



中国科学技术大学
University of Science and Technology of China

MATLAB

2021.4.18



中国科学技术大学

University of Science and Technology of China

安装 MATLAB 软件

- ▶ 使用vlab实验中心
 - ▶ 网页：<https://vlab.ustc.edu.cn/>
- ▶ 正版软件网页
 - ▶ 网页：<http://zbh.ustc.edu.cn/zbh.php>

- ▶ 访问vlab网页，选择虚拟机管理



- ▶ 使用统一身份认证登录

密码
Password

登录

你也可以使用统一身份认证登录。



▶ 创建新虚拟机

虚拟机名称*

镜像选择* 我该选择哪个镜像?

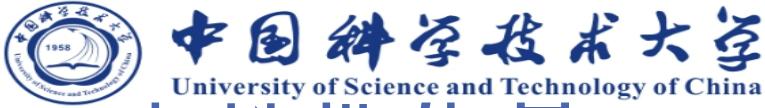
root 密码* 仅用于 SSH 登录和系统内 `su` 命令

请使用一个安全的密码, **任何知道该密码的人都拥有虚拟机的最高权限**

我已阅读并接受[服务条款](#)和[使用限制](#)*

创建

▶ 等待虚拟机生成



中国科学技术大学

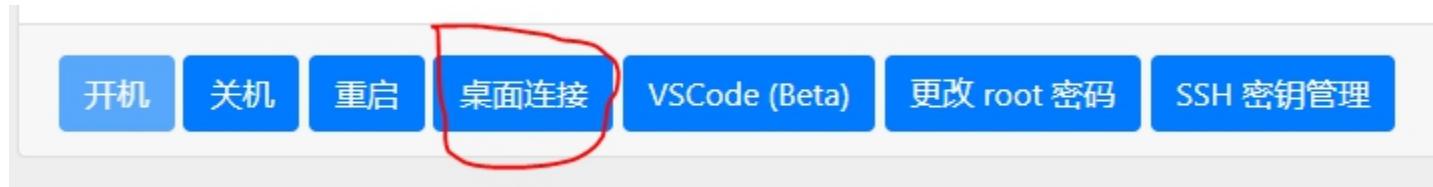
University of Science and Technology of China

虚拟机使用

▶ 开机



▶ 点击桌面连接



虚拟机使用

▶ 打开Matlab

▶ 点击桌面左上方的应用程序再按照下图打开Matlab



- ▶ 新建matlab脚本
 - ▶ 新建一个脚本文件并另存为test.m
- ▶ 常用语法
 - ▶ %用来注释
 - ▶ clc:清除命令窗口的内容
 - ▶ clear: 清除工作空间的所有变量
 - ▶ close all: 关闭所有的Figure窗口
 - ▶ 可以将这些命令写在Matlab程序第一行, 以便重复运行程序时, 释放上次使用的资源

▶ imread 函数

- ▶ $A = \text{imread}(\text{filename})$ 从 filename 指定的文件读取图像，并从文件内容推断出其格式
- ▶ $A = \text{imread}(\text{filename}, \text{fmt})$ 如果 imread 找不到具有 filename 指定的名称的文件，则会查找名为 filename(fmt) 的文件。
- ▶ $[A, \text{map}] = \text{imread}(\text{filename})$ 将 filename 中的索引图像读入 A，并将其关联的颜色图读入 map。图像文件中的颜色图值会自动重新调整到范围 [0,1] 中。
- ▶ $[A, \text{map}, \text{transparency}] = \text{imread}(\text{filename})$ 另外还返回图像透明度。此语法仅适用于 PNG、CUR 和 ICO 文件。对于 PNG 文件，如果存在 alpha 通道，transparency 会返回该 alpha 通道。对于 CUR 和 ICO 文件，它为 AND (不透明度) 掩码。

▶ figure函数

- ▶ `figure` 使用默认属性值创建一个新的图窗窗口。生成的图窗为当前图窗。
- ▶ `figure(Name,Value)` 使用一个或多个名称-值对组参数修改图窗的属性。例如，`figure('Color','white')` 将背景色设置为白色。
- ▶ `f = figure(____)` 返回 `Figure` 对象。可使用 `f` 在创建图窗后查询或修改其属性。
- ▶ `figure(f)` 将 `f` 指定的图窗作为当前图窗，并将其显示在其他所有图窗的上面。
- ▶ `figure(n)` 查找 `Number` 属性等于 `n` 的图窗，并将其作为当前图窗。如果不存在具有该属性值的图窗，MATLAB® 将创建一个新图窗并将其 `Number` 属性设置为 `n`。
- ▶ `subplot(m,n,p)` 将当前图窗划分为 $m \times n$ 网格，并在 `p` 指定的位置创建坐标区。
- ▶ `title(name)` 对网格进行命名

