第 15 讲　MATLAB 数字图像处理

司守奎
烟台市，海军航空大学
Email：sishoukui@163.com

15.1　数字图像处理简介

数字图像处理（Digital Image Processing）又称为计算机图像处理，是一种将图像信号数字化后利用计算机进行处理的过程。

15.1.1　数字图像处理的研究内容

数字图像处理的研究内容主要有以下方向。

1. 图像运算与变换

图像的运算主要以图像的像素为运算对象，对两幅或多幅图像进行点运算、代数运算及逻辑运算，但逻辑运算中逻辑非的运算对象是单幅图像；图像的变换主要是对图像像素空间关系的改变，从而改变图像的空间结构。

2. 图像增强

图像增强是为了提高图像的质量，图像增强的方法有灰度变换、直方图修正、图像平滑和图像锐化等。

3. 图像复原

图像复原，即利用退化过程的先验知识，去恢复已被退化图像的本来面目。图像复原也是为了提高图像的质量，当图像品质下降的原因已知时，图像复原可以对图像进行校正。图像复原的关键是根据图像品质下降过程建立一个合理的降质模型，然后再采用某种滤波方法，恢复或重建原来的图像。

4. 图像的锐化处理及边缘检测

图像的锐化和边缘检测就是补偿图像的轮廓，增强图像的边缘及灰度跳变的部分，使图像变得清晰，处理方法分为空间域处理和频域处理两类。

5. 图像分割

图像分割是将图像分成区域，将感兴趣的部分提取出来，为进一步进行图像识别、分析和理解提供方便。虽然目前已研究出不少边缘提取、区域分割的方法，但还没有一种普遍适用于各种图像的有效方法。现有的图像分割方法主要分为基于阈值的分割方法、基于区域的分割方法、基于边缘的分割方法，以及基于特定理论的分割方法等。

6. 图像编码压缩

图像编码压缩是指在不影响图像质量的情况下，减少图像的数据量，以便节省图像传输、处理时间和减少所占用的存储器容量。图像压缩编码可分为两类，一类压缩是可逆的，即压缩后的数据可以完全恢复原来的图像，信息没有损失，称为无损压缩编码；另一类压缩是不可逆的，即从压缩后的数据无法完全恢复原来的图像，信息有一定损失，称为有损压缩编码。编码是压缩技术中最重要的方法，它在图像处理技术中是发展最早且比较成熟的技术。

15.1.2　图像的表示方法

下面简单介绍 5 种图像表示方法，分别是二进制图像、灰度图像、RGB 图像、索引图像和多帧图像。

1. 二进制图像

二进制图像也称为二值图像，通常用一个二维数组来描述，1 位表示一个像素，组成图像的像素值非 0 即 1，没有中间值，通常 0 表示黑色，1 表示白色。二进制图像一般用来描述文字或者图形，其优点是占用空间少，缺点是当表示人物或风景图像时只能描述轮廓。

2. 灰度图像

灰度图像也称为单色图像，通常也由一个二维数组表示一幅图像，8 位表示一个像素，0 表示黑色，255 表示白色，1~254 表示不同的深浅灰色。

在 MATLAB 中，灰度图像可以用不同的数据类型来表示，如 8 位无符号整数、16 位无符号整数或双精度类型。无符号整型表示的灰度图像每个像素在[0,255]或[0,65535]范围内取

值；双精度类型表示的灰度图像，每个像素在[0,1.0]范围内取值。

3. RGB 图像

RGB 图像也称为真彩色，是一种彩色图像的表示方法，利用 3 个大小相同的二维数组表示一个像素，3 个数组分别代表 R、G、B 这 3 个分量，R 表示红色，G 表示绿色，B 表示蓝色，通过 3 种基本颜色可以合成任意颜色。

在 MATLAB 中，RGB 图像存储为一个 $M \times N \times 3$ 的多维数据矩阵，其中元素可以为 8 位无符号数、16 位无符号数和双精度数。

4. 索引图像

在 MATLAB 中，索引图像包含一个数据矩阵 X 和一个颜色映射（调色板）矩阵 map。数据矩阵可以是 8 位无符号整型、16 位无符号整型或双精度类型。颜色映射矩阵 map 是一个 $m \times 3$ 的数据阵列，其中每个元素的值均为[0,1]之间的双精度浮点型数据，map 矩阵中的每一行分别表示红色、绿色和蓝色的颜色值。索引图像可把像素的值直接映射为调色板数值，每个像素的颜色通过使用 X 的值作为 map 的下标来获得。

5. 多帧图像

多帧图像是一种包含多幅图像或帧的图像文件，又称为多页图像或图像序列，主要用于需要对时间或场景上相关图像集合进行操作的场合。例如，计算机 X 线断层扫描图像或电影帧等。

在 MATLAB 中，用一个四维数组表示多帧图像，其中第四维用来指定帧的序号。图像处理工具箱支持在同一个数组中存储多幅图像，每一幅图像称为一帧。如果一个数组中包含多帧，那么这些图像的第四维是相互关联的。

15.2 MATLAB 图像处理基础

15.2.1 图像类型的转换

在许多图像处理过程中，常常需要进行图像类型转换，否则对应的操作没有意义甚至出错。

1. RGB 图像转换为灰度图像

在 MATLAB 中，将 RGB 图像转换为灰度图像，需要调用函数 rgb2gray，其调用格式如下。

X=rgb2gray(I)：该函数是将 RGB 图像 I 转换为灰度图像 X。

newmap=rgb2gray(map)：该函数是将彩色颜色映射表 map 转化成灰度颜色映射表 newmap。

例 15.1 将 RGB 图像转换为灰度图像。

```
I=imread('board.tif');  %MATLAB 工具箱中的图像文件
J=rgb2gray(I);
subplot(121), imshow(I), title('彩色电路板图像')
subplot(122), imshow(J), title('灰度电路板图像')
D1=size(I), D2=size(J)  %显示图像矩阵的维数
```

例 15.2 将彩色颜色映射表转化成灰度颜色映射表。

```
[X,map]=imread('trees.tif'); %MATLAB 工具箱中的图像文件
gmap=rgb2gray(map);
figure, imshow(X,map), figure, imshow(X,gmap)
%用颜色映射表时，彩色图片和灰度图片不能在一个图像界面上显示
figure, subplot(121), imshow(X,map), subplot(122), imshow(X,gmap)
D1=size(X), D2=size(map), D3=size(gmap)
```

2. RGB 图像转换为索引图像

在 MATLAB 中，将真彩色图像转换成为索引图像直接调用函数 rgb2ind。函数 rgb2ind 具体调用格式如下。

[X,map]=rgb2ind(I,tol)：该函数是利用均匀量化的方法将 RGB 图像转换为索引图像。其中，I 就是原 RGB 图像，tol 的范围是从 0 至 1，[X,map]对应生成的索引图像，map 包含至少

$(\text{floor}(1/\text{tol})+1)^3$个颜色。

[X,map]=rgb2ind(I,N)：该函数是利用最小方差量化的方法，将 RGB 图像转换索引图像。其中，I 就是原 RGB 图像，N 为 map 中包含的颜色数。

X=rgb2ind(I,map)：该函数是利用颜色映射表 map 的逆算法，将 RGB 图像转换为索引图像。

例 15.3　将 RGB 图像转换为索引图像。

```
RGB = imread('ngc6543a.jpg'); %MATLAB 工具箱中的图像文件
subplot(131), imagesc(RGB), zoom(4)　%图像放大 4 倍
[IND,map]=rgb2ind(RGB,32);
subplot(132), image(IND), colormap(map), zoom(4)
subplot(133), imshow(RGB) %不带坐标轴刻度的显示，图像不放大
```

例 15.4　将 RGB 图像转换为索引图像。

```
clc, clear
a=imread('football.jpg'); %MATLAB 工具箱中的图像文件
[x1,m1]=rgb2ind(a,128); %将 RGB 图像转换成索引图像，颜色种数 N 至多 128 种
[x2,m2]=rgb2ind(a,0.1); %将 RGB 图像转换成索引图像，颜色种数 N 至多 11^3 种
m3=colorcube(128); %创建一个指定颜色数目的 RGB 颜色映射表
x3=rgb2ind(a,m3);
subplot(131), imshow(x1,m1) %显示用最小方差法转换后的索引图像
subplot(132), imshow(x2,m2) %显示用均匀量化法转换后的索引图像
subplot(133), imshow(x3,m3) %显示用颜色近似法转换后的索引图像
```

3. 索引图像转换为 RGB 图像

在 MATLAB 中，利用函数 ind2rgb 函数可以将索引图像转换为 RGB 图像。其调用格式为

RGB=ind2rgb(X,map)：其中[X,map]指向索引图像，RGB 指向转换后的真彩色图像。

例 15.5　将索引图像转换为真彩色图像。

```
clc, clear
[x,a]=imread('kids.tif'); %MATLAB 工具箱中的图像文件
b=ind2rgb(x,a); %将索引图像转换为真彩色图像
subplot(121), imshow(x,a), subplot(122), imshow(b)
```

15.2.2　MATLAB 图像工具箱中的几个函数介绍

1.imread 和 imwrite

imread 函数是从图像文件读图像，它的一般调用格式为

A=imread(filename)

返回值 A 为矩阵，当图像是黑白和灰度图像时，A 为二维矩阵，当图像是彩色图像时，A 是三维矩阵，即 A 为 3 个二维矩阵，分别为 R、G、B 的像素值。

imwrite 函数是把图像写到图像文件中，它的一般调用格式为

imwrite(A,filename)

把图像 A（二维矩阵或三维矩阵）写到图像文件 filename 中。

例 15.6　读入一个 bmp 图像，然后再把图像保存成 jpg 格式。

```
clc, clear
a=imread('data6.bmp');　%非工具箱图像文件
imwrite(a,'data7.jpg')　%把图像保存成 jpg 格式
subplot(121), imshow(a) %显示原图像
subplot(122), imshow('data7.jpg')　%显示 jpg 图像
```

2. 文件信息读取函数 imfinfo

在 MATLAB 中，对图像进行操作和处理时，经常需要知道图像文件的文件名、文件格式、图像大小、图像类型和数据类型等信息，可以直接调用 MATLAB 函数 imfinfo 来读取图像文

件的信息。其调用格式如下。

info=imfinfo(filename)：该函数读取文件 filename 的信息。其中，filename 指的是图像文件的“文件名”（包括后缀名）。info 是一个结构数组。不同格式的文件最终得到的 info 所包含的结构成员不同，但基本都包含前 9 个结构成员，具体如表 15.1 所示。

