histroi(f, c, r);
```

其中，f 为输入图像；c、r 为图像 ROI 区域边界上点的列坐标、行坐标；返回值 h 为 ROI 区域直方图；npix 为 ROI 区域的像素个数。

（3）计算中心矩

令 z_i 是用来表示一幅图像灰度级的离散随机变量，并且令 $p(z_i)(i=0,1,2...L-1)$是相应的归一化直方图。其中，L 是可能的灰度值的数目。直方图的分量 $p(z_i)$是灰度值 z_i 出现概率的估计，这个直方图也可以被看做灰度 PDF 的近似。

描述直方图形状的主要方法之一是使用直方图的中心矩(也被称做平均值的矩)，定义如下，易知，$u_0=1,u_1=0,u_2=\sigma^2$。

$$\mu_n = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i)$$

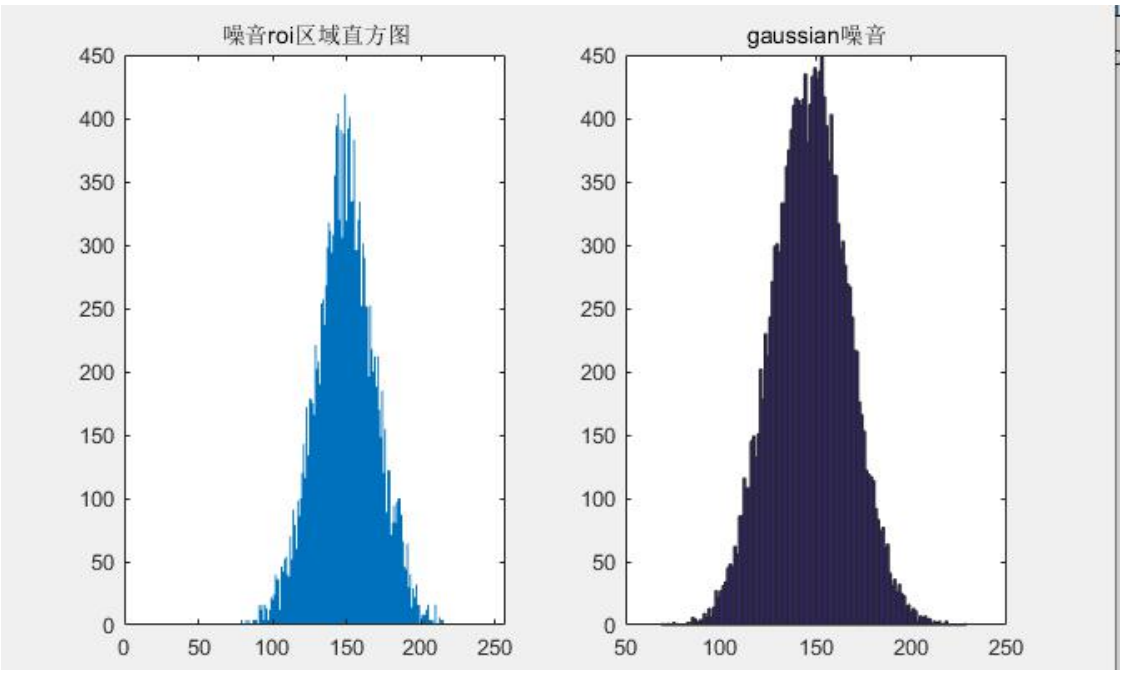
其中，n 为矩的阶，m 为均值。$m = \sum_{i=0}^{L-1} z_i p(z_i)$

```
[v, unv] = statmoments(p, n);
```

其中，p 为 ROI 区域的直方图；n 为计算的矩的数量；v、unv 分别是计算的中心距的值，两者区别在于 v 为归一化的值。

3．例子

```
f = imread('Fig0404(a).tif');
[B, c, r] = roipoly(f);          % 获得 roi 区域
[p, npix] = histroi(f, c, r);    % 计算 roi 直方图
[v, unv] = statmoments(p, 2);    % 计算中心距
% u = unv(1); sigma = unv(2);
X = imnoise2('gaussian', npix, 1, 147, 20);
figure;
subplot(121),bar(p,1);title('噪音 roi 区域直方图');
subplot(122),hist(X, 130);title('gaussian 噪音');
```



（四）仅有噪声的复原——去噪

1．模型

当退化模型只有噪声时，对应的模型如下。此时选择降低噪声的方法为空间滤波。

$$g(x,y)=f(x,y)+\eta(x,y)$$

2．空间噪声滤波器——去随机噪声

（1）函数