▶ imshow 函数

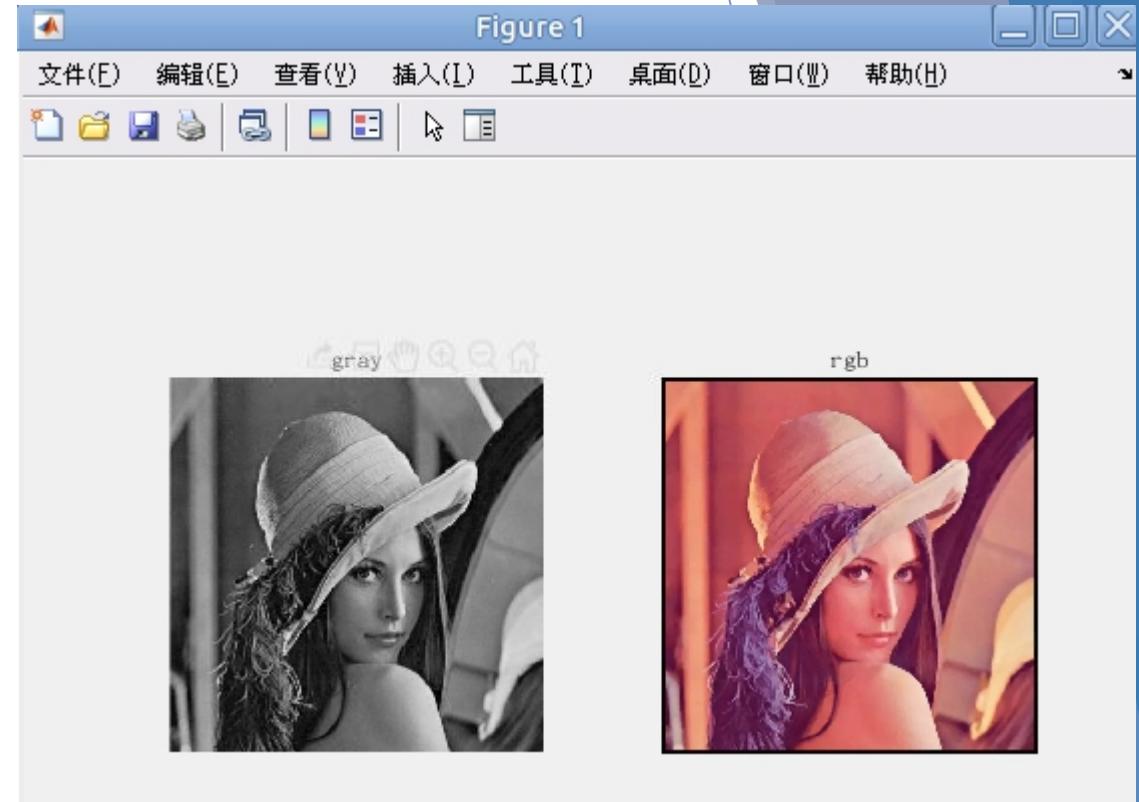
- ▶ imshow(I) 在图窗中显示图像 I。
 - ▶ imshow(X,map) 显示具有颜色图 map 的索引图像 X。
 - ▶ imshow(filename) 显示存储在由 filename 指定的图形文件中的图像。
- ▶ 图片路径
- ▶ 由于虚拟机没有输入法，对于图片存放路径要么选择英文存放路径；要么将Matlab脚本文件存放至与图片同一路徑；或者通过文件管理器复制图像路径。



▶ imwrite函数

- ▶ `imwrite(A,filename)` 将图像数据 A 写入 filename 指定的文件，并从扩展名推断出文件格式。imwrite 在当前文件夹中创建新文件。输出图像的位深取决于 A 的数据类型和文件格式。
- ▶ `imwrite(A,map,filename)` 将 A 中的索引图像及其关联的颜色图写入由 map filename 指定的文件。

```
1 %clear
2 - clc,clear,close all
3
4 - i1 = imread("/home/ubuntu/图片/1.jpeg")
5 - i2 = imread("/home/ubuntu/图片/2.jpeg")
6 - figure()
7 - subplot(1,2,1)
8 - imshow(i1)
9 - title('gray')
10 - subplot(1,2,2)
11 - imshow(i2)
12 - title('rgb')
13 - imwrite(i2,"/home/ubuntu/图片/3.jpeg")
```



- ▶ 在使用像素索引时，像素值与索引有一一对应的关系，例如，位于第3行第3列的像素值存储在矩阵元素 $(3,3)$ 中，可以使用MATLAB提供的函数进行访问。例如，使用命令 $A(3,3)$ 可以获取第3行第3列的像素值。
- ▶ 还可以使用命令 $A(3,3, :)$ 获取RGB图像中第3行第3的R,G,B值。



中国科学技术大学

University of Science and Technology of China

图像之间的转换

- ▶ $[X, \text{map}] = \text{rgb2ind}(\text{RGB})$: 彩色图转换成索引图
- ▶ $I = \text{rgb2gray}(\text{RGB})$: 彩色图转换成灰度图
- ▶ $[X, \text{cmap}] = \text{rgb2ind}(\text{RGB}, n)$: 使用具有 n 种量化颜色的最小方差量化法并加入抖动，将 RGB 图像转换为索引图像 X，关联颜色图为 cmap。
- ▶ $[X, \text{cmap}] = \text{rgb2ind}(\text{RGB}, \text{tol})$: 使用均匀量化法并加入抖动，将 RGB 图像转换为索引图像，容差为 tol。返回的颜色图 cmap 包含 $(\text{floor}(1/\text{tol})+1)^3$ 种或更少的颜色
- ▶ $X = \text{rgb2ind}(\text{RGB}, \text{inmap})$: 使用逆颜色图算法并加入抖动，将 RGB 图像转换为索引图像，指定的颜色图为 inmap。
- ▶ $\text{RGB} = \text{ind2rgb}(X, \text{map})$: 索引图转换成彩色图



中国科学技术大学

University of Science and Technology of China

图像之间的转换

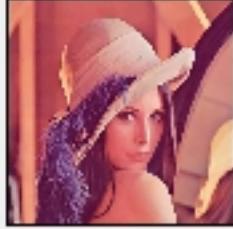
- ▶ $I = \text{ind2gray}(X, \text{map})$: 索引图转换成灰度图
- ▶ $[X, \text{map}] = \text{gray2ind}(I, n)$: 灰度图转换成索引图
 - ▶ 根据指定的灰度级数n和颜色图map, 将灰度图像I转换成索引图像X, n的默认值为64.
- ▶ $I = \text{mat2gray}(X, [\text{Xmin}, \text{Xmax}])$: 矩阵转换成灰度图像
 - ▶ 按指定的取值区间 $[\text{Xmin}, \text{Xmax}]$ 将数据矩阵X转化为图像I.如果不指定区间 $[\text{Xmin}, \text{Xmax}]$ 时, MATLAB则自动将X矩阵中最小设为Xmin, 最大设为Xmax。



示例

```
1 %clear
2 clc,clear,close all
3
4 rgb1 = imread("./home/ubuntu/图片/2.jpeg")
5 f=figure()
6 subplot(3,2,1)
7 imshow(rgb1)
8 title('RGB1')
9
10 [x1,map1] = rgb2ind(rgb1,32)
11 subplot(3,2,2)
12 imshow(x1,map1)
13 title('IND1')
14
15 rgb2 = ind2rgb(x1,map1)
16 subplot(3,2,3)
17 imshow(rgb2)
18 title('RGB2')
19
20 i1 = ind2gray(x1,map1)
21 subplot(3,2,4)
22 imshow(i1)
23 title('GRAY1')
24
25 [x2,map2] = gray2ind(i1)
26 subplot(3,2,5)
27 imshow(x2,map2)
28 title('IND2')
29
30 A = magic(10)
31 i2 = mat2gray(A)
32 subplot(3,2,6)
33 imshow(i2)
34 title('GRAY2')
```