表 15.1　函数 imfinfo 返回的结构数组基本内容

结构数组成员名	所代表函数
Filename	文件名称
FileMoDate	文件最后修改日期和时间
FileSize	文件大小（单位是字节）
Format	文件格式或扩展名（tif、jpf 和 png 等）
FormatVersion	文件格式版本号
Width	图像文件的宽度，单位为像素
Height	图像文件的高度，单位为像素
BitDepth	图像文件中每一个像素所占位宽（真彩色图像每个像素占 24 位）
ColorType	图像类型（grayscale-灰度图像，truecolor-RGB 图像，indexed-索引图像）

3. 函数 imtool

利用函数 imtool 可以将图像在图像工具浏览器中显示，其实现方式见下面的例子。

例 15.7　imtool 示例。

```
fig=imread('moon.tif'); imtool(fig)
```

4. 函数 imshow，image 和 imagesc

常用的显示图像函数除了函数 imtool 和 imshow 以外，还有函数 image 和 imagesc。这两个函数的功能基本与前者相近，可以显示一幅图像，自动设置图像的一些属性。这些自动设置的属性包括图像对象的 CData 属性、CDataMapping 属性和坐标轴对象的属性等。

例 15.8　利用函数 imshow、image 和 imagesc 显示图像进行比较。

```
clc, clear
a=imread('data6.bmp');  %非工具箱图像文件
subplot(221), imshow(a) %不带坐标轴刻度
subplot(222), image(a)  %带坐标轴刻度
subplot(223), image([100,500],[100,400],a) %显示调整坐标后的图像
subplot(224), imagesc(a,[60,80]) %显示经过拉伸后的图像
```

5. 函数 montage

在 MATLAB 中，要同时显示多帧图像序列，需要调用函数 montage，其调用格式如下。

montage(I)：该函数是显示多帧灰度图像、二值图像或者 RGB 图像。其中 I 表示图像序列数组，如果显示的是灰度图像或者二值图像，则 I 是 $M\times N\times 1\times K$ 的数组，如果显示的是 RGB 图像，则 I 是 $M\times N\times 3\times K$ 的数组。

montage(X,map)：该函数是显示多帧索引图像。其中 X 是一个 $M\times N\times 1\times K$ 的数组，所有的索引图像的颜色值都用颜色映射表 map 中的颜色值。

例 15.9　利用函数 montage 同时显示多帧图像序列。

```
clc, clear
I=zeros(128,128,1,27); %建立四维数组
for i=1:27
    [I(:,:,:,i),map]=imread('mri.tif',i);  %读取多帧图像序列，存放在数组I中
end
montage(I,map)
```

6. 像素信息的显示函数 impixel

MATLAB 图像处理工具箱中的函数 impixel 可以返回用户指定的图像像素值。函数 impixel 可以返回选中像素或像素集的数据值。用户可以直接将像素坐标作为该函数的输入参数或者

用鼠标选中像素。其调用格式如下。

（1）用鼠标选中像素

P=impixel(I)：该函数是显示图像指定的像素值。

在图像显示出来后，单击图像选择像素，按下【Backspace】键或者【Delete】键可以移除之前选择的像素点，双击或者右击，表示你选择的是最后一个像素点结束选择，或者按下回车，最后将在命令窗口显示所选择点的像素值。

（2）将像素坐标作为函数输入参数，具体调用格式如下：

P=impixel(I,C,R)：该函数是输出图像中指定的像素值，其中 I 表示图像，R 和 C 是长度相同的向量，R 对应横坐标，C 对应纵坐标。

例 15.10 利用函数 impixel 显示图像指定像素信息。

```
clc, clear
a=imread('peppers.png');
c=[12 146 410];   %存放像素纵坐标
r=[104 156 129]; %存放像素横坐标
pixels1=impixel(a)   %交互式用鼠标选择像素
pixels2=impixel(a,c,r)   %显示特征坐标的像素值
```

15.3 数字图像的运算

15.3.1 图像代数运算

图像加法运算的一个重要应用是通过同一幅图像叠加取平均，消除原图像中的附加噪声，其基本原理：对于原图像 $f(x,y)$，有一个噪声图像集$\{g_i(x,y)\}$，$i=1,2,\cdots,M$，其中

$$\underbrace{g_i(x,y)}_{\text{混入噪声的图像}}=\underbrace{f(x,y)}_{\text{原始图像}}+\underbrace{e_i(x,y)}_{\text{随机噪声}}.$$

M 个图像的均值为：

$$\bar{g}(x,y)=\frac{1}{M}\sum_{i=1}^{M}[f(x,y)+e_i(x,y)]=f(x,y)+\frac{1}{M}\sum_{i=1}^{M}e_i(x,y).$$

当噪声 $e_i(x,y)$ 为互不相关，且均值为 0 时，上述图像均值将降低图像的噪声。

在 MATLAB 图像处理工具箱中提供了函数 imnoise 实现在图像中加入噪声，其具体的调用格式如下。

J=imnoise(I,type,parameters)：该函数是对图像 I 添加典型噪声后生成的有噪图像，结果返回给 J。其中 I 为原始图像；type 为添加的噪声类型，取值可以是高斯噪声 gaussian，零均值的高斯噪声 localvar、poisson、椒盐噪声 salt & pepper 和乘性噪声 speckle；Parameter 为不同类型的噪声参数。

例 15.11 使用加法运算消除一幅图像的附加噪声。

```
clc, clear
a=imread('eight.tif'); %MATLAB工具箱中的图像文件
b=imnoise(a,'gaussian',0,0.05); %加入高斯白噪声
c=b; n=3;
c=im2double(c); a=im2double(a); %将数据类型转换成双精度
for i=1:n
    c=imadd(c,a); %将原图像与带噪声图像进行多次叠加
end
ave=c/(n+1); %求叠加的平均图像
subplot(121), imshow(b)   %显示加入椒盐噪声后的图像
subplot(122), imshow(ave) %显示叠加平均后的图像
```

注 15.1　两幅图像的像素值代数运算时产生的结果很可能超过图像数据类型所支持的最大值，尤其对于 uint8 类型的图像，溢出情况最为常见。当数据值发生溢出时，MATLAB 将数据截取为数据类型所支持的最大值，这种截取效果称之为饱和。为了避免出现饱和现象，在进行代数计算前最好将图像转换为一种数据范围较宽的数据类型。例如，在加法操作前将 uint8 图像转换为 double 类型。

15.3.2　图像的剪切

在进行图像处理的过程中，有时用户只对采集的图像部分区域感兴趣，这时候就需要对原始图像进行剪切。在 MATLAB 图像处理工具箱中提供了函数 imcrop 进行图像的剪切操作，其具体的调用格式如下。

I2=imcrop(I,rect)：该函数是按照四元素数组 rect 剪切图像 I，rect 的具体形式为[xmin,ymin,width,height]说明剪切矩形区域大小。

[I2,rect]=imcrop(I)：该函数是执行后首先显示原图像，然后利用鼠标选择剪切区域，并将剪切区域图像返回给 I2，将剪切区域的范围大小返回给 rect。

[X,Y,I2,rect]=imcrop(I)：该函数是执行后首先显示原图像，然后利用鼠标选择剪切区域，返回当前剪切区域图像的像素点 x 和 y 坐标给 X 和 Y，并将剪切区域的范围大小返回给 rect。

例 15.12　利用函数 imcrop，通过指令方式实现图像的剪切操作。

```
clc, clear
a=imread('peppers.png'); %MATLAB 工具箱中的图像文件
rect=[20,200,200,100]; %定义剪切区域
b=imcrop(a,rect);  %定义剪切区域
subplot(121), imshow(a);
rectangle('Position',rect,'LineWidth',2,'EdgeColor','r'); %将图像的剪切区域标出
subplot(122), imshow(b)
```

例 15.13　利用函数 imcrop，通过鼠标操作实现图像的剪切操作。

```
clc, clear
a=imread('peppers.png');  %MATLAB工具箱中的图像文件
[b,rect]=imcrop(a);  %定义剪切区域
subplot(121), imshow(a);
rectangle('Position',rect,'LineWidth',2,'EdgeColor','r'); %将图像的剪切区域标出
subplot(122), imshow(b)
```

程序执行时，首先显示原图像 peppers，当鼠标移至图像区域后变成“+”，用户按住鼠标左键选择剪切区域，再在选择的剪切区域内双击鼠标确定剪切区域，或者在选择的剪切区域内右击鼠标，弹出剪切菜单，选择 Crop Image 选项，然后单击，确定剪切区域。

15.3.3　图像的空间变换

在MATLAB的图像处理工具箱中提供了一个专门的函数imwarp，用户可以定义参数实现多种类型的空间变换，包括仿射变换（如平移、缩放、旋转、剪切）、投影变换等。函数imwarp具体的调用格式如下。

B=imwarp(A,tform)：该函数中A是待变换的图像矩阵；tform表示为执行空间变换的所有参数的的结构体；B为按照tform参数变换后的图像矩阵。

在MATLAB中利用函数imwarp实现图像的空间变换时，都需要先定义空间变换的参数。对于空间变换参数的定义，MATLAB也提供了相应的函数affine2d，affined3d，fitgeotrans等函数，它们的作用是创建进行空间变换的参数结构体。affine2d的具体调用方式如下。

tform=affine2d(C)：该函数返回一个 N 维的仿射变换参数结构体tform，输入参数C是一个 $(N+1)*(N+1)$ 的矩阵。

用户结合使用函数affine2d和函数imwarp，就可以灵活实现图像的线性变换，而变换的

结果和变换参数结构体密切相关。以二维仿射变换为例，原图像 $f(x,y)$ 和变换后图像 $g(x',y')$，仿射变换中原图像中某个像素点坐标 (x,y) 和变换后该像素点坐标 (x',y') 满足关系式

$$(x',y')=T(x,y),$$

具体数学表达式为

$$x'=a_0x+a_1y+a_2,$$
$$y'=b_0x+b_1y+b_2,$$

写成矩阵形式

$$\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}a_0&a_1&a_2\\b_0&b_1&b_2\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}.$$

例15.14　利用函数imwarp，实现图像的平移、缩放、旋转和剪切。

```
clc, clear
a=imread('peppers.png'); %MATLAB工具箱的图像文件
tf1 = affine2d([cosd(30),-sind(30),0;sind(30),cosd(30),0;0 0 1]); %创建旋转参数结构体
ta1=imwarp(a,tf1);
tf2=affine2d([5 0 0;0 10.5 0;0 0 1]);%创建缩放参数结构体
ta2=imwarp(a,tf2); %实现图像缩放
subplot(121), imshow(ta1), subplot(122),imshow(ta2),axis on
```