```
f = spfilt(g, type, varargin); 其中，M=N=3(默认值).
f = spfilt(g, 'amean', M, N);          算术平均
f = spfilt(g, 'gmean', M, N);          几何平均
f = spfilt(g, 'hmean', M, N);          调和平均
f = spfilt(g, 'chmean', M, N, Q=1.5); 反调和平均
f = spfilt(g, 'median', M, N);         中值
f = spfilt(g, 'max', M, N);            最大值
f = spfilt(g, 'midpoint', M, N);       中点值
f = spfilt(g, 'min', M, N);            最小值
f = spfilt(g, 'atrimmed', M, N, D=2); a 修正
```

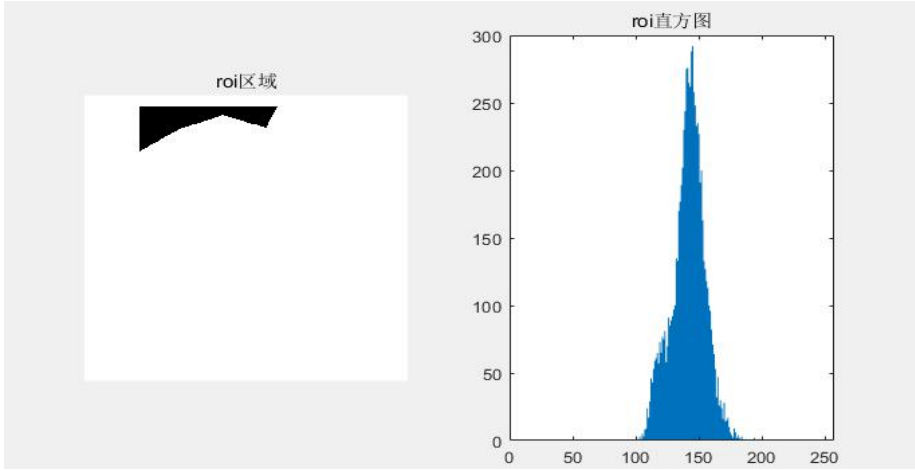
表 4-3　空间滤波器：变量 m 和 n 分别表示滤波器跨越的行数和列数

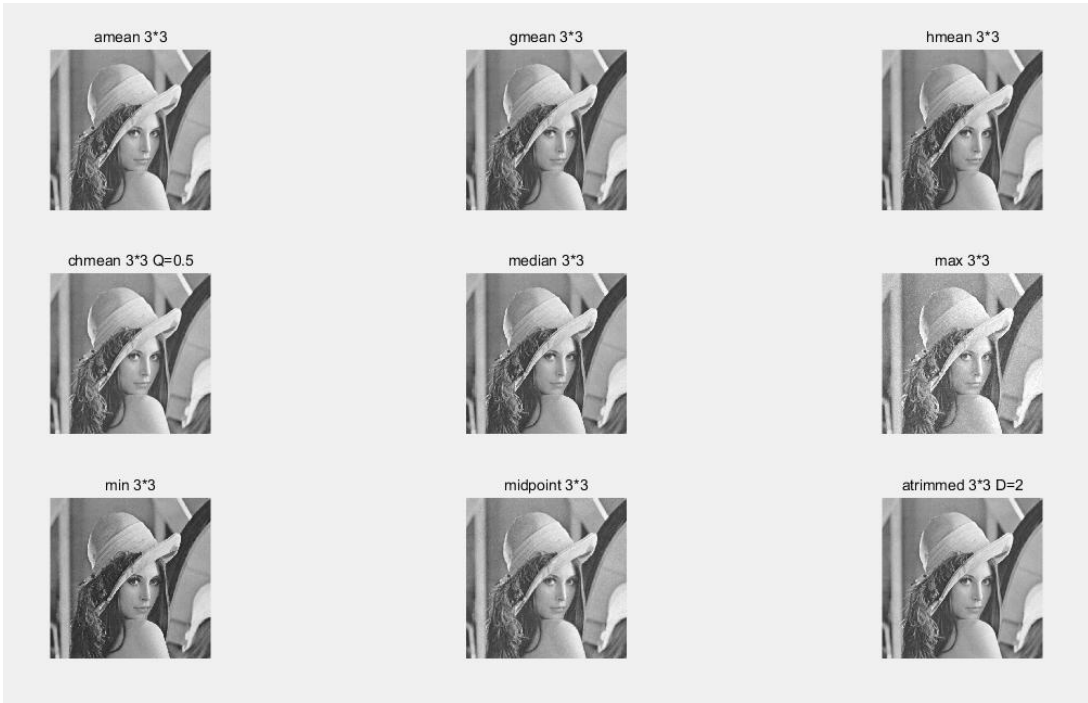
滤波器名称	公　式	注　释
算术均值	$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}}g(s,t)$	用工具箱函数 w =fspecial('average',[m, n]) 和 f = imfilter(g, w)来实现
几何均值	$\hat{f}(x,y)=\left[\prod_{(s,t)\in S_{xy}}g(s,t)\right]^{\frac{1}{mn}}$	该非线性滤波器用函数 gmean(见本节的自定义函数 spfilt)来实现
调和均值	$\hat{f}(x,y)=\frac{mn}{\sum_{(s,t)\in S_{xy}}\frac{1}{g(s,t)}}$	该非线性滤波器用函数 harmean(见本节的自定义函数 spfilt)来实现