RGB1



IND1



RGB2



GRAY1

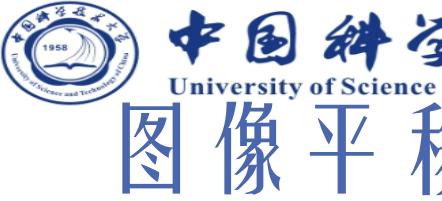


IND2



GRAY2





图像平移

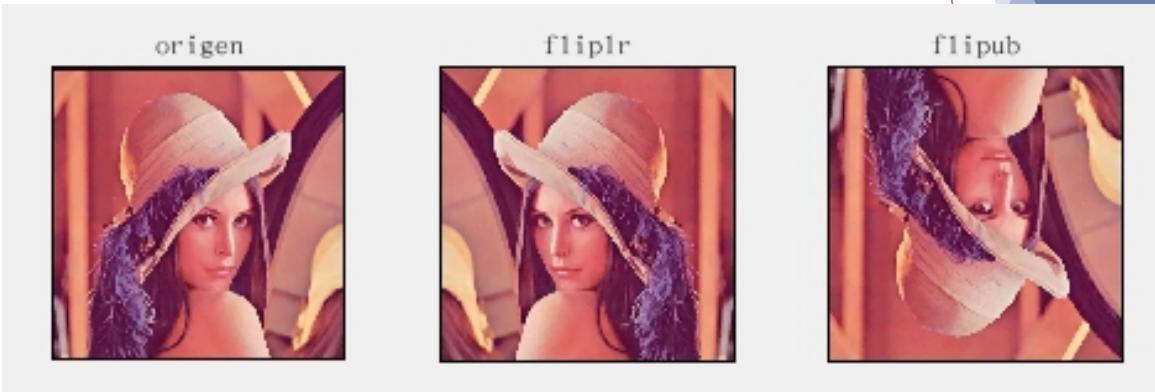
- ▶ strel用来创建形态学结构元素
- ▶ translate(SE,[y,x])在原结构元素上y和x方向平移
- ▶ imdilate是膨胀图像函数
- ▶ 可以利用上面3个结构及函数完成图像的平移
- ▶ 示例：

```
1 %clear
2 - clc,clear,close all
3
4 - rgb1 = imread('/home/ubuntu/图片/2.jpeg')
5 - f = figure()
6 - se = translate(strel(1),[100,100])
7 - rgb2 = imdilate(rgb1,se)
8 - subplot(1,2,1)
9 - imshow(rgb1)
10 - title('origen')
11 - subplot(1,2,2)
12 - imshow(rgb2)
13 - title('translation')
```



- ▶ `fliplr(I)`: 对图像I进行水平翻转
- ▶ `flipud(I)`: 对图像I进行垂直翻转
- ▶ 示例：

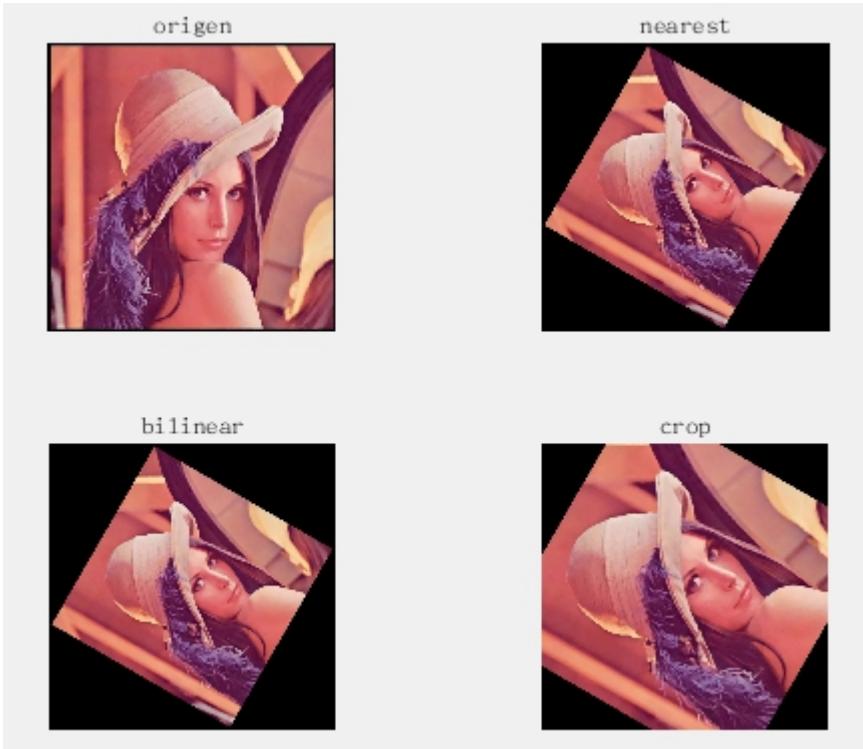
```
1 %clear
2 - clc,clear,close all
3
4 - rgb1 = imread('/home/ubuntu/图片/2.jpeg')
5 - f = figure()
6 - rgb2 = fliplr(rgb1)
7 - rgb3 = flipud(rgb1)
8 - subplot(1,3,1)
9 - imshow(rgb1)
10 - title('origen')
11 - subplot(1,3,2)
12 - imshow(rgb2)
13 - title('fliplr')
14 - subplot(1,3,3)
15 - imshow(rgb3)
16 - title('flipud')
```