15.3.4　图像的邻域和块操作

1. 图像的邻域操作

例 15.15　利用函数 blockproc 实现图像的分离块操作。

```
clc, clear
a=imread('peppers.png');
fun1=@(x) imrotate(x.data,30); %获取分离块操作的函数句柄
a1=blockproc(a,[64,64],fun1); %进行分离块操作
fun2=@(x)x.data(:,:,[3 1 2]); %获取分离块操作的函数句柄
a2=blockproc(a,[50,50],fun2);
blockproc(a,[100,100],fun2,'Destination','a3.tif') %把处理结果写到图像文件中
subplot(131), imshow(a1), subplot(132), imshow(a2),
subplot(133), imshow('a3.tif')
```

2. 图像的区域选取

例 15.16　实现图像的区域选取和操作。

```
clc, clear
a=imread('pout.tif');
a1=roicolor(a,55,100); %根据灰度范围 55~100，选取灰度区域
c=[87 171 201 165 79 32 100]; %定义多边形区域顶点横坐标
r=[133 133 205 220 259 209 133]; %定义顶点纵坐标
a2=roipoly(a,c,r); %选择区域
a3=roifill(a,a2); %根据生成 a2 掩膜图像进行区域填充
h=fspecial('motion',20,45); %创建 motion 滤波器
a4=roifilt2(h,a,a2); %进行区域滤波
subplot(221), imshow(a1), subplot(222), imshow(a2), axis on
subplot(223), imshow(a3), subplot(224), imshow(a4)
```

15.4　图像增强技术

图像增强的目的是为了改善图像的视觉效果，使图像更清晰。图像增强按作用域可分为空域内处理和频域内处理，空域内处理是直接对图像进行处理；频域内处理是在图像的某个变换域内，对图像的变换系数进行运算，然后通过逆变换获得图像增强效果。

空间域内的图像增强技术主要有灰度变换方法和直方图方法等。此外，还可以对图像进行滤波，主要包括线性滤波和非线性滤波，其中非线性滤波又包括中值滤波、顺序统计滤波和自适应滤波等。

通过傅里叶变换可以将图像从空间域转换到频域，在频域进行滤波，频域滤波主要包括低频滤波、高频滤波、带阻滤波器和同态滤波等，然后通过傅里叶反变换转换到空间域。

15.4.1 空域滤波

空域滤波是空域图像增强的常用方法。空域滤波是对图像中每个像素为中心的邻域进行一系列的运算，然后将得到的结果代替原来的像素值。

1.线性空域滤波

线性平均滤波是一种最常用的线性空域滤波。线性平均滤波实际是一种低通滤波，信号的低频部分通过，阻止高频部分通过。由于图像的边缘处于高频部分，因此线性平均滤波后，会造成图像边缘的模糊。

在进行线性平均滤波时，常用的模板大小为3×3，如下所示。

$$T=\frac{1}{5}\begin{bmatrix}0 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 0\end{bmatrix},$$

对应的函数表达式为

$$f(x,y)=\frac{1}{5}[f(x,y-1)+f(x-1,y)+f(x,y)+f(x+1,y)+f(x,y+1)].$$

例 15.17　通过函数 imfilter 对图像进行平滑。

```
clc, clear
a=imread('coins.png');   %MATLAB工具箱图像文件
b=imnoise(a, 'salt & pepper', 0.02); %添加噪声
h=ones(3)/5; h([1 3],[1 3])=0; %建立模板
c=imfilter(b, h); %进行滤波
subplot(131); imshow(a); %显示原始图像
subplot(132); imshow(b); %显示添加噪声的图像
subplot(133); imshow(c); %显示滤波后的图像
```

2.非线性空域滤波

中值滤波是一种常用的非线性平滑滤波器，其基本原理是把数字图像或数字序列中一点的值用该点的一个邻域中各点值的中值来替换，其主要功能是让周围像素灰度值差别比较大的像素改取与周围的像素值接近的值，从而可以消除孤立的噪声点，所以中值滤波对于滤除图像的椒盐噪声非常有效。中值滤波在消除噪声的同时还具有保护边界信息的优点，对图像中的某些细节起到保护作用，因而在图像去噪处理中得到了比较广泛的应用。

例 15.18　通过函数 medfilt2 对图像进行中值滤波。

```
clc, clear
a=imread('coins.png');
b=imnoise(a,'salt & pepper',0.03); %添加噪声
c=medfilt2(b); %中值滤波
subplot(131);imshow(a); title('原始图像')
subplot(132); imshow(b); title('添加椒盐噪声图像')
subplot(133); imshow(c); title('中值滤波后的图像')
```

在 MATLAB 中，采用函数 ordfilt2 进行排序滤波。函数 ordfilt2 进行滤波时，可以通过模板来选择排序后的某个值作为输出。当函数 ordfilt2 的调用形式为

J= ordfilt2(I,median(1:m*n),[m,n])

时，相当于中值滤波。

例 15.19　通过函数 ordfilt2 对图像进行排序滤波。

```
clc, clear
a=imread('coins.png');
b=ordfilt2(a,1,true(5)); %取最小值的排序滤波
c=ordfilt2(a,25,true(5)); %取最大值的排序滤波
subplot(131);imshow(a); title('原始图像')
subplot(132); imshow(b); title('最小值滤波图像')
subplot(133); imshow(c); title('最大值滤波图像')
```

在 MATLAB 中，函数 wiener2 根据图像的噪声进行自适应滤波。该函数根据图像的局部方差来调整滤波器的输出。当局部方差大时，滤波器的平滑效果较弱；当局部方差小时，滤波器的平滑效果强。

例 15.20　通过函数 wiener2 对图像进行自适应滤波。

```
clc, clear
a=imread('coins.png');
b=imnoise(a,'gaussian',0,0.01); %添加噪声
c=wiener2(b,[5,5]); %自适应滤波
subplot(131);imshow(a); title('原始图像')
subplot(132); imshow(b); title('添加高斯噪声图像')
subplot(133); imshow(c); title('自适应滤波后的图像')
```

对于模糊的图像，通过锐化滤波器能够补偿图像的轮廓，让图像变得清晰。锐化滤波器常用拉普拉斯算子。拉普拉斯算子比较适合用于改善因为光线的漫反射造成的图像模糊。离散函数的拉普拉斯算子公式为

$$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j).$$

对应的滤波模板为

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

使用拉普拉斯滤波器的公式为

$$g(x,y) = f(x,y) - \nabla^2 f(x,y).$$

例 15.21　通过拉普拉斯算子对图像进行锐化滤波。

```
clc, clear
a=imread('rice.png'); %MATLAB工具箱图像文件
h=fspecial('laplacian',0)　%定义模板
b=a-imfilter(a,h);
subplot(121);imshow(a); title('原始图像')
subplot(122); imshow(b); title('锐化滤波图像')
```

15.4.2　频域滤波

频率域图像增强首先通过傅里叶变换将图像从空间域转换为频率域，然后在频率域内对图像进行处理，最后通过傅里叶反变换转换到空间域。频率域内的图像增强通常包括低通

滤波、高通滤波等。

设 $f(x,y)$ 为原始图像函数， $h(x,y)$ 为滤波器脉冲响应函数，则空域内的滤波是基于卷积运算的，如下所示。

$$g(x,y)=f(x,y)*h(x,y)\text{，}$$

其中 $h(x,y)$ 可以是低通或高通滤波， $g(x,y)$ 为空域滤波的输出图像函数。根据卷积定理，上式的傅里叶变换如下

$$G(u,v)=F(u,v)H(u,v)\text{，}$$

其中 $G(u,v)$ 、$F(u,v)$ 和 $H(u,v)$ 分别是 $g(x,y)$ 、$f(x,y)$ 和 $h(x,y)$ 的傅里叶变换。$H(u,v)$ 为滤波系统的传递函数，根据具体的要求进行设计，再与 $F(u,v)$ 相乘，即可获得频谱改善的 $G(u,v)$，从而实现低通或高通等滤波。最后求 $G(u,v)$ 的傅里叶反变换，可以获得滤波后的图像 $g(x,y)$ 。频域滤波的关键是 $G(u,v)$ 的设计。

1.低通滤波

低通滤波器的功能是让低频率通过而滤掉或衰减高频，其作用是过滤掉包含在高频中的噪声。所以低频滤波的效果是图像去噪声平滑增强，但同时也抑制了图像的边界，造成图像不同程度上的模糊。对于大小为 $M\times N$ 的图像，频率点 (u,v) 与频域中心的距离为 $D(u,v)$，其表达式为

$$D(u,v)=[(u-M/2)^2+(v-N/2)^2]^{1/2}.$$

（1）理想低通滤波器

理想低通滤波器的产生公式为

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

其中 D_0 为理想低通滤波器的截止频率。理想低通滤波器在半径为 D_0 的范围内，所有频率都可以没有衰减地通过滤波器，该半径之外的所有频率都完全被衰减掉。理想低通滤波器具有平滑图像的作用。

（2）巴特沃斯低通滤波器

巴特沃斯低通滤波器的产生公式为

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

其中 D_0 为巴特沃斯低通滤波器的截止频率，参数 n 为巴特沃斯滤波器的阶数， n 越大则滤波器的形状越陡峭。

（3）高斯低通滤波

高斯低通滤波的产生公式为

$$H(u,v)=e^{-\frac{D^2(u,v)}{2D_0^2}}\text{，}$$

其中 D_0 为高斯低通滤波器的截止频率。

例 15.22　利用理想低通滤波器对图像进行滤波。

```
clc, clear
I=imread('coins.png');
[m,n]=size(I); %滤波器的行数和列数
u=-n/2:(n/2-1); v=-m/2:(m/2-1);
[u,v]=meshgrid(u,v);
D=sqrt(u.^2+v.^2); D0=80; %截止频率
H=(D<=D0); %理想低通滤波器
J=fftshift(fft2(I)); %时域图像转换到频域，并进行中心变换
K=J.*H; %滤波处理
L=ifft2(ifftshift(K)); %傅里叶逆变换
L=uint8(L); %必须转换为原来的数据格式，否则显示结果是不正确的
```

```
subplot(121); imshow(I); %显示原始图像
subplot(122); imshow(L); %显示滤波后的图像
```