反调和均值	$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}}g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}}g(s,t)^{Q}}$	该非线性滤波器用函数 charmean(见本节的自定义函数 spfilt)来实现
中值	$\hat{f}(x,y)=\underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$	用工具箱函数 medfilt2 来实现：f = medfilt2(g, [m n],'symmetric')
最大值	$\hat{f}(x,y)=\max_{(s,t)\in S_{xy}}\{g(s,t)\}$	用工具箱函数 imdilate 来实现：f = imdilate(g, ones(m, n))
最小值	$\hat{f}(x,y)=\min_{(s,t)\in S_{xy}}\{g(s,t)\}$	用工具箱函数 imerode 来实现：f = imerode(g, ones(m, n))
中点值	$\hat{f}(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right]$	由最大、最小滤波结果的 0.5 倍来实现
字母平衡值	$\hat{f}(x,y)=\frac{1}{mn-d}\sum_{(s,t)\in S_{xy}}g_r(s,t)$	在 S_{xy} 中，选择 $g(s,t)$ 的 $d/2$ 最低像素值和 $d/2$ 最高像素值。函数 $g(s,t)$ 表示在邻域中保留 $mn-d$ 个像素。用函数 alphatrim 实现(见本节的自定义函数 spfilt)

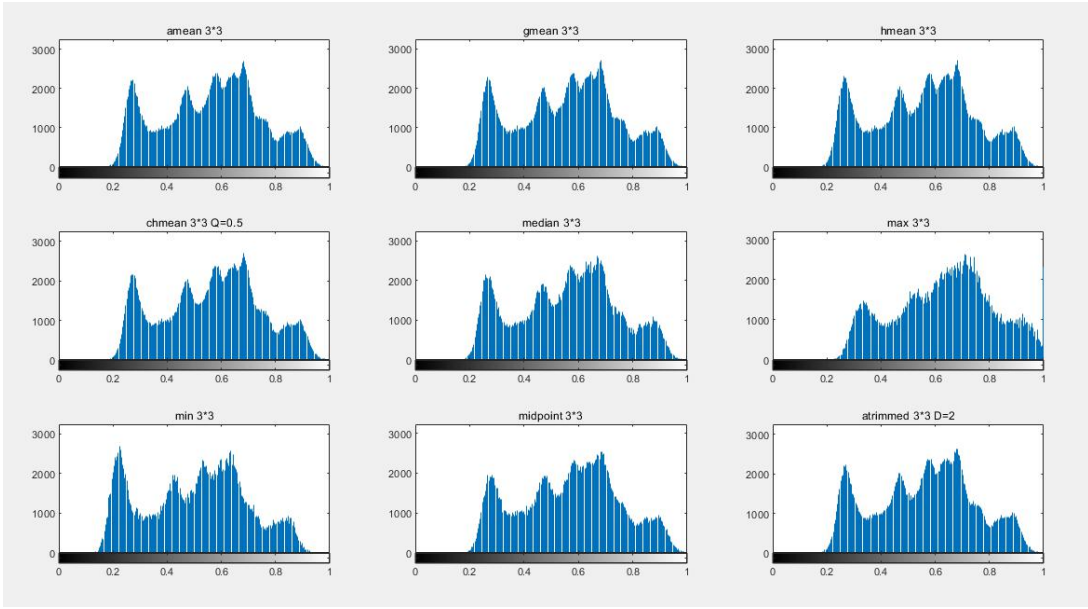
（2）例子