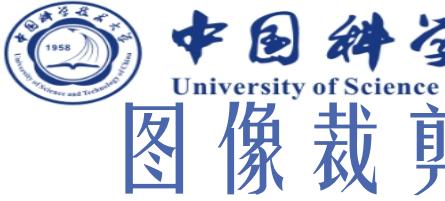


- ▶ imrotate(I,angle,method,bbox):
 - ▶ angle决定旋转的角度
 - ▶ method属于{, , }, 决定插值方式
 - ▶ ‘nearest’:最近邻插值，为默认值。
 - ▶ ‘bilinear’:双线性插值。
 - ▶ ‘bicubic’:双三次插值。输出像素值是最近的 4×4 邻域中像素的加权平均值。
 - ▶ bbox 参数来定义输出图像的大小。
 - ▶ ‘crop’使输出图像 J 与输入图像 I 大小相同，裁剪旋转后的图像以适应边界框。
 - ▶ ‘loose’:使输出图像 J 足够大，以包含整个旋转后的图像。J 大于 I。为默认值

► 示例

```
1 %clear
2 clc,clear,close all
3
4 rgb1 = imread('/home/ubuntu/图片/2.jpeg')
5 f = figure()
6 rgb2 = imrotate(rgb1,60)
7 rgb3 = imrotate(rgb1,60,'bilinear')
8 rgb4 = imrotate(rgb1,60,'bilinear','crop')
9 subplot(2,2,1)
10 imshow(rgb1)
11 title('origen')
12 subplot(2,2,2)
13 imshow(rgb2)
14 title('nearest')
15 subplot(2,2,3)
16 imshow(rgb3)
17 title('bilinear')
18 subplot(2,2,4)
19 imshow(rgb4)
20 title('crop')
```





图像裁剪

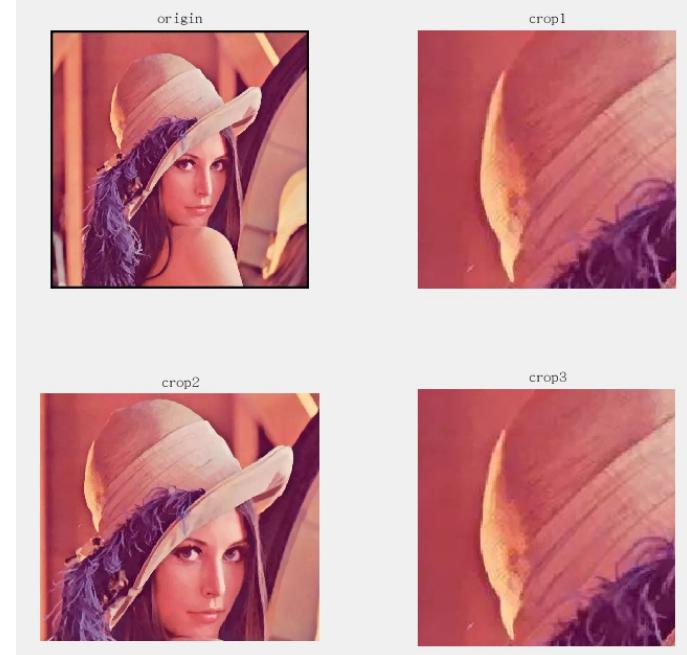
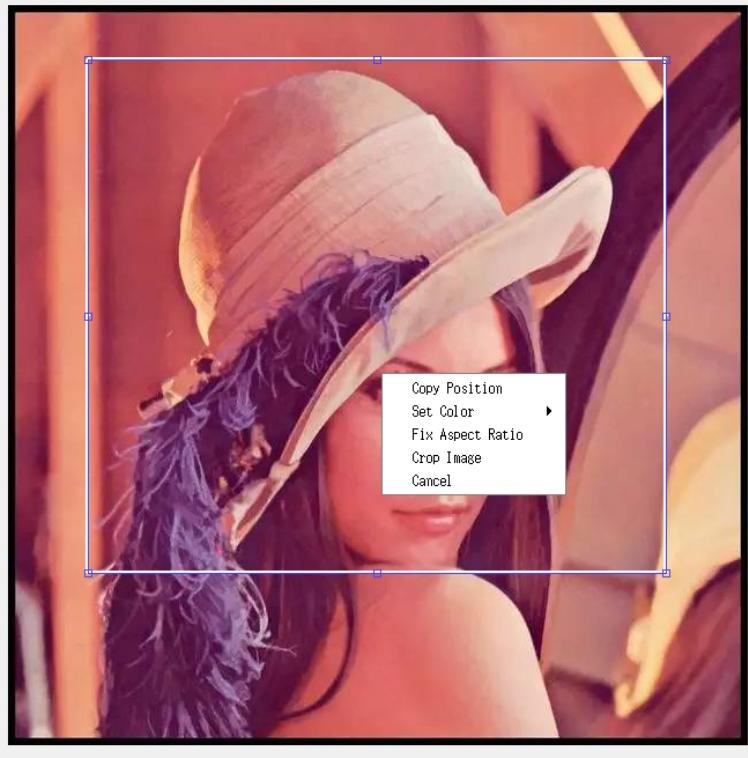
- ▶ $[I1,rect] = \text{imcrop}(I)$ 在图窗窗口中显示灰度图像、真彩色图像或二值图像 I ，并创建与该图像相关联的交互式裁剪图像工具。手动裁剪后，将裁剪后的图像返回给 $I1$ ，裁剪框返回给 $rect$ 。
- ▶ $I1 = \text{imcrop}(I,rect)$ 根据在裁剪矩形 $rect$ 中指定的位置和维度裁剪图像 I 。裁剪的图像包括输入图像中该矩形完全或部分包围的所有像素。 $rect$ 为 $[xmin\ ymin\ width\ height]$
- ▶ 对于索引图，加上 map 即可。