例 15.23　利用巴特沃斯低通滤波器对图像进行滤波。

```
clc, clear
I=imread('liftingbody.png');I=im2double(I);%读入工具箱中的图像,并转换数据类型
[m,n]=size(I); %滤波的行数和列数
u=-n/2:(n/2-1); v=-m/2:(m/2-1);
[u,v]=meshgrid(u, v);
D=sqrt(u.^2+v.^2); D0=50; n=6;
H=1./(1+(D./D0).^(2*n)); %设计巴特沃斯滤波器
J=fftshift(fft2(I)); %时域图像转换到频域，并进行中心变换
K=J.*H; %滤波处理
L=ifft2(ifftshift(K)); %数据类型与 I 的类型相同，不需再转换为 uint8 格式
subplot(121); imshow(I); %显示原始图像
subplot(122); imshow(L); %显示滤波后的图像
```

2.高通滤波

衰减或抑制低频分量，让高频分量通过称为高通滤波，其作用是使图像得到锐化处理，突出图像的边界。经理想高频滤波后的图像把信息丰富的低频去掉了，丢失了许多必要的信息。一般情况下，高通滤波对噪声没有任何抑制作用，若简单的使用高通滤波，图像质量可能由于噪声严重而难以达到满意的改善效果。为了既加强图像的细节又抑制噪声，可采用高频加强滤波。这种滤波实际上是由一个高通滤波器和一个全通滤波器构成的，这样便能在高通滤波的基础上保留低频信息。

（1）理想高通滤波器

理想高通滤波器的产生公式为

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

其中 D_0 为理想高通滤波器的截止频率。

（2）巴特沃斯高通滤波器

巴特沃斯高通滤波器的产生公式为

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

其中 D_0 为巴特沃斯低通滤波器的截止频率，参数 n 为巴特沃斯滤波器的阶数，用来控制滤波器的陡峭程度。

（3）高斯高通滤波

高斯高通滤波的产生公式为

$$H(u,v)=1-e^{-\frac{D^2(u,v)}{2D_0^2}},$$

其中 D_0 为高斯高通滤波器的截止频率。

例 15.24　利用巴特沃斯高通滤波器对图像进行滤波。

```
clc, clear
I=imread('rice.png'); I=im2double(I); %读入图像，并转换数据类型
[m,n]=size(I); %滤波的行数和列数
u=-n/2:(n/2-1); v=-m/2:(m/2-1);
[u,v]=meshgrid(u, v);
D=sqrt(u.^2+v.^2); D0=30; n=6;
H=1./(1+(D0./D).^(2*n)); %设计巴特沃斯滤波器
```

```
J=fftshift(fft2(I)); %时域图像转换到频域，并进行中心变换
K=J.*H; %滤波处理
L=ifft2(ifftshift(K)); %傅里叶逆变换
subplot(121); imshow(I); %显示原始图像
subplot(122); imshow(L); %显示滤波后的图像
```

例15.25　利用高斯高通滤波器对图像进行滤波。

```
clc, clear
I=imread('coins.png'); I=im2double(I); %读入图像，并转换数据类型
[m,n]=size(I); %滤波的行数和列数
u=-n/2:(n/2-1); v=-m/2:(m/2-1);
[u,v]=meshgrid(u, v);
D=sqrt(u.^2+v.^2); D0=20;
H=1-exp(-D.^2/2/D0); %设计高斯高通滤波器
J=fftshift(fft2(I)); %时域图像转换到频域，并进行中心变换
K=J.*H; %滤波处理
L=ifft2(ifftshift(K)); %傅里叶逆变换
subplot(121); imshow(I); %显示原始图像
subplot(122); imshow(L); %显示滤波后的图像
```

3.带阻滤波器

带阻滤波器是用来抑制距离频域中心一定距离的一个圆环区域的频率，可以用来消除一定频率范围的噪声。带阻滤波器包括理想带阻滤波器、巴特沃斯带阻滤波器和高斯带阻滤波器。

（1）理想带阻滤波器

理想带阻滤波器的产生公式为

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

其中 D_0 为需要阻止的频率点与频率中心的距离，W 为带阻滤波器的带宽。

（2）巴特沃斯高通滤波器

巴特沃斯带阻滤波器的产生公式为

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

其中 D_0 为需要阻止的频率点与频率中心的距离，W 为带阻滤波器的带宽，n 为巴特沃斯滤波器的阶数。

（3）高斯高通滤波

高斯带阻滤波的产生公式为

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

其中 D_0 为需要阻止的频率点与频率中心的距离，W 为带阻滤波器的带宽。

例 15.26　利用理想带阻滤波器对图像进行滤波。

```
clc, clear
I0=imread('coins.png');   %读入图像，
I=imnoise(I0,'gaussian',0,0.01);   %添加噪声
I=im2double(I);
[m,n]=size(I); %滤波的行数和列数
u=-n/2:(n/2-1); v=-m/2:(m/2-1);
[u,v]=meshgrid(u, v);
D=sqrt(u.^2+v.^2); D0=30; W=30
H=double(or(D<D0-W/2,D>D0+W/2)); %设计理想带阻滤波器
J=fftshift(fft2(I)); %时域图像转换到频域，并进行中心变换
K=J.*H; %滤波处理
L=ifft2(ifftshift(K)); %傅里叶逆变换
subplot(121); imshow(I0); %显示原始图像
subplot(122); imshow(L); %显示滤波后的图像
```

15.5　图像复原技术

在图像的采集、传送和转换过程中，会加入一些噪声，表现为图像模糊、失真和有噪声等。在实际应用中需要清晰、高质量的图像。图像复原就是要尽可能恢复退化图像的本来面目，它是沿图像退化的逆过程进行处理。图像复原技术主要包括图像的噪声模型、图像的滤波及常用的图像复原方法等。

图像复原和图像增强都是为了改善图像的质量，但是两者是有区别的。图像复原和图像增强的区别在于：图像增强不考虑图像是如何退化的，而是试图采用各种技术来增强图像的视觉效果。而图像复原不同，需要知道图像退化的机制和过程等先验知识，据此找到一种相应的逆处理方法，从而得到恢复的图像。

假定成像系统是线性位移不变系统，则获取的图像 $g(x,y)$ 表示为

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

其中 $f(x,y)$ 表示理想的、没有退化的图像，$g(x,y)$ 是退化后观察得到的图像，$n(x,y)$ 为加性噪声。图像复原是在已知 $g(x,y)$、$h(x,y)$ 和 $n(x,y)$ 的一些先验知识的条件下，来求解 $f(x,y)$ 的过程。

15.5.1　图像噪声模型

图像噪声按照噪声和信号之间的关系可以分为加性噪声和乘性噪声两种。假设图像的像素值为 $F(x,y)$，噪声信号为 $N(x,y)$。如果混合叠加信号为 $F(x,y)+N(x,y)$ 的形式，则这种噪声为加性噪声。如果叠加后信号为 $F(x,y)[1+N(x,y)]$ 的形式，则这种噪声为乘性噪声。

15.5.2　空域内的滤波复原

1.均值滤波

均值滤波复原包括算术均值滤波器和集合均值滤波器。在坐标点 (x,y)，大小为 $m\times n$ 的矩形窗口表示为 S_{xy}，算术平均值是窗口 S_{xy} 中被干扰图像 $g(x,y)$ 的平均值，即

$$f(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}}g(s,t)\,.$$

几何均值滤波器复原图像时，表达式为

$$f(x,y)=\left[\prod_{(s,t)\in S_{xy}}g(s,t)\right]^{\frac{1}{mn}}\,.$$

逆谐波均值滤波器的表达式为

$$f(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}},$$

其中，Q为滤波器的阶数。当Q为正数时，该滤波器可以去除椒噪声；当Q为负数时，该滤波器可以去除盐噪声。但是，该滤波器不能同时消除椒噪声和盐噪声。当$Q=-1$时，该滤波器为谐波均值滤波器。

例 15.27　对图像进行算术均值和几何均值滤波。

```
clc, clear
I=imread('cameraman.tif');I=im2double(I); %读入图像，
I=imnoise(I,'gaussian',0.05);  %添加高斯噪声
PSF=fspecial('average',3); %产生滤波器
J1=imfilter(I,PSF); %算术均值滤波
J2=exp(imfilter(log(I),PSF)); %几何均值滤波
subplot(131); imshow(I); %显示含有噪声图像
subplot(132); imshow(J1); %显示算术均值滤波图像
subplot(133); imshow(J2); %显示几何均值滤波图像
```

例 15.28　采用逆谐波均值滤波器对图像进行滤波。

```
clc, clear
I=imread('cameraman.tif');I=im2double(I);
I=imnoise(I,'salt & pepper',0.01);  %添加椒盐噪声
PSF=fspecial('average',3); %产生滤波器
Q1=1.6; Q2=-1.6;
J1=imfilter(I.^(Q1+1),PSF);
J2=imfilter(I.^Q1,PSF);
J=J1./J2; %逆谐波滤波，Q为正
K1=imfilter(I.^(Q2+1),PSF);
K2=imfilter(I.^Q2,PSF);
K=K1./K2; %逆谐波滤波，Q为负
subplot(131); imshow(I); %显示含有噪声图像
subplot(132); imshow(J); %显示滤波图像，Q为正
subplot(133); imshow(K); %显示滤波图像，Q为负
```

2.顺序统计滤波

顺序统计滤波包括中值滤波、最大值滤波和最小值滤波。中值滤波能够很好地保留图像的边缘，非常适合去除椒盐噪声，效果优于均值滤波。

下面首先介绍中值滤波。在坐标点(x,y)，大小为$m\times n$的窗口表示为S_{xy}，中值滤波是选取窗口S_{xy}中被干扰图像$g(x,y)$的中值，作为坐标点(x,y)输出，公式为

$$f(x,y)=\underset{(x,y)\in S_{xy}}{\text{Median}}\, g(x,y).$$

最大值滤波器也能够去除椒盐噪声，但会从黑色物体的边缘去除一些黑色像素。最大值滤波器的公式为

$$f(x,y)=\underset{(x,y)\in S_{xy}}{\text{Max}}\ g(x,y).$$

最小值滤波器和最大值滤波器类似，但是会从白色物体的边缘去除一些白色像素。最小

值滤波器的公式如下

$$f(x,y)=\min_{(x,y)\in S_{xy}} g(x,y).$$

在 MATLAB 中，可以通过函数 ordfilt2 进行二维排序滤波。函数 medfilt2 在进行滤波时，选择的是排序后的中值。函数 ordfilt2 在进行滤波时，可以选择排序后的任意一个值作为输出。该函数的调用格式如下。