```
f = imread('pic_Lena_gray_512.tif');
f = tofloat(f);
[M, N] = size(f);
n = imnoise2('erlang', M, N);            % 爱尔兰噪声
g = f + 0.03*n;
```

```
[B, c, r] = roipoly(g);                 % 选择 roi 区域
[h, npix] = histroi(g, c, r);           % 计算 roi 直方图
[v, unv] = statmoments(h, 2);           % 计算 mu, sigma
figure;
subplot(121),imshow(1-B),title('roi 区域');
subplot(122);bar(h, 1),title('roi 直方图');

g1 = spfilt(g, 'amean', 3, 3);
g2 = spfilt(g, 'gmean', 3, 3);
g3 = spfilt(g, 'hmean', 3, 3);
g4 = spfilt(g, 'chmean', 3, 3, 0.5);
g5 = spfilt(g, 'median', 3, 3);
g6 = spfilt(g, 'max', 3, 3);
g7 = spfilt(g, 'min', 3, 3);
g8 = spfilt(g, 'midpoint', 3, 3);
g9 = spfilt(g, 'atrimmed', 3, 3, 2);
figure;
subplot(331),imshow(g1),title('amean 3*3');
subplot(332),imshow(g2),title('gmean 3*3');
subplot(333),imshow(g3),title('hmean 3*3');
subplot(334),imshow(g4),title('chmean 3*3 Q=0.5');
subplot(335),imshow(g5),title('median 3*3');
subplot(336),imshow(g6),title('max 3*3');
subplot(337),imshow(g7),title('min 3*3');
subplot(338),imshow(g8),title('midpoint 3*3');
subplot(339),imshow(g9),title('atrimmed 3*3 D=2');
figure;
subplot(331),imhist(g1),title('amean 3*3');
subplot(332),imhist(g2),title('gmean 3*3');
subplot(333),imhist(g3),title('hmean 3*3');
subplot(334),imhist(g4),title('chmean 3*3 Q=0.5');
subplot(335),imhist(g5),title('median 3*3');
subplot(336),imhist(g6),title('max 3*3');
subplot(337),imhist(g7),title('min 3*3');
subplot(338),imhist(g8),title('midpoint 3*3');
subplot(339),imhist(g9),title('atrimmed 3*3 D=2');
```


roi区域
roi直方图
300
250
200
150
100
50
0
0
50
100
150
200
250


amean 3*3
gmean 3*3
hmean 3*3
chmean 3*3 Q=0.5
median 3*3
max 3*3
min 3*3
midpoint 3*3
atrimmed 3*3 D=2


amean 3*3
gmean 3*3
hmean 3*3
chmean 3*3 Q=0.5
median 3*3
max 3*3
min 3*3
midpoint 3*3
atrimmed 3*3 D=2

3．自适应中值空间滤波器

（1）原理

S_{xy} 表示一幅将被处理的中心位于(x, y)的子图像。算法如下：

令

Z_{min} = S_{xy} 中的最小亮度值

Z_{max} = S_{xy} 中的最大亮度值

Z_{med} = S_{xy} 中的中值，

Z_{xy} = 坐标(x, y)处的亮度值

自适应中值滤波算法工作在两个层面，表示为 level A 和 level B：

Level A：如果 $Z_{min}<Z_{med}<Z_{max}$，转向 level B；否则增大窗口尺寸

如果窗口尺寸小于等于 S_{max}，重复 level A；否则输出 Z_{med}

Level B：如果 $Z_{min}<Z_{xy}<Z_{max}$，输出 Z_{xy}；否则输出 Z_{med}

其中，S_{max} 表示自适应滤波器窗口允许的最大尺寸。Level A 最后一步的另一种选择是输出 Z_{xy} 代替中值。这将产生稍微清楚一些的结果，但是却可能探测不到与椒盐噪声的值相同的内含于常数背景的盐粒(胡椒)噪声。

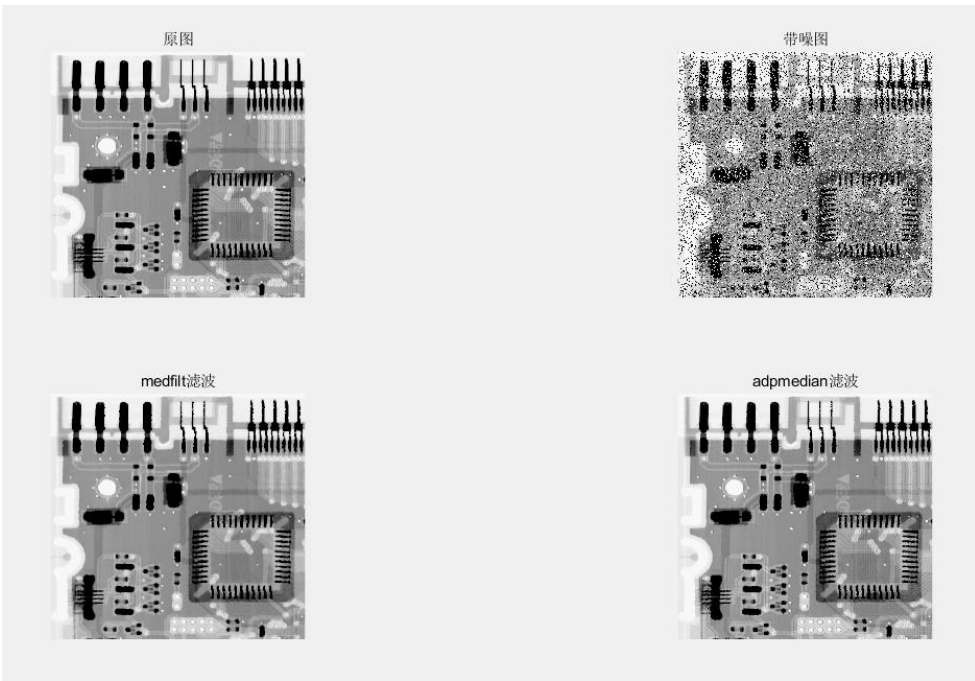
（2）函数

```
f = adpmedian(g, Smax);
```

其中，g 为退化图像；Smax 是允许的最大自适应滤波器的尺寸。

（3）例子

```
f = imread('Fig0219(a).tif');
g = imnoise(f, 'salt & pepper', 0.25);
f1 = medfilt2(g, [7 7], 'symmetric');
f2 = adpmedian(g, 7);
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(g),title('带噪图');
subplot(223),imshow(f1),title('medfilt 滤波');
subplot(224),imshow(f2),title('adpmedian 滤波');
```



4．空间噪声滤波器——去周期噪声

（1）带阻带通滤波器

（2）陷波带阻带通滤波器

以上详见七·（五）

（五）带有卷积的复原——去卷积

1．模型

当退化模型中具有卷积和噪声时，当噪声影响不大时，主要操作是去卷积。对应模型如下：

$$g(x,y)=f(x,y)**h(x,y)+\eta(x,y)$$

2．测试图像

测试图像使用棋盘测试图像，因为其可以缩放，且不会影响主要特征。