图像裁剪

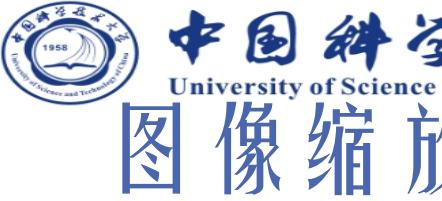
► 示例

```
1 %clear
2 - clc,clear,close all
3
4 - rgb1 = imread('/home/ubuntu/图片/2.jpeg')
5 - f = figure()
6 - rgb2 = imcrop(rgb1,[100,100,200,200])
7 - [rgb3,rect] = imcrop(rgb1)
8 - rgb4 = rgb1(100:300,100:300,:)
9
10 - subplot(2,2,1)
11 - imshow(rgb1)
12 - title('origin')
13 - subplot(2,2,2)
14 - imshow(rgb2)
15 - title('crop1')
16 - subplot(2,2,3)
17 - imshow(rgb3)
18 - title('crop2')
19 - subplot(2,2,4)
20 - imshow(rgb4)
21 - title('crop3')
```



rect =

69.5100 47.5100 499.9800 443.9800



中国科学技术大学
University of Science and Technology of China

图像缩放

- ▶ `imresize(I,scale)` 返回图像 J , 它是将 I 的长宽大小缩放 $scale$ 倍之后的图像。输入图像 I 可以是灰度图像、RGB 图像、二值图像或分类图像
- ▶ `imresize(I,[numrows numcols])` 返回图像 J , 其行数和列数由向量 $[numrows numcols]$ 指定。
- ▶ `imresize(____,method)` 指定使用的插值方法。

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - [h,w,c]=size(rgb)
6 - rgb1 = imresize(rgb,0.5)
7 - rgb2 = imresize(rgb,2)
8 - rgb3 = imresize(rgb,[h/2,w])
9
10 - f = figure()
11 - subplot(2,2,1)
12 - imshow(rgb)
13 - title('origin')
14 - subplot(2,2,2)
15 - imshow(rgb1)
16 - title('resizel')
17 - subplot(2,2,3)
18 - imshow(rgb2)
19 - title('resize2')
20 - subplot(2,2,4)
21 - imshow(rgb3)
22 - title('resize3')
```



- ▶ `imnoise(I,type,,parameter)`: 添加类型为type的噪声，噪声的参数为parameter。
 - ▶ `imnoise(I,'gaussian',m,var_gauss)`: 添加均值为m方差为var_gauss的高斯噪声，m默认值为0, var_gauss默认值为0.01。
 - ▶ `imnoise(I,'salt & pepper',d)` 添加椒盐噪声，其中d是噪声密度，默认为0.05。这会影响大约 $d * \text{numel}(I)$ 个像素。
 - ▶ `imnoise(I,'speckle',var_speckle)` 使用方程 $J = I + n * I$ 添加乘性噪声，其中n是均值为0、方差为var_speckle的均匀分布随机噪声，var_speckle默认为0.05。
 - ▶ `imnoise(I,'poisson')` 从数据中生成泊松噪声，而不是向数据中添加人为噪声



示例

```
1 %clear
2 clc,clear,close all
3
4 rgb1 = imread('/home/ubuntu/图片/2.jpeg')
5 f = figure()
6 rgb2 = imnoise(rgb1,'gaussian')
7 rgb3 = imnoise(rgb1,'salt & pepper')
8 rgb4 = imnoise(rgb1,'speckle')
9 rgb5 = imnoise(rgb1,'poisson')
10 subplot(3,2,1)
11 imshow(rgb1)
12 title('origin')
13 subplot(3,2,2)
14 imshow(rgb2)
15 title('gaussian')
16
17 subplot(3,2,3)
18 imshow(rgb3)
19 title('salt & pepper')
20 subplot(3,2,4)
21 imshow(rgb4)
22 title('speckle')
23 subplot(3,2,[5,6])
24 imshow(rgb5)
25 title('poisson')
26
```



- ▶ `filter2(H,X,shape)`: 根据矩阵 H 中的系数，对数据矩阵 X 应用有限脉冲响应滤波器。并根据 shape 返回滤波数据的子区。
 - ▶ 'same'，返回滤波数据的中心部分，大小与 X 相同。为默认值
 - ▶ 'full'，返回完整的二维滤波数据。
 - ▶ 'valid' - 仅返回计算的没有补零边缘的滤波数据部分。
- ▶ `conv2(A,B,shape)`: 与 fliter2 类似，只是将 A 作为主体。并且其卷积公式为

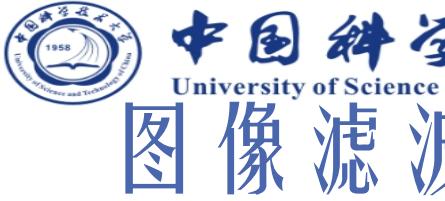
对于离散的二维变量 A 和 B，以下方程定义 A 和 B 的卷积：

$$C(j,k) = \sum_p \sum_q A(p,q) B(j-p+1, k-q+1)$$

p 和 q 会遍历所有可得到 $A(p,q)$ 和 $B(j-p+1, k-q+1)$ 的合法下标的值。

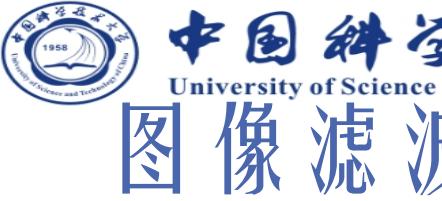
图像滤波

- ▶ `imfilter(A,h,options,...)` 根据一个或多个指定的选项) 使用多维滤波器 h 对多维数组 A 进行滤波。
- ▶ 边界
 - ▶ 数值标量， X 数组边界之外的输入数组值被赋予值 X 。如果未指定填充选项， 默认值为 0。
 - ▶ '`symmetric`' 数组边界之外的输入数组值是通过沿数组边界对数组进行镜面反射得到。
 - ▶ '`replicate`' 数组边界之外的输入数组值假定为等于最近的数组边界值。
 - ▶ '`circular`' 数组边界之外的输入数组值是通过隐式假设输入数组具有周期性来计算的。



图像滤波

- ▶ `imfilter(A,h,options,...)` 根据一个或多个指定的选项) 使用多维滤波器 h 对多维数组 A 进行滤波。
- ▶ 输出大小
 - ▶ '`'same`'输出数组与输入数组大小相同。这是未指定输出大小选项时的默认行为。
 - ▶ '`'full`'输出数组是完全滤波后的结果，因此比输入数组大。
- ▶ 相关性和卷积选项
 - ▶ '`'corr'``imfilter` 使用相关性执行多维滤波，这与 `filter2` 执行滤波的方式相同。当未指定相关性或卷积选项时，`imfilter` 使用相关性。
 - ▶ '`'conv'``imfilter` 使用卷积执行多维滤波。



图像滤波

▶ `fspecial(type)` 创建具有指定 `type` 的二维滤波器 `h`。一些滤波器类型具有可选的附加参数，如以下语法所示。`fspecial` 以相关性核形式返回 `h`，该形式适用于 `imfilter`。

- ▶ `fspecial('average',hsize)` 返回大小为 `hsize` 的平均值滤波器 `h`。
- ▶ `fspecial('disk',radius)` 在大小为 $2*radius+1$ 的方阵中返回圆形平均值滤波器 (pillbox)。
- ▶ `fspecial('prewitt')` 返回一个 3×3 滤波器，该滤波器通过逼近垂直梯度来强调水平边缘。要强调垂直边缘，请转置滤波器 `h`。

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

- ▶ `fspecial('sobel')` 返回一个 3×3 滤波器，该滤波器通过逼近垂直梯度来使用平滑效应强调水平边缘。要强调垂直边缘，请转置滤波器 `h'`。

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

▶ fspecial(type)

- ▶ $h = \text{fspecial}('laplacian', \alpha)$ 返回逼近二维拉普拉斯算子形状的 3×3 滤波器， α 控制拉普拉斯算子的形状。
- ▶ $h = \text{fspecial}('gaussian', hsize, \sigma)$ 返回大小为 $hsize$ 的旋转对称高斯低通滤波器，标准差为 σ 。不推荐。请改用 `imgaussfilt` 或 `imgaussfilt3`。
- ▶ $h = \text{fspecial}('log', hsize, \sigma)$ 返回大小为 $hsize$ 的旋转对称高斯拉普拉斯滤波器，标准差为 σ 。
- ▶ $h = \text{fspecial}('motion', len, \theta)$ 返回与图像卷积后逼近照相机线性运动的滤波器。 len 指定运动的长度， θ 以逆时针方向度数指定运动的角度。滤波器成为一个水平和垂直运动的向量。默认 len 是 9，默认 θ 是 0，对于 9 个像素的水平运动。



示例

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - il = rgb2gray(rgb)
6 - f = figure()
7 - fil1 = fspecial('sobel')
8 - fil2 = fspecial('prewit')
9 - fil3 = [1 1 1; 0 0 0;-1 -1 -1]
10 - i2=imfilter(il,fil1)
11 - i3 = uint8(filter2(fil1,il))
12 - i4 = uint8(conv2(fil1,il))
13 - i5 = imfilter(il,fil2)
14 - i6 = imfilter(il,fil3)
15
16 - subplot(3,2,1)
17 - imshow(il)
18 - title('origin')
19 - subplot(3,2,2)
20 - imshow(i2)
21 - title('imfilter')
22 - subplot(3,2,3)
23 - imshow(i3)
24 - title('filter2')
25 - subplot(3,2,4)
26 - imshow(i4)
27 - title('conv2')
28 - subplot(3,2,5)
29 - imshow(i5)
30 - title('fil2')
31 - subplot(3,2,6)
32 - imshow(i6)
33 - title('fil3')
```