J= ordfilt2(I,order,domain)：该函数对图像 I 进行二维排序滤波，将矩阵 domain 中的非 0 值对应的元素进行排序，order 为选择的像素位置。返回值 J 为排序滤波后得到的图像。

例 15.29　采用二维排序滤波对图像进行复原。

```
clc, clear
I=imread('cameraman.tif'); I=im2double(I);
I=imnoise(I, 'salt & pepper', 0.1); %添加椒盐噪声
domain=[0 1 1 0; 1 1 1 1; 1 1 1 1; 0 1 1 0]; %窗口模板
J=ordfilt2(I, 6, domain); %顺序滤波
subplot(121);    imshow(I);
subplot(122);    imshow(J);
```

15.5.3　图像复原方法

图像复原的常用方法，主要包括逆滤波复原、维纳滤波复原、约束最小二乘法复原、Lucy-Richardson 复原和盲解卷积复原等。

1.逆滤波复原

$f(x,y)$ 表示输入图像，即理想的、没有退化的图像，$g(x,y)$ 是退化后观察得到的图像，$n(x,y)$ 为加性噪声。通过傅里叶变换到频域后为

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

图像复原的目的是给定 $G(u,v)$ 和退化函数 $H(u,v)$，以及关于加性噪声的相关知识，得到原图像 $F(u,v)$ 的估计图像 $\hat{F}(u,v)$，使该图像尽可能地逼近原图像 $F(x,y)$。用于复原一幅图像的最简单的方法是构造如下的公式

$$\hat{F}(x,y)=\frac{G(u,v)}{H(u,v)}.$$

然后通过 $\hat{F}(u,v)$ 的傅里叶反变换得到图像的估计值，称为逆滤波。逆滤波是一种非约束复原方法。非约束复原是指在已知退化图像 $G(u,v)$ 的情况下，根据对退化模型 $H(u,v)$ 和噪声 $N(u,v)$ 的一些知识，做出对原图像的估计 $\hat{F}(u,v)$，使得某种事先确定的误差准则为最小。在得到误差最小的解的过程中，没有任何约束条件。

对于直接逆滤波，由于存在噪声的影响，退化图像的估计公式为

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

在进行逆滤波时，如果某个区域 $H(u,v)$ 为 0 或非常小，而 $N(u,v)$ 不为 0 且不是很小，则上式中的第 2 项往往比第 1 项大得多，从而使噪声放大，产生较大的误差。为了避免 $H(u,v)$ 的值太小，可以在逆滤波时加一些限制，只在原点附近的有限邻域内进行复原，称为伪逆滤波。

2.维纳滤波复原

在 MATLAB 中，采用函数 deconvwnr 进行图像的维纳滤波复原。该函数的调用格式如下。

J=deconvwnr(I,PSF,NSR)：该函数中对输入图像 I 进行维纳滤波复原，PSF 为点扩展函数，NSR 为信噪比。该函数的返回值 J 为采用维纳滤波复原后得到的图像。

J=deconvwnr(I,PSF,NCORR,ICORR)：该函数中参数 NCORR 为噪声的自相关函数，ICORR 为原始图像的自相关函数。

例 15.30　通过维纳滤波对含有噪声的运动模糊图像进行复原。

```
clc, clear
I=imread('cameraman.tif'); I=im2double(I);
LEN=21;  %运动位移为21像素
THETA=11; %角度为11度
PSF=fspecial('motion', LEN, THETA);
J=imfilter(I, PSF, 'conv', 'circular'); %运动模糊
noise_mean=0;
noise_var=0.0001;
K=imnoise(J, 'gaussian', noise_mean, noise_var); %添加高斯噪声
subplot(121);  imshow(I); %显示原始图像
subplot(122);  imshow(K); %显示退化图像
NSR1=0;
L1=deconvwnr(K, PSF, NSR1); %维纳滤波复原
NSR2=noise_var/var(I(:)); %计算真实的信噪比
L2=deconvwnr(K, PSF, NSR2); %维纳复原
figure; subplot(121);  imshow(L1); %显示NSR为0时的复原图像
subplot(122);  imshow(L2); %显示NSR为真实值时的复原图像
```

3.约束最小二乘法复原

在 MATLAB 中，采用函数 deconvreg 进行图像的约束最小二乘法复原。该函数的调用格式如下。

J=deconvreg(I,PSF,NOISEPOWER,LRANGE,REGOP)：该函数对输入图像 I 进行约束最小二乘法复原，PSF 为点扩展函数，NOISEPOWER 为噪声的强度，默认值为 0，LRANGE 为拉格朗日算子的搜索范围，默认值为$[10^{-9},10^{9}]$，REGOP 为约束算子，返回值 J 为复原后得到的图像。

例 15.31　通过拉格朗日算子进行图像复原。

```
clc, clear
I=imread('rice.png'); I=im2double(I);
PSF=fspecial('gaussian', 10, 5); %产生PSF
J=imfilter(I, PSF, 'conv'); %图像退化
v=0.02;
K=imnoise(J, 'gaussian', 0, v); %添加噪声
NP=v*prod(size(I));
[L, LAGRA]=deconvreg(K, PSF, NP); %图像复原
edged=edgetaper(K, PSF); %提取边缘
subplot(131); imshow(I); %显示原图像
subplot(132);  imshow(K); %显示退化图像
subplot(133);  imshow(edged); %显示边缘
M1=deconvreg(edged, PSF, [], LAGRA); %图像复原
M2=deconvreg(edged, PSF, [], LAGRA*30); %增大拉格朗日算子
M3=deconvreg(edged, PSF, [], LAGRA/30); %减少拉格朗日算子
figure; subplot(131);  imshow(M1);
subplot(132);  imshow(M2);
subplot(133);  imshow(M3);
```

4.Lucy-Richardson 复原

在 MATLAB 中，函数 deconvlucy 采用加速收敛的 Lucy-Richardson 算法对图像进行复原。该函数的调用格式如下。

J=deconvlucy(I,PSF,NUMIT,DAMPAR,WEIGHT,READOUT,SUBSMPL)：该函数对输入图像 I 采用 Lucy-Richardson 算法进行图像复原，PSF 为点扩展函数，NUMIT 为算法的重复次数，默认值为 10，DAMPAR 为偏差阈值，默认值为 0，WEIGHT 为像素的加权值，默认为原始图像的数值，,READOUT 为噪声矩阵，默认值为 0，SUBSMPL 为子采样时间，默认值为 1，返回值 J 为复原后得到的图像。

例 15.32　对高斯噪声采用 Lucy-Richardson 算法进行图像复原。

```
clc, clear
I=imread('cameraman.tif'); I=im2double(I);
PSF=fspecial('gaussian', 7, 10);
v=0.0001;
J=imnoise(imfilter(I, PSF), 'gaussian',0, v); %图像退化
subplot(121); imshow(I); %显示原图像
subplot(122); imshow(J); %显示退化图像
WT=zeros(size(I));
WT(5:end-4, 5:end-4)=1;
K=deconvlucy(J, PSF, 20, sqrt(v)); %图像复原，迭代20次
L=deconvlucy(J, PSF, 20, sqrt(v), WT);
figure, subplot(121);   imshow(K);
subplot(122);   imshow(L);
```

5.盲解卷积复原

前面介绍的图像复原方法，需要预先知道退化图像的 PSF。在实际应用中，经常在不知道 PSF 的情况下对图像进行复原。盲解卷积复原方法，不需要预先知道 PSF，而且可以对 PSF 进行估计。盲解卷积复原算法的优点是在退化图像无先验知识的情况下，仍然能够进行复原。

在 MATLAB 中，采用函数 deconvblind 进行盲解卷积复原。该函数的调用格式如下。

[J,PSF]=deconvblind(I,INITPSF,NUMIT,DAMPAR, WEIGHT,READOUT)：该函数对输入图像 I 采用盲解卷积复原，INITPSF 是 PSF 的初始值。

例 15.33　对运动模糊图像采用盲解卷积算法进行复原。

```
clc, clear
I=imread('cameraman.tif'); I=im2double(I);
LEN=20; THETA=20; %设置参数
PSF0=fspecial('motion', LEN, THETA); %产生PSF
J=imfilter(I,PSF0, 'circular', 'conv'); %运动模糊
INITPSF=ones(size(PSF0));
[K, PSF]=deconvblind(J, INITPSF, 30); %图像复原
subplot(121);   imshow(PSF0,[]); %显示原PSF
subplot(122);   imshow(PSF,[]); %显示估计的PSF
axis auto;
figure(2), subplot(121);   imshow(J); %显示退化图像
subplot(122);   imshow(K); %显示复原图像
```

15.6　图像分割技术

图像分割，简单地说就是将一幅数字图像分割成不同的区域，在同一区域内具有在一定的准则下可认为是相同的性质，如灰度、颜色、纹理等，而任何相邻区域之间其性质具有明显的区别。图像分割技术主要包括边缘分割技术、阈值分割技术和区域分割技术等。

15.6.1 边缘分割技术

边缘检测是检测图像特性发生变化的位置。边缘检测是利用物体和背景在某种图形特性上的差异来实现的。常见的边缘检测方法有微分算子、Canny 算子和 Log 算子等。常用的微分算子有 Sobel 算子、Roberts 算子和 Prewitt 算子等。

1.图像中的线段

将图像点 (x, y) 某个邻域中每个像素值都与模板中对应的系数相乘，然后将结果进行累加，从而得到该点的新像素值。如果邻域的大小为 $m \times n$，则总共有 mn 个系数。这些系数组成的矩阵，称为模板或算子。通常采用的最小模板是 3×3。

对于图像中的间断点，常用的检测模板为

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

对于图像中的线段，常用的检测模板为

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & -1 & -2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}.$$

这些模板分别对应的线段为水平线段、$+45^\circ$ 线段、垂直线段和 -45° 线段。在 MATLAB 中，可以利用模板，然后通过函数 imfilter 来实现对图像中间断点和线段的检测。

例 15.34　检测图像中的线段。

```
clc, clear
I=imread('gantrycrane.png'); I=rgb2gray(I); %读入图像，并转换为灰度图像
h1=[-1, -1. -1; 2, 2, 2; -1, -1, -1]; %模板
h2=[-1, -1, 2; -1, 2, -1; 2, -1, -1];
h3=[-1, 2, -1; -1, 2, -1; -1, 2, -1];
h4=[2, -1, -1; -1, 2, -1; -1, -1, 2];
J1=imfilter(I, h1); %线段检测
J2=imfilter(I, h2);
J3=imfilter(I, h3);
J4=imfilter(I, h4);
J=J1+J2+J3+J4; %4条线段叠加
subplot(121);   imshow(I); %显示灰度图像
subplot(122);   imshow(J); %显示检测到的线段
```