```
C = checkerboard(NP, M, N);
```

其中，NP 为每个正方形边长的像素数，默认值为 10；M、N 为行数和列数，默认值为 8；输出图像 C 为 double 类型。

3．放大图像

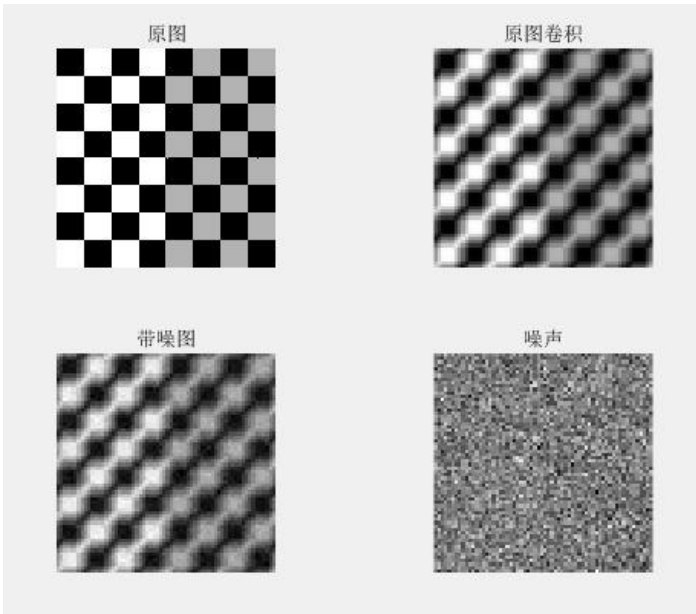
通过像素复制来放大图像。

```
B = pixeldup(A, m, n);
```

其中，A 为图像，m、n 表示分别将图像在垂直方向和水平方向复制的次数。

4．例子

```
f = checkerboard(8);
[M, N] = size(f);
h = fspecial('motion', 7, 45);
gb = imfilter(f, h, 'circular');
n = imnoise2('gaussian', M, N, 0, sqrt(0.001));
g = gb + n;
figure;
subplot(221),imshow(pixeldup(f, 8), []),title('原图');
subplot(222),imshow(pixeldup(gb, 8), []),title('原图卷积');
subplot(223),imshow(pixeldup(g, 8), []),title('带噪图');
subplot(224),imshow(pixeldup(n, 8), []),title('噪声');
```



5. 图像复原

（1）直接逆滤波

对于下面的模型，若考虑噪声，可得到原图像的估计。但噪声是一个随机函数，其 Fourier 变换 N(u,v)是未知的，而 H(u,v)在(0,0)处的值较大，远离中心位置的值接近于 0，故估计的图像几乎由噪声支配，效果不乐观。

$$G(u,v)=H(u,v)*F(u,v)+N(u,v)$$

$$\hat{F}(u,v)=F(u,v)+\frac{N(u,v)}{H(u,v)}$$

（2）维纳滤波

原理：$\min e^2=\min E\{(f-\hat{f})^2\}$

图像的估计：$\hat{F}(u,v)=\left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+\frac{|N(u,v)|^2}{|F(u,v)|^2}}\right]*G(u,v)$

其中，$\frac{|N(u,v)|^2}{|H(u,v)|^2}$ 称为噪性功率比，用 $R=\frac{\frac{1}{MN}\sum_u\sum_v|N(u,v)|^2}{\frac{1}{MN}\sum_u\sum_v|F(u,v)|^2}$ 近似计算。

（3）自相关逆滤波

原理：$|F(u,v)|^2=\mathrm{DFT}(f(x,y)\star f(x,y))$

（4）M 函数

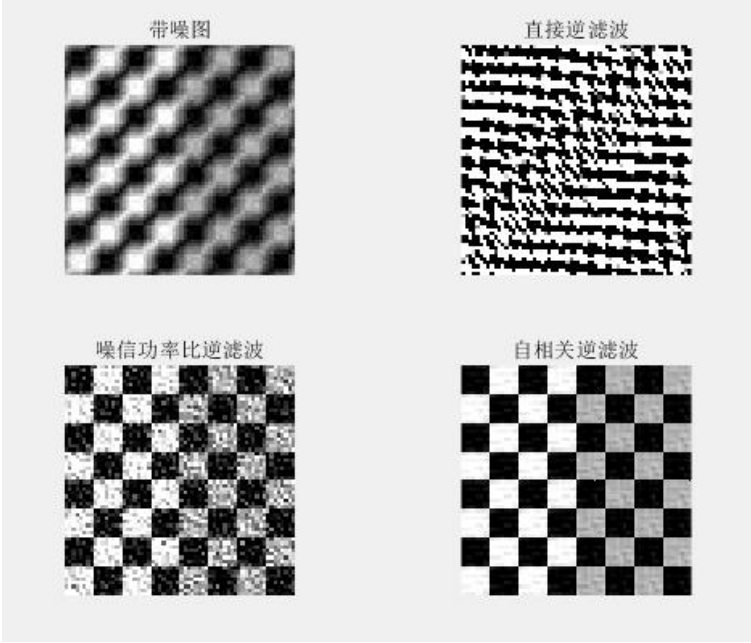
```
f = deconvwnr(g, h, parameters);
f = deconvwnr(g, h);                    直接逆滤波
f = deconvwnr(g, h, R);                 维纳滤波
f = deconvwnr(g, h, NACORR, FACORR);    自相关逆滤波
```

其中，f 为复原图像，g 为带噪图像，h 为滤波器，R 为噪性功率比，NACORR、FACORR 为噪声 noise 和原图 f 的自相关。

（5）例子