origin



imfilter



filter2



conv2



fil2



fil3

- ▶ `medfilt2(I,[m n],padopt)` 执行中位数滤波. 其中每个输出像素包含输入图像中对应像素周围的 $m \times n$ 邻域中的中位数值, 默认为 3×3 。 `padopt` 控制 `medfilt2` 如何填充图像边界。
 - ▶ '`'zeros'`(默认值): 用 0 填充图像。
 - ▶ '`'symmetric'`: 在边界处对称延伸图像。
 - ▶ '`'indexed'` 如果 I 的类是 `double`, 则用 1 填充图像; 否则, 用 0 填充。



示例

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - i1 = rgb2gray(rgb)
6 - i2 = imnoise(i1,'salt & pepper',0.1)
7 - fil1 = fspecial('average',3)
8 - i3 = imfilter(i2,fil1)
9 - i4 = medfilt2(i2,[3,3])
10
11 - f = figure()
12 - subplot(2,2,1)
13 - imshow(i1)
14 - title('origin')
15 - subplot(2,2,2)
16 - imshow(i2)
17 - title('salt & pepper')
18 - subplot(2,2,3)
19 - imshow(i3)
20 - title('average')
21 - subplot(2,2,4)
22 - imshow(i4)
23 - title('medfil')
```



- ▶ `edge(I)` 返回二值图像 `BW`, 其中的值 1 对应于灰度或二值图像 `I` 中函数找到边缘的位置, 值 0 对应于其他位置。默认情况下, `edge` 使用 `Sobel` 边缘检测方法。
- ▶ `BW = edge(I,method,threshold,direction)` 指定要检测的边缘的方向。`Sobel` 和 `Prewitt` 方法可以检测垂直方向和/或水平方向的边缘。`Roberts` 方法可以检测与水平方向成 45 度角和/或 135 度角的边缘。仅当 `method` 是 '`Sobel`'、'`Prewitt`' 或 '`Roberts`' 时, 此语法才有效。
- ▶ `BW = edge(__,'nothinning')` 跳过边缘细化阶段, 这可以提高性能。仅当 `method` 是 '`Sobel`'、'`Prewitt`' 或 '`Roberts`' 时, 此语法才有效。
- ▶ `BW = edge(I,method,threshold,sigma)` 指定 `sigma`, 即滤波器的标准差。仅当 `method` 是 '`log`' 或 '`Canny`' 时, 此语法才有效。

边缘检测