2.微分算子

常用的微分算子有 Sobel 算子、Roberts 算子和 Prewitt 算子等。通过这些算子对图像进行滤波，就可以得到图像的边缘。

下面以 Sobel 算子为例说明微分算子的用法。Sobel 算子的模板如下表示。

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

在 MATLAB 中，函数 edge 可以采用 Sobel 算子进行边缘检测。该函数的调用格式如下。

BW=edge(I,'sobel',thresh,direction)：该函数采用 Sobel 算子对图像 I 进行边缘检测，thresh 为分割阈值，如果不设置 thresh 或为空矩阵[]，则采用系统自动计算 thresh 的值，direction 为方向参数，可以取值为 horizontal、vertical 或 both，系统的默认值为 both。

例 15.35　采用 Sobel 算子进行图像的水平边缘检测。

```
clc, clear
I=imread('gantrycrane.png'); I=rgb2gray(I); I=im2double(I); %读入图像并转换
```

```
[J, thresh]=edge(I, 'sobel', [], 'horizontal'); %采用Sobel算子检测边缘
subplot(121);   imshow(I); %显示灰度图像
subplot(122);   imshow(J); %显示水平边缘图像
```

15.6.2 阈值分割技术

阈值分割技术是最简单的一种图像分割方法，关键在于寻找合适的阈值，通常根据图像的直方图来选取。

1.全局阈值

如果在整个图像中只使用一个阈值，则这种方法叫做全局阈值法。

对于物体和背景对比比较明显的图像，其灰度直方图为双峰形状，可以选择两峰之间的波谷对应的像素值作为全局阈值，将图像分割为目标对象和背景。其公式如下

$$g(x,y)=\begin{cases}1, & f(x,y)>T,\\ 0, & f(x,y)\le T,\end{cases}$$

其中 $f(x,y)$ 为点 (x,y) 的像素值，$g(x,y)$ 为分割后的图像，T 为全局阈值，通常通过直方图来获取全局阈值。

例 15.36　获取灰度图像的直方图。

```
clc, clear
I=imread('rice.png'); %读入图像
subplot(121);   imshow(I); %显示图像
subplot(122);   imhist(I, 200); %显示直方图
```

例 15.37　采用全局阈值对图像进行分割。

```
clc; clear;
I=imread('rice.png');
J=I>120; %图像分割，分割阈值为 120
K=I>130; %图像分割，分割阈值为 130
subplot(131), imshow(I) %显示原图像
subplot(132);   imshow(J)
subplot(133);   imshow(K)
```

在 MATLAB 中，函数 im2bw 可以通过全局阈值将灰度图像和彩色图像转换为二值图像。该函数的调用格式如下。

BW=im2bw(I,level)：该函数将灰度图像 I 转换为二值图像，采用的阈值为 level，level 的大小介于[0,1]之间。函数的返回值 BW 为二值图像。

BW=im2bw(X,map,level)：该函数将彩色的索引图像 X 转换为二值图像，其中 map 为颜色表，level 为阈值，函数的返回值 BW 为二值图像。

BW=im2bw(RGB,level)：该函数将 RGB 真彩色图像转换为二值图像，level 为阈值，函数的返回值 BW 为二值图像。

例 15.38　采用函数 im2bw 进行彩色图像分割。

```
load trees   %加载数据
BW=im2bw(X,map,0.4);
imshow(X,map), figure, imshow(BW)
```

2.Otsu 阈值分割

最大类间方差法，又称为 Otsu 算法，该算法是在灰度直方图的基础上采用最小二乘法原理推导出来的，具有统计意义上的最佳分割。它的基本原理是以最佳阈值将图像的灰度值分割成两部分，使两部分之间的方差最大，即具有最大的分离性。

在 MATLAB 中，函数 graythresh 采用 Otsu 算法获取全局阈值，获取全局阈值后，可以采用函数 im2bw 进行图像分割。函数 graythresh 的调用格式如下。

level=graythresh(I)：该函数采用 Otsu 算法获取灰度图像 I 的最优阈值，函数的返回值 level 为获取的阈值，大小介于[0,1]之间。

例 15.39　采用 Otsu 算法进行图像分割。

```
I=imread('coins.png');
level=graythresh(I); %获取阈值
BW=im2bw(I,level); %图像分割
imshow(I), figure, imshow(BW)
```

15.6.3　区域分割技术

图像分割的方法很多，除了边缘分割和阈值分割等方法以外，还可以采用区域分割。区域分割有分水岭分割等方法。

分水岭算法借鉴了形态学理论，是一种比较新的基于区域的图像分割算法。在该方法中，将一幅图像看成一个地形图，灰度值对应地形的高度值，高灰度值对应着山峰，低灰度值对应着山谷。水总是朝地势低的地方流动，直到某个局部低洼处，这个低洼处就是盆地。最终所有的水都会处于不同的盆地，盆地之间的山脊称为分水岭。

分水岭分割相当于是一个自适应的多阈值分割算法。在 MATLAB 中，函数 watershed 可以进行图像的分水岭分割。该函数的调用格式如下。

L=watershed(I)：该函数采用分水岭算法对图像 I 进行分割，返回值 L 为标记矩阵，其元素为整数值。第 1 个水盆被标记为 1，第 2 个水盆标记为 2。分水岭被标记为 0。

L=watershed(I,conn)：该函数中通过参数 conn 设置连通区域。对于二维图像，参数 conn 可取值为 4 和 8，默认值为 8。对于三维图像，conn 可取值为 6、18 和 26，默认值为 26。

例 15.40　采用分水岭算法分割图像。

```
clc, clear
I=imread('circbw.tif'); %读入图像
J=watershed(I,8); %分水岭分割
subplot(121);   imshow(I); %显示原图像
subplot(122);   imshow(J); %显示分割结果
```

15.7　图像变换技术

图像变换是将图像从空间域变换到变换域。在变换域内处理结束后，将处理结果进行反变换到空间域。图像变换技术主要包括 Radon 变换和反变换、傅里叶变换和反变换、离散余弦变换和反变换等。图像变换可以应用于图像滤波、图像压缩和图像识别等很多领域。

15.7.1　离散余弦变换

在 MATLAB 中，采用函数 blkproc 进行图像的块操作，该函数的详细使用情况如下。

B=blkproc(A,[m,n],fun)：该函数对矩阵 A 进行块操作，块的大小为 $m \times n$，对块的操作函数为 fun。返回值 B 为进行块操作后得到的矩阵，A 和 B 的大小相同。

离散余弦变换主要应用于图像数据压缩方面：在 JPEG 图像压缩算法中，首先将输入图像划分为 8×8 或者是 16×16 的方块，然后对每一个方块执行二维离散余弦变换，最后将得到的量化的变换系数进行编码和传送，形成压缩后的图像格式。在接收端，将量化的离散余弦系数进行解码，并对每个 8×8 或 16×16 方块进行二维离散余弦反变换，最后将操作完成后的块组合成一幅完整的图像。8×8 方块经正变换后得到的系数矩阵的左上角代表图像的低频分量，右下角代表图像的高频分量。

例 15.41　通过离散余弦变换进行图像压缩。

```
clc, clear;
I=imread('rice.png'); J=im2double(I);
T=dctmtx(8);
K=blkproc(J, [8 8], 'P1*x*P2', T, T'); %计算离散余弦变换矩阵
mask=flip(tril(ones(4,4))), mask(8,8)=0 %定义掩膜矩阵，只选择左上角的10个系数
K2=blkproc(K, [8 8], 'P1.*x', mask); %系数选择
```

```
L=blkproc(K2, [8 8], 'P1*x*P2', T', T); %对每个小方块进行离散余弦反变换
subplot(121); imshow(J), title('显示原图像')
subplot(122); imshow(L), title('显示变换图像')
```

15.7.2 Hough 变换

Hough 变换是图像处理中从图像中识别几何形状的基本方法之一。由 Paul Hough 于 1962 年提出，最初只用于二值图像直线检测，后来扩展到任意形状的检测。Hough 变换的基本原理在于利用点与线的对偶性，将原始图像空间给定的曲线通过曲线表达形式变为参数空间的一个点。这样就把原始图像中给定曲线的检测问题转化为寻找参数空间中的峰值问题。

Hough 变换根据如下公式

$$x\cos\theta + y\sin\theta = \rho$$

把 $x-y$ 平面的图像转换为 $\theta-\rho$ 参数平面上的图像矩阵。在 MATLAB 中，Hough 变换的函数包括函数 hough、函数 houghpeaks 和函数 houghlines。函数 hough 用来进行 Hough 变换，该函数的调用格式如下。

[H,theta,rho]=hough(BW)：该函数对二值图像 BW 进行 Hough 变换，返回值 H 为 Hough 变换矩阵，theta 为变换角度 θ，rho 为变换半径 r。

[H,theta,rho]=hough(BW,ParameterName,ParameterValue)：当参数 ParameterName 取值为‘RhoResolution’时，指明变换的 ρ 轴间隔，ρ 的取值在 ParameterValue 中给出，默认为 1。当参数 ParameterName 取值为‘Theta’时，指明变换 θ 的取值，θ 的范围为 $-90^\circ \le \theta < 90^\circ$。

例15.42　通过图像的Hough变换检测直线。

```
clc, clear
I=imread('gantrycrane.png'); I=rgb2gray(I);
BW=edge(I, 'canny'); %获取图像的边缘
[H, Theta, Rho]=hough(BW, 'RhoResolution', 0.5, 'Theta', -90:0.5:89.5);
P=houghpeaks(H, 5); %获取5个最值点
x=Theta(P(:, 2)); %横坐标
y=Rho(P(:, 1)); %纵坐标
set(0,'defaultFigurePosition',[100,100,1000,500]);
set(0,'defaultFigureColor',[1 1 1])
subplot(121);
imshow(imadjust(mat2gray(H)), 'XData', Theta, 'YData', Rho,...
    'InitialMagnification', 'fit'); %绘制Hough变换结果
axis on; axis normal; hold on;
plot(x, y, 's', 'color', 'white');
lines=houghlines(BW, Theta, Rho, P, 'FillGap', 5, 'MinLength', 7);%检测直线
subplot(122);
imshow(I); hold on; maxlen=0;
for k=1:length(lines)   %绘制多条直线
    xy=[lines(k).point1; lines(k).point2];
    plot(xy(:,1), xy(:, 2), 'linewidth', 2, 'color', 'green');
    plot(xy(1,1), xy(1, 2), 'linewidth', 2,   'color', 'yellow');
    plot(xy(2,1), xy(2, 2), 'linewidth', 2,   'color', 'red');
    len=norm(lines(k).point1-lines(k).point2);
    if (len>maxlen) %获取最长直线坐标
        maxlen=len;
```

```
            xylong=xy;
        end
    end
    hold on;
    plot(xylong(:, 1), xylong(:, 2), 'color', 'blue'); %绘制最长的直线
```