```
f = checkerboard(8);
[M, N] = size(f);
h = fspecial('motion', 7, 45);  % 滤波器

n = imnoise2('gaussian', M, N, 0, sqrt(0.001)); % 噪声
Sn = abs(fft2(n)).^2;
nA = sum(Sn(:)) / (M*N);
Sf = abs(fft2(f)).^2;
fA = sum(Sf(:)) / (M*N);
R = nA / fA;                    % 噪信功率比
gb = imfilter(f, h, 'circular');% 卷积
g = gb + n;                     % 加噪图
f1 = deconvwnr(g, h);           % 直接逆滤波
f2 = deconvwnr(g, h, R);        % 维纳滤波
NCORR = fftshift(real(ifft2(Sn)));
ICORR = fftshift(real(ifft2(Sf)));
```

```
f3 = deconvwnr(g, h, NCORR, ICORR); % 自相关逆滤波
figure;
subplot(221),imshow(pixeldup(g, 8)),title('带噪图');
subplot(222),imshow(pixeldup(f1, 8)),title('直接逆滤波');
subplot(223),imshow(pixeldup(f2, 8)),title('噪信功率比逆滤波');
subplot(224),imshow(pixeldup(f3, 8)),title('自相关逆滤波');
```



九、彩色图像

（一）彩色图像表示

1．彩色图像展示

（1）RGB 图像

一幅 RGB 图像就是 M×N×3 大小的彩色像素的数组，其中的每个彩色像素点都是在特定空间位置的彩色图像所对应的红、绿、蓝三个分量，RGB 图像可以看做是由三个灰度图像形成的“堆栈”。

（2）索引图像

索引图像有两个分量：整数数据矩阵 X 和彩色映射矩阵 map。矩阵 map 是 m×3 大小、由 double 类型且范围在[0,1]之间的浮点数构成的数组。map 的长度 m 等于定义的颜色数。map 的每一行都定义有单色的红、绿、蓝分量。索引图像将像素的亮度值“直接映射”到彩色值。每个像素的颜色由对应的整数矩阵 X 的值作为指向 map 的索引决定。

（3）相关函数

```
rgbcube();                          彩色立方体
g = cat(dim, fr, fg, fb);           连接三幅灰度图为彩色图
whitebg('g' | 'green' | [0 1 0]);  修改图像背景色
```

参数说明：

dim: 1,垂直方向拼接；2，水平方向拼接；3，三维方式堆叠。

（4）例子

```
load mandrill;
colormap(copper);
subplot(121), imshow(X, map),title('索引图像');
subplot(122), imshow(X, copper),title('预定义的彩色映射');
```


```
X = zeros(200);
X(:, 1:50) = 1;
X(:, 51:100) = 64;
X(:, 101:150) = 128;
X(:, 151:200) = 256;
map = [0 0 0; 1 1 1];
figure,
subplot(121),imshow(X, map),title('map1');
map(64, :) = [1 0 0];
map(128, :) = [0 1 0];
map(256, :) = [0 0 1];
subplot(122),imshow(X, map),title('map2');
```

（5）处理 RGB 图像与索引图像的函数

图像类型转换	说明
[X, map] = dither(f)	采用抖动方法从 RGB 图像 f 创建索引图像
X = grayslice(f, n)	从灰度图像 f 通过阈值 n 处理创建索引图像
[X, map] = gray2ind(f)	从灰度图像 f 创建索引图像
f = ind2gray(X, map)	从索引图像创建灰度图像 f
[X, map] = rgb2ind(f)	从 RGB 图像 f 创建索引图像
f = ind2rgb(X, map)	从索引图像创建 RGB 图像 f
g = rgb2gray(f)	从 RGB 图像 f 创建灰度图像 g

（6）例子