- ▶ $BW = \text{edge}(I, \text{method}, \text{threshold}, h)$ 使用 'zerocross' 方法和您指定的滤波器 h 检测边缘。仅当 method 是 'zerocross' 时，此语法才有效。
- ▶ method
 - ▶ 'Sobel' 使用导数的 Sobel 逼近，通过寻找图像 I 的梯度最大的那些点来查找边缘。
 - ▶ 'Prewitt' 使用导数的 Prewitt 逼近，通过寻找 I 的梯度最大的那些点来查找边缘。
 - ▶ 'Roberts' 使用导数的 Roberts 逼近，通过寻找 I 的梯度最大的那些点来查找边缘。
 - ▶ 'log' 使用高斯拉普拉斯 (LoG) 滤波器对 I 进行滤波后，通过寻找过零点来查找边缘。
 - ▶ 'zerocross' 使用您指定的滤波器 h 对 I 进行滤波后，通过寻找过零点来查找边缘
 - ▶ 'Canny' 通过寻找 I 的梯度的局部最大值来查找边缘。edge 函数使用高斯滤波器的导数计算梯度。此方法使用双阈值来检测强边缘和弱边缘，如果弱边缘与强边缘连通，则将弱边缘包含到输出中。通过使用双阈值，Canny 方法相对其他方法不易受噪声干扰，更可能检测到真正的弱边缘。
 - ▶ 'approxcanny' 使用近似版 Canny 边缘检测算法查找边缘，该算法的执行速度较快，但检测不太精确。浮点图像应归一化到范围 $[0, 1]$ 。

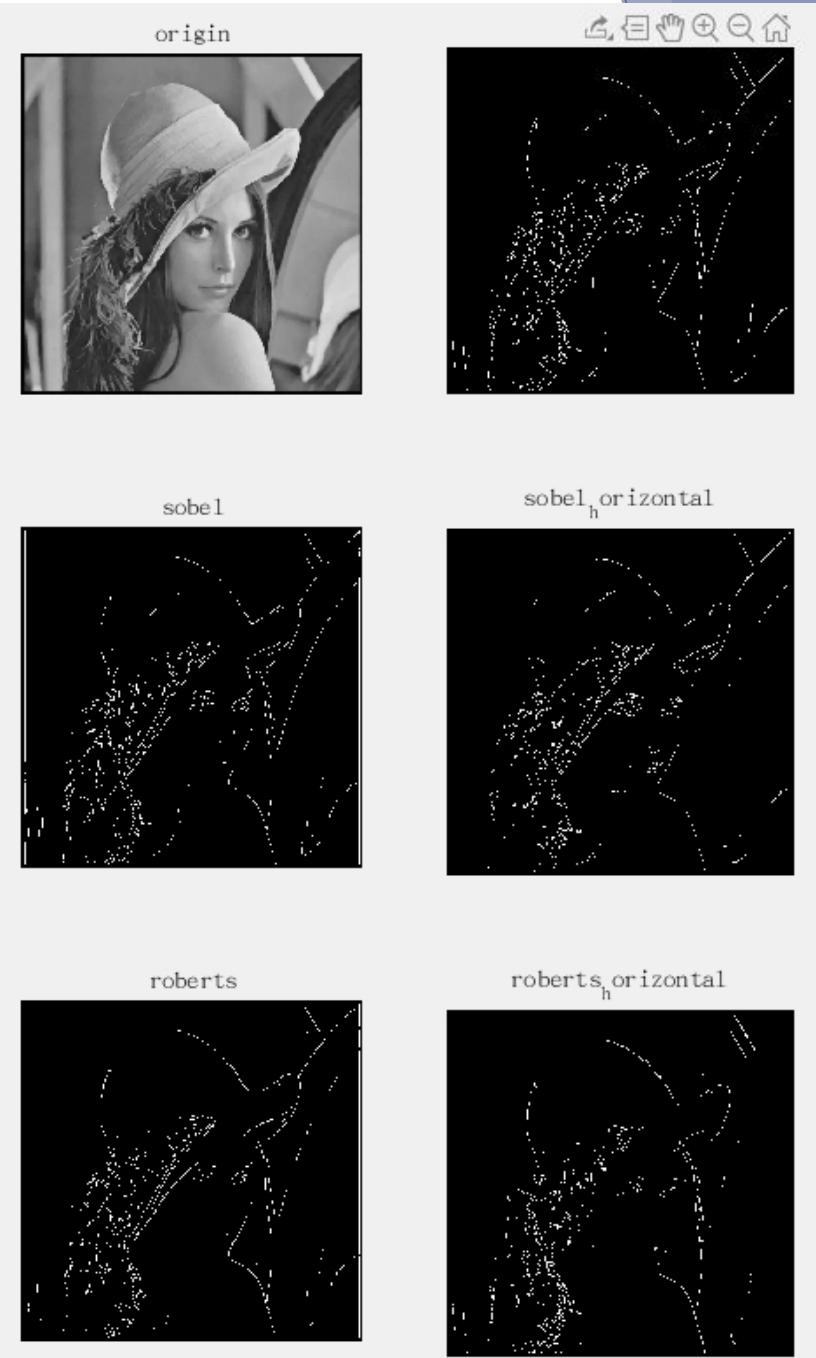
边缘检测

- ▶ direction - 要检测的边缘的方向
 - ▶ 'both' (默认)
 - ▶ 'horizontal' 水平方向, 对于roberts是135度
 - ▶ 'vertical'



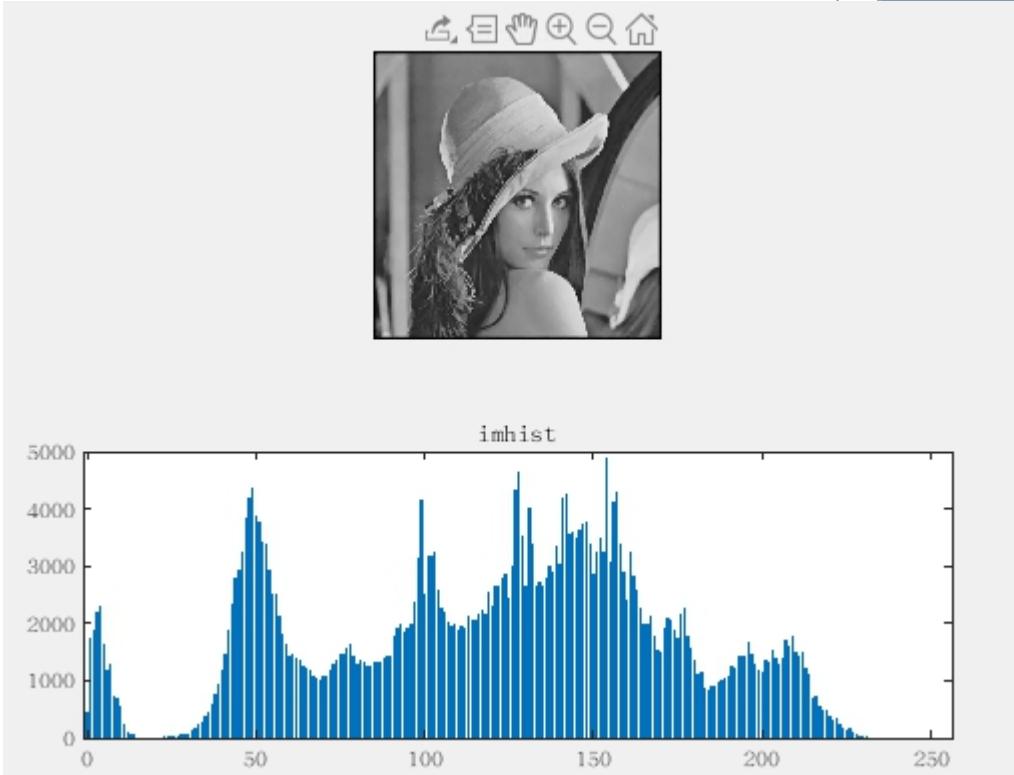
示例

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - i1 = rgb2gray(rgb)
6 - i2 = edge(i1)
7 - i3 = edge(i1,'Sobel')
8 - i4 = edge(i1,'Sobel','horizontal')
9 - i5 = edge(i1,'Roberts')
10 - i6 = edge(i1,'Roberts','horizontal')
11
12 - f = figure()
13 - subplot(3,2,1)
14 - imshow(i1)
15 - title('origin')
16 - subplot(3,2,2)
17 - imshow(i2)
18 - title('edge')
19 - subplot(3,2,3)
20 - imshow(i3)
21 - title('sobel')
22 - subplot(3,2,4)
23 - imshow(i4)
24 - title('sobel_horizontal')
25 - subplot(3,2,5)
26 - imshow(i5)
27 - title('roberts')
28 - subplot(3,2,6)
29 - imshow(i6)
30 - title('roberts_horizontal')
```