15.8 图像特征分析

例 15.43 利用 edge 函数提取图像轮廓，绘制出对象的边界和提取边界坐标信息。

```
I= imread('leaf1.bmp'); %读入非工具箱图像文件
c= im2bw(I, graythresh(I)); %I转换为二值图像
subplot(131);imshow(I); %显示原图
c=flipud(c); %实现矩阵c上下翻转，由于xy坐标与像素坐标有垂直反转关系
b=edge(c,'canny'); %基于canny算子进行轮廓提取
[u,v]=find(b); %返回边界矩阵b中非零元素的位置
xp=v; yp=u;
x0=mean([min(xp),max(xp)]); y0=mean([min(yp),max(yp)]); %求极点O的直角坐标
xp1=xp-x0; yp1=yp-y0;
[cita,r]=cart2pol(xp1,yp1);%直角坐标转换成极坐标
q=sortrows([cita,r]); %把cita中元素按照从小到大排列，r中的元素跟着联动
cita=q(:,1); %赋角度值
r=q(:,2); %赋半径值
subplot(132);polar(cita,r); %画出极坐标下的轮廓图
[x,y]=pol2cart(cita,r); %极坐标转换成直角坐标
x=x+x0; y=y+y0;
subplot(133);plot(x,y); %画出直角坐标下的轮廓图
```

区域重心的坐标是根据所有属于区域的点计算出来的。对于 $M \times N$ 的数字图像 $f(x,y)$，其重心定义

$$\overline{X} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} xf(x,y),$$

$$\overline{Y} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} yf(x,y).$$

在 MATLAB 中，图像的区域重心可以通过函数 regionprops 的‘Centroid’属性得到。

例 15.44 利用函数 regionprops 求区域的面积和重心。

```
clc, clear
I= imread('leaf1.bmp'); I= im2bw(I); %转换为二值图像
Ar=regionprops(I,'Area')   %求C的面积
Ce=regionprops(I,'Centroid')   %求C的重心
```

15.9 形态学图像处理

形态学是一种应用于图像处理和模式识别领域的新的方法。腐蚀运算和膨胀运算是数学形态学的两个基本运算。图形 A（感兴趣目标）和结构集合 B，B 称为结构元素。结构元素通常是个圆盘，但它其实可以是任何形状。

15.9.1 结构元素

结构元素是膨胀和腐蚀的最基本组成部分，用于测试输入图像。二维结构元素是由数

值 0 和 1 组成的矩阵。结构元素的原点执行了图像中需要处理的像素范围，结构元素中数值为 1 的点决定结构元素的邻域像素在进行膨胀或腐蚀操作时是否参与运算。

在 MATLAB 中，结构元素定义为一个 STREL 对象。采用函数 strel 创建任意大小和形状的 STREL 对象。函数 strel 支持常用的形状，例如线性（line）、矩形（rectangle）、方形（square）、球形（ball）、钻石型（diamond）和自定义的任意型（arbitrary）。

对于 STREL 对象的结构元素可以分解为较小的块，称为结构元素的分解。通过结构元素的分解，可以提高执行效率和运行速度。例如，对大小为 8×8 的正方形结构元素进行膨胀操作，可以首先对 1×8 的结构元素进行膨胀，然后再对 8×1 的结构元素进行膨胀。在 MATLAB 软件中，可以通过函数 getsequence 进行结构元素的分解。

例 15.45 创建结构元素，并对结构元素进行分解。

```
clc, clear
se1=strel('square', 3) %矩形结构元素
se2=strel('line', 10, 45) %长度10，角度45度
se3=strel('diamond',3)  %钻石型结构元素
seq=getsequence(se3) %结构元素分解
seq(1), seq(2), seq(3) %显示3个较小的结构元素
```

15.9.2 膨胀与腐蚀

膨胀是将与物体接触的所有背景点合并到该物体中，使边界向外部扩张的过程。

设 A 和 B 是 Z^2 的子集，则把图像 A 沿矢量 x 平移一段距离记作 $A+x$ 或 A_x，其定义为

$$A_x = \{c : c = a + x, \forall a \in A\}.$$

结构元素 B 的映射－B 或 $\hat{B}$，定义为

$$\hat{B} = \{x, x = -b, \forall b \in B\}.$$

A 补集记作 $\overline{A}$ 或 A^c，定义为

$$A^c = \{x : x \notin A\}.$$

对于两幅图像 A 和 B，如果 $A \bigcap B \neq \Phi$，则称 B 击中 A，记作 $B \uparrow A$；否则，如果 $A \bigcap B = \Phi$，则称 B 击不中 A。

结构元素 B 对图像 A 的膨胀，记作 $A \oplus B$，定义为

$$A \oplus B = \{x : \hat{B}_x \bigcap A \neq \Phi\}.$$

腐蚀和膨胀是对偶操作。腐蚀是一种消除边界点，使边界向内部收缩的过程。利用腐蚀操作，可以消除小且无意义的物体。集合 A 被结构元素 B 腐蚀，记作 $A \Theta B$，定义为

$$A \Theta B = \{x : B_x \subset A\}.$$

在 MATLAB 中，采用函数 imdilate 进行膨胀操作，采用函数 imerode 进行腐蚀操作。

例 15.46 创建图像并进行膨胀操作。

```
clc, clear
bw=zeros(9,9); bw(3:5, 4:6)=1 %创建二值图像
se=strel('square', 3) %矩形结构元素
bw2=imdilate(bw, se) %图像膨胀
subplot(121);   imshow(bw); %显示原图
subplot(122);   imshow(bw2); %显示膨胀后的图像
```

例 15.47 对二值图像进行膨胀。

```
clc, clear
bw=imread('text.png'); %读入图像
se=strel('line', 11, 90); %线性结构元素，长度为11，角度为90度
```

```
bw2=imdilate(bw, se); %图像膨胀
subplot(121);    imshow(bw); %显示原图
subplot(122);    imshow(bw2); %显示膨胀后的图像
```

例15.48　对灰度图像进行腐蚀。

```
clc, clear
bw=imread('circles.png'); %读入图像
se=strel('disk', 11) %结构元素
bw2=imerode(bw, se); %图像腐蚀
subplot(121);    imshow(bw); %显示原图
subplot(122);    imshow(bw2); %显示腐蚀后的图像
```

例11.49　图像进行腐蚀和膨胀操作。

```
clc, clear
se=strel('rectangle', [40, 30]); %结构元素
bw1=imread('circbw.tif'); %读入图像
bw2=imerode(bw1, se); %腐蚀操作
bw3=imdilate(bw2, se); %膨胀操作
subplot(131);    imshow(bw1); %显示原图
subplot(132);    imshow(bw2); %显示腐蚀后的图像
subplot(133);    imshow(bw3); %先腐蚀后膨胀得到的图像
```

15.9.3　开运算和闭运算

结构元素 B 对 A 的开运算，记作 $A\circ B$，定义为

$$A\circ B=(A\Theta B)\oplus B,$$

即首先采用结构元素对 A 做腐蚀运算，然后再做膨胀运算，使用相同的结构元素。

闭运算是开运算的对偶运算，记作 $A\cdot B$，定义为

$$A\cdot B=(A\oplus B)\Theta B,$$

即首先用结构元素 B 对 A 做膨胀运算，然后再做腐蚀运算，使用相同的结构元素。

在 MATLAB 中，采用函数 imopen 进行二值图像或灰度图像的开运算，采用函数 imclose 进行闭运算。

例 15.50　对图像进行闭运算。

```
clc, clear
I=imread('circles.png'); %读入图像
se=strel('disk', 10) %结构元素
J=imclose(I, se); %闭运算
subplot(121);    imshow(I);    %显示原图像
subplot(122);    imshow(J); %显示闭运算得到的图像
```

15.9.4　组合形态学运算

图像的膨胀和腐蚀是最基本的形态学运算。组合形态学运算，主要包括高帽滤波、低帽滤波、边界提取和区域填充等。

1.高帽滤波和低帽滤波

图像的形态学高帽滤波定义为

$$H=A-(A\circ B),$$

其中 A 为输入的图像， B 为采用的结构元素，即从图像中减去形态学开操作后的图像。通过高帽滤波可以增强图像的对比度。

图像的形态学低帽滤波定义为

$$H = A - (A \cdot B),$$

其中 A 为输入的图像，B 为采用的结构元素，即从图像中减去形态学闭操作后的图像。通过低帽滤波可以获取图像的边缘。

在 MATLAB 中，采用函数 imtophat 对二值图像或灰度图像进行高帽滤波，采用函数 imbothat 进行低帽滤波。

例 15.51　通过高帽滤波和低帽滤波增强图像的对比度。

```
clc, clear
I=imread('pout.tif'); %读入图像
se=strel('disk', 3); %结构元素
J=imtophat(I, se); %高帽滤波
K=imbothat(I, se); %低帽滤波
L=imsubtract(imadd(I, J), K); %加减运算
subplot(121);   imshow(I); %显示原图像
subplot(122);   imshow(L); %显示增强图像
```

2.图像填充操作

在 MATLAB 中，采用函数 imfill 对二值图像或灰度图像进行填充操作。函数 imfill 的调用格式如下。

BW2=imfill(BW,locations)：该函数中通过参数 locations 指定进行填充时的点的坐标。

BW2=imfill(BW,‘holes’)：该函数通过参数 holes 可以填充二值图像中的空洞。

例 15.52　对二值图像进行填充操作。

```
clc, clear
I=imread('coins.png'); J=im2bw(I); %读入图像，并变为二值图像
K=imfill(J, 'holes'); %图像填充
subplot(121);   imshow(J); %显示二值图像
subplot(122);   imshow(K); %显示填充后的图像
```