```
f = imread('Fig0604(a).tif');
[X1, map1] = rgb2ind(f, 8, 'nodither');
%  8 表示处理后图像只有 8 种颜色
[X2, map2] = rgb2ind(f, 8, 'dither');
figure;
subplot(131),imshow(f),title('原图');
subplot(132),imshow(X1, map1),title('非抖动处理');
subplot(133),imshow(X2, map2),title('抖动处理');
g = rgb2gray(f);
g1 = dither(g);
figure;
subplot(121),imshow(g),title('灰度图像');
subplot(122),imshow(g1),title('抖动处理');
```

2．彩色图像空间转换

彩色空间	说明	用途	函数
NTSC	Y(亮度)、I(色调)、Q(饱和度)	模拟电视	rgb2ntsc()、ntsc2rgb()
YCbCr	Y(亮度)、Cb、Cr(蓝色、红色分量与参考值的差)	数字视频	rgb2ycbcr()、ycbcr2rgb()
HSV	H(色调)、S(饱和度)、V(亮度)	调色板	rgb2hsv()、hsv2rgb()
HSI	H(色调)、S(饱和度)、I(强度)	-	rgb2hsi()、hsi2rgb()
CMY	C(青色)、M(洋红)、Y(黄色)	彩色打印机、彩色复印机	imcompliment()
CMYK	C(青色)、M(洋红)、Y(黄色)、K(黑色)	彩色打印机、彩色复印机	-

（二）彩色图像处理

1．彩色图像空间滤波

（1）步骤

抽取三个分量 >> 每个分量分别滤波 >> 重建新图像

（2）分类

平滑处理、锐化处理

（3）例子

```
f = imread('Fig0622(a).tif');
[f, revertclass] = tofloat(f);
w = fspecial('average', 25);
% ======================================= rgb 图像平滑处理
f1 = imfilter(f, w, 'replicate'); % 作用相同于对分量分别滤波
f1 = revertclass(f1);
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(f1),title('平滑处理');
% ======================================= hsi 图像平滑处理
f_hsi = rgb2hsi(f);
H = f_hsi(:, :, 1);
S = f_hsi(:, :, 2);
I = f_hsi(:, :, 3);
H_filtered = imfilter(H, w, 'replicate');
S_filtered = imfilter(S, w, 'replicate');
I_filtered = imfilter(I, w, 'replicate');
f2 = cat(3, H, S, I_filtered);
f2 = revertclass(hsi2rgb(f2));
f3 = cat(3, H_filtered, S_filtered, I_filtered);
f3 = revertclass(hsi2rgb(f3));
subplot(223),imshow(f2),title('滤波 I 分量');
subplot(224),imshow(f3),title('滤波 HSI 分量');
```

2．彩色图像边缘检测

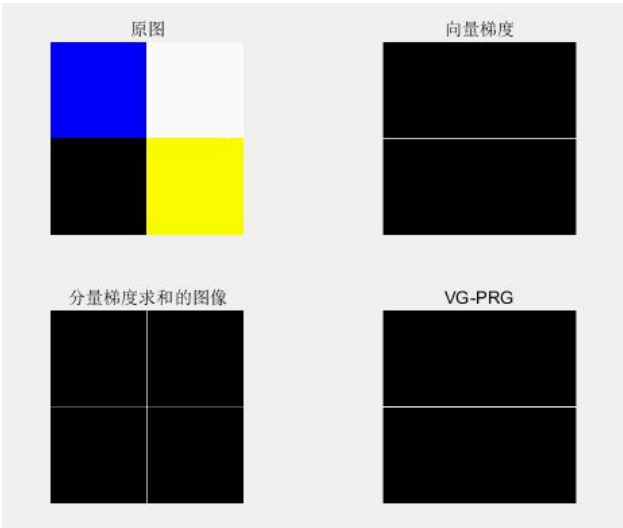
（1）函数

```
[VG, A, PRG] = colorgrad(f)
```

其中，VG 为图像的向量梯度，A 为梯度方向，PRG 为三个分量的梯度之和。

（2）例子

```
fr = imread('Fig0627(a).tif');
fg = imread('Fig0627(b).tif');
fb = imread('Fig0627(c).tif');
f = cat(3, fr, fg, fb);
f = tofloat(f);
[VG, A, PRG] = colorgrad(f);     % 计算梯度
err = abs(VG - PRG);             % 差值
figure;
subplot(221),imshow(f),title('原图');
subplot(222),imshow(VG),title('向量梯度');
subplot(223),imshow(PRG),title('分量梯度求和的图像');
subplot(224),imshow(err,[]),title('VG-PRG');
```



3．彩色图像图像分割

（1）函数