- ▶ imhist 函数
 - ▶ $[counts,binLocations] = imhist(I,n)$, 指定用于计算直方图的 bin 的数量 n, 灰度图默认为 256.
 - ▶ $[counts,binLocations] = imhist(X,map)$ 计算具有颜色图 map 的索引图像 X 的直方图。对于颜色图中的每个条目，直方图中都有一个对应的 bin。
- ▶ bar(X,Y) 在 X 指定的值的位置绘制数据序列 Y。X 和 Y 输入必须是大小相同的向量或矩阵。另外，X 可以是行或列向量，Y 必须是包含 $\text{length}(X)$ 行的矩阵。可以用来绘制直方图。

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - i1 = rgb2gray(rgb)
6 - [counts,bins] = imhist(i1)
7
8 - f = figure()
9 - subplot(2,1,1)
10 - imshow(i1)
11 - title('origin')
12 - subplot(2,1,2)
13 - bar(bins,counts)
14 - title('imhist')
15
```

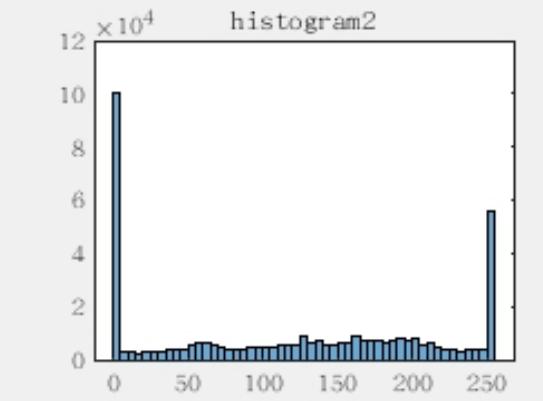
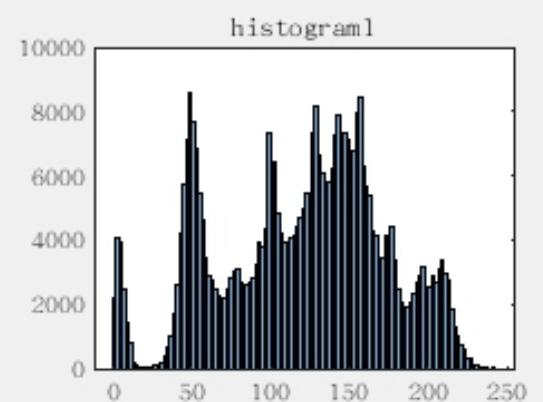


- ▶ $J = \text{imadjust}(I, [\text{low_in } \text{high_in}], [\text{low_out } \text{high_out}], \text{gamma})$
将 I 中的强度值映射到 J 中的新值，其中 gamma 指定描述 I 和 J 中的值之间关系的曲线形状。
 - ▶ 如果 gamma 小于 1，则 imadjust 会对映射加权，使之偏向更高（更亮）输出值。
 - ▶ 如果 gamma 大于 1，则 imadjust 会对映射加权，使之偏向更低（更暗）输出值。
 - ▶ 如果 gamma 是 1×3 向量，则 imadjust 会对每个颜色分量或通道分别应用不同的 gamma。
 - ▶ 如果省略该参数，则 gamma 取默认值 1（线性映射）。
- ▶ `histogram(I)`可以直接绘制直方图



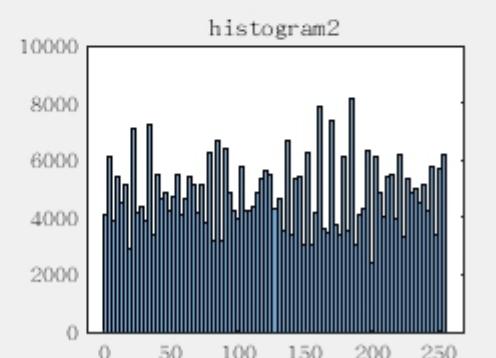