3.最大值和最小值

对于一幅图像可以有多个局部极大值或极小值，但只有一个最大值或最小值。在 MATLAB 中，采用函数 imregionalmax 获取图像的所有局部极大值，采用函数 imregionalmin 获取局部极小值。

例 15.53　获取图像的所有局部极大值。

```
clc, clear;
I=10*ones(6, 10); I(3:4, 3:4)=13; %创建矩阵
I(4:5, 7:8)=18; I(2,8)=15
bw=imregionalmax(I) %获取局部极大值
```

在 MATLAB 中，函数 imextendedmax 获取指定阈值的局部极大值，函数 imextendedmin 获取指定阈值的局部极小值。

在 MATLAB 中，函数 imhmax 可以对图像中的极大值进行抑制，函数 imhmin 对图像中的极小值进行抑制。

例 15.54　通过函数 imhmax 对极大值进行抑制。

```
clc; clear
I=2*ones(5, 10); I(2:4, 2:4)=3; %创建矩阵
I(4:5, 6:7)=9; I(2,8)=5
J=imregionalmax(I) %获取所有极大值
K=imhmax(I, 4) %对小于阈值的极大值进行抑制
```

4. 图像的边界测定

例15.55　通过膨胀和腐蚀获取灰度图像的边缘。

```
clc, clear;
I=imread('rice.png'); %读入图像
se=strel('disk', 2); %结构元素
J=imdilate(I, se); %膨胀
K=imerode(I, se); %腐蚀
L=J-K; %相减
subplot(121);   imshow(I); %原始图像
subplot(122);   imshow(L); %边缘图像
```

在MATLAB中，采用函数bwperim获取二值图像的边缘。

例15.56　获取二值图像的边缘。

```
clc, clear
I=imread('circbw.tif'); %读入图像
J=bwperim(I, 8); %获取边缘
subplot(121);   imshow(I); %显示原始图像
subplot(122);   imshow(J); %显示边缘图像
```

在二值图像中，若一点的像素不为0，并且其邻域内至少有一个像素为0，则认为该点为边界点。

5.二值图像的形态学操作

在MATLAB中，通过函数bwmorph可以进行二值图像的大量形态学操作，例如图像的骨骼化、图像的细化，以及开操作和闭操作等。该函数的功能非常强大。

例15.57　移除二值图像的内部像素点。

```
clc, clear
I=imread('circles.png');   %读入图像
J=bwmorph(I, 'remove');   %移除内部像素点
subplot(121);   imshow(I); %显示原图像
subplot(122);   imshow(J); %显示结果图像
```

进行移除二值图像的内部像素点操作时，若某个像素点的4个邻域都为1，则将该像素值设置为0，只剩下图像的边界像素点。

15.9.5　二值图像的其它形态学操作

1.二值图像的标记

对于属于同一个像素连通区域的所有像素分配相同的编号，对不同的连通区域分配不同的编号，称为连通区域的标记。在MATLAB中，采用函数bwlabel和函数bwlabeln进行连通区域的标记操作。函数bwlabel只支持二维的二值图像，函数bwlabeln支持任意维数的二值图像。

例15.58　对二值图像进行标记。

```
clc, clear
BW=zeros(4, 8); %建立矩阵
BW(2:3, 2:3)=1; BW(2, 5)=1; BW(3, 7)=1
[L, num]=bwlabel(BW, 8) %二值图像的标记
```

运行结果为，该二值图像中包含3个连通区域。

2.二值图像的对象选择

在MATLAB中，采用函数bwselect在二值图像中选择单个的对象，要求图像必须是二维的。函数bwselect的调用格式如下。

BW2=bwselect(BW,c,r,n)：该函数对输入的二值图像BW进行对象选择，输入参数(c,r)为对象的像素点位置，c和r的维数相同，参数n为对象的连通类型，可取值为4或8。返回值BW2为选择了指定对象的二值图像，和原图像有相同大小。

BW2=bwselect(BW,n)：该函数采用交互的方式，用户采用鼠标选择像素点的位置。

例15.59　通过函数bwselect进行对象的选择。

```
clc, clear
I=imread('text.png'); %读入图像
c=[43, 185, 212]; r=[38, 68, 181]; %选择对象的坐标
J=bwselect(I,c,r,4); %对象选择
subplot(121);   imshow(I); %显示原图像
subplot(122);   imshow(J); %显示结果图像
```

3.二值图像的面积

面积是二值图像中像素值为1的像素个数。在MATLAB中，采用函数bwarea计算二值图像的面积，在计算二值图像的面积时，不是简单地计算像素值为1的像素个数，而是为每个像素设置一个权值，采用加权求和的方式得到面积。

例15.60　通过函数bwarea计算二值图像的面积。

```
clc, clear
I=imread('circbw.tif'); %读入图像
se=strel('disk', 3); %结构元素
J=imdilate(I, se); %膨胀
a1=bwarea(I) %原图像的面积
a2=bwarea(J) %膨胀后图像的面积
(a2-a1)/a1 %面积增加的百分比
subplot(121);   imshow(I); %显示原图像
subplot(122);   imshow(J); %显示膨胀后的图像
```

15.10　基于Hough变换的车牌图像倾斜校正算法的实现

Hough变换是一种形状匹配技术，它将原始图像空间中给定形状的曲线或直线变换成Hough空间中的一个点，即原始图像空间中给定形状的曲线或者直线上的所有点，都将集中到变换空间中的某个点上形成峰点。这样原始图像空间中给定曲线或者直线的检测问题就变成寻找变换空间的峰点问题，也就把检测整体特征（原始图像空间中给定曲线或者直线的点集特征）转换为检测局部特征（Hough空间中点的特征）。举一个简单例子来说明，在 xoy 平面内有一条直线，它与坐标原点 o 的距离为 ρ，它的法线与 x 轴正向夹角为 θ，直线上任意一点 (x,y)，均满足直线方程

$$\rho = x\cos\theta + y\sin\theta .$$

对于原图像空间中某一点 (x_i,y_i)，对应 (ρ,θ) 空间中一条正弦曲线

$$\rho = x_i\cos\theta + y_i\sin\theta .$$

在 (x,y) 平面内同一直线的点序列

$$\rho_0 = x\cos\theta_0 + y\sin\theta_0$$

变换到 (ρ,θ) 空间中，则表示经过同一点 (ρ_0,θ_0) 的所有正弦曲线。由于该直线上的所有点的Hough变换曲线均经过 (ρ_0,θ_0)，所以 (ρ_0,θ_0) 必成为 (ρ,θ) 空间中的一个峰点。

将(ρ,θ)量化为许多个小格。根据每个(x_i,y_i)点代入θ的量化值，算出每个ρ所得值经量化后落入某个小格内，使该小格的计数累加器加1，当全部(x_i,y_i)点变换完毕后，对(ρ,θ)空间中的小格进行统计，有大的计数值的小格对应于共线点，其(ρ,θ)可以作为直线拟合参数。

基于Hough变换的车牌图像倾斜校正的基本原理，利用Hough变换检测车牌的边框，确定边框直线的倾斜角度，根据倾斜角度旋转，获得校正后图像，其具体步骤如下。

（1）图像预处理。读取图像，转换成灰度图，去除离散噪声点。

（2）利用边缘检测，对图像中的水平线进行强化处理。

（3）基于Hough变换检测车牌图像中的边框，获取倾斜角度。

（4）根据倾斜角度，对车牌图像进行倾斜校正。

例15.61　利用Hough变换实现车牌图像的倾斜校正。

```
clc, clear
I=imread('车牌1.jpg'); %图像输入
I1=rgb2gray(I); %转换成灰度图像I1
I2=wiener2(I1,[5,5]); %对图像进行维纳滤波I2
I3=edge(I2,'sobel', 'horizontal');%用Sobel水平算子对图像边缘化
[H, Theta, Rho]=hough(I3, 'RhoResolution', 0.5, 'Theta', -90:0.5:89.5);
P=houghpeaks(H, 1) %获取1个最值点
theta=Theta(P(:, 2))
rho=Rho(P(:, 1))
I4=imrotate(I2,theta+90,'crop'); %对图像进行旋转矫正
subplot(121),imshow(I)
subplot(122),imshow(I2)
figure, subplot(121),imshow(I3)
subplot(122),imshow(I4)
```

习题15

15.1　碎纸片的拼接复原（2013 年全国大学生数学建模竞赛 B 题）

破碎文件的拼接在司法物证复原、历史文献修复以及军事情报获取等领域都有着重要的应用。传统上，拼接复原工作需由人工完成，准确率较高，但效率很低。特别是当碎片数量巨大，人工拼接很难在短时间内完成任务。随着计算机技术的发展，人们试图开发碎纸片的自动拼接技术，以提高拼接复原效率。请讨论以下问题：

（1）对于给定的来自同一页印刷文字文件的碎纸机破碎纸片（仅纵切），建立碎纸片拼接复原模型和算法，并针对附件 1、附件 2 给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，请写出干预方式及干预的时间节点。复原结果以图片形式及表格形式表达（见【结果表达格式说明】）。

（2）对于碎纸机既纵切又横切的情形，请设计碎纸片拼接复原模型和算法，并针对附件 3、附件 4 给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，请写出干预方式及干预的时间节点。复原结果表达要求同上。

（3）上述所给碎片数据均为单面打印文件，从现实情形出发，还可能有双面打印文件的碎纸片拼接复原问题需要解决。附件 5 给出的是一页英文印刷文字双面打印文件的碎片数

据。请尝试设计相应的碎纸片拼接复原模型与算法，并就附件 5 的碎片数据给出拼接复原结果，结果表达要求同上。

【数据文件说明】

（1）每一附件为同一页纸的碎片数据。

（2）附件 1、附件 2 为纵切碎片数据，每页纸被切为 19 条碎片。

（3）附件 3、附件 4 为纵横切碎片数据，每页纸被切为 11×19 个碎片。

（4）附件 5 为纵横切碎片数据，每页纸被切为 11×19 个碎片，每个碎片有正反两面。该附件中每一碎片对应两个文件，共有 2×11×19 个文件，例如，第一个碎片的两面分别对应文件 000a、000b。

【结果表达格式说明】

复原图片放入附录中，表格表达格式如下：

（1）附件 1、附件 2 的结果：将碎片序号按复原后顺序填入 1×19 的表格；

（2）附件 3、附件 4 的结果：将碎片序号按复原后顺序填入 11×19 的表格；

（3）附件 5 的结果：将碎片序号按复原后顺序填入两个 11×19 的表格；

（4）不能确定复原位置的碎片，可不填入上述表格，单独列表。