```
S = colorseg(method, f, T, parameters)
```

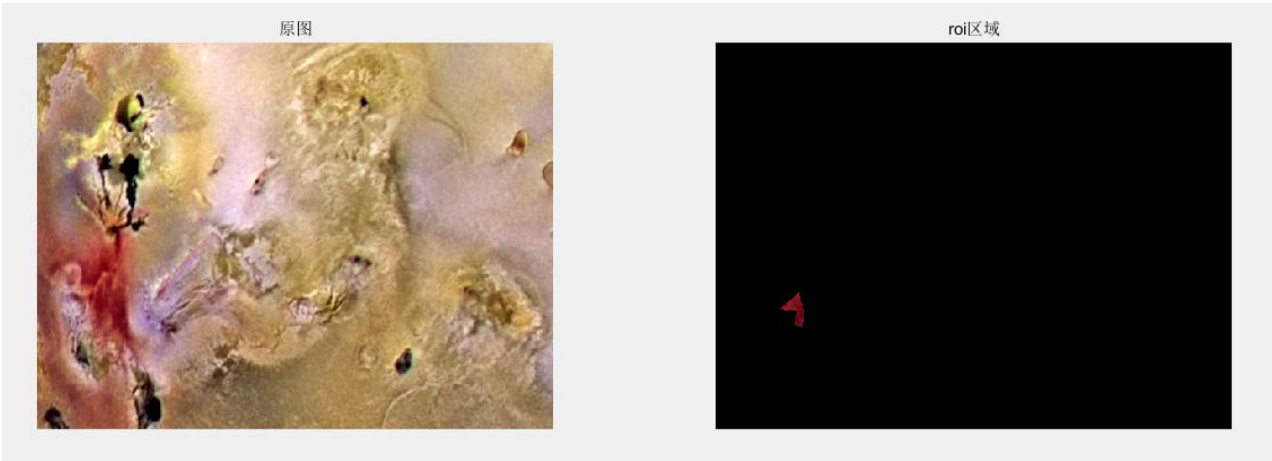
S = colorseg(‘euclidean’, f, T, m)	欧式距离算法
S = colorseg(‘mahalanobis’, f, T, m, C)	马氏距离算法

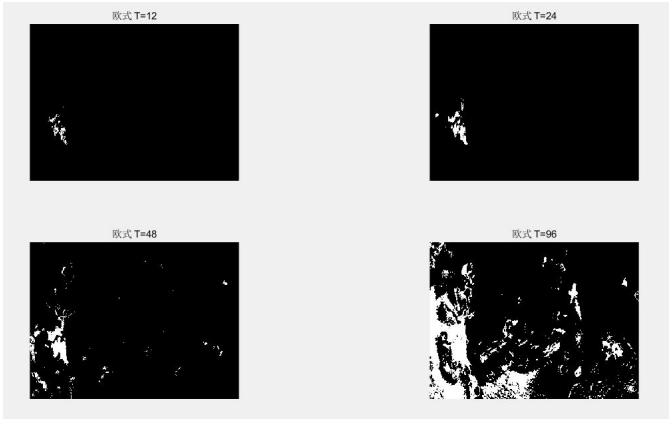
其中，f 为 rgb 图像，T 为阈值，m 为 roi 区域的均值，C 为 roi 区域的协方差矩阵，S 为逻辑图像。

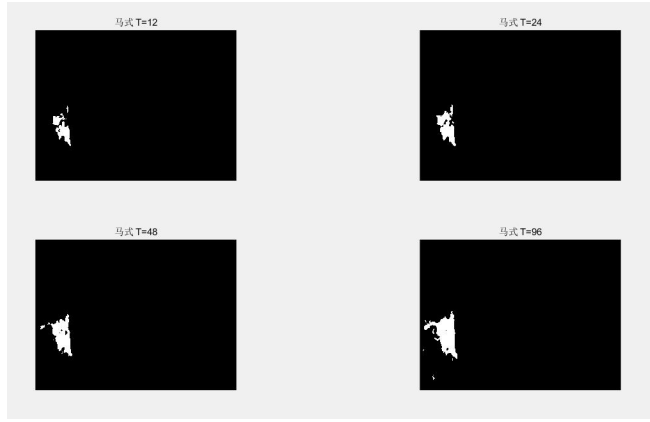
（2）例子

```
f = imread('Fig0630(a).tif');
mask = roipoly(f);           % 选取 roi 区域
fr = immultiply(mask, f(:,:,1));
fg = immultiply(mask, f(:,:,2));
fb = immultiply(mask, f(:,:,3));
g = cat(3, fr, fg, fb);
```

```
[M, N, K] = size(g);
I = reshape(g, M*N, K);      % K=3
idx = find(mask);
I = double(I(idx, :));
[C, m] = covmatrix(I);       % 计算协方差矩阵 C 和均值 m
d = diag(C);                 % 方差 d 为位于协方差矩阵 C 的对角线上
sd = sqrt(d);                % 标准差 sd
T = max(ceil(sd));           % 阈值为标准差最大值的向上取整
figure;
subplot(121),imshow(f),title('原图');
subplot(122),imshow(g),title('roi 区域');
% ================================ 欧式距离算法
T12 = colorseg('euclidean', f, 1*T, m);
T24 = colorseg('euclidean', f, 2*T, m);
T48 = colorseg('euclidean', f, 4*T, m);
T96 = colorseg('euclidean', f, 8*T, m);
figure;
subplot(221),imshow(T12),title('欧式 T=12');
subplot(222),imshow(T24),title('欧式 T=24');
subplot(223),imshow(T48),title('欧式 T=48');
subplot(224),imshow(T96),title('欧式 T=96');
% ================================ 马式距离算法
T12 = colorseg('mahalanobis', f, 1*T, m, C);
T24 = colorseg('mahalanobis', f, 2*T, m, C);
T48 = colorseg('mahalanobis', f, 4*T, m, C);
T96 = colorseg('mahalanobis', f, 8*T, m, C);
figure;
subplot(221),imshow(T12),title('马式 T=12');
subplot(222),imshow(T24),title('马式 T=24');
subplot(223),imshow(T48),title('马式 T=48');
subplot(224),imshow(T96),title('马式 T=96');
```




欧式 T=12
欧式 T=24
欧式 T=48
欧式 T=96


马式 T=12
马式 T=24
马式 T=48
马式 T=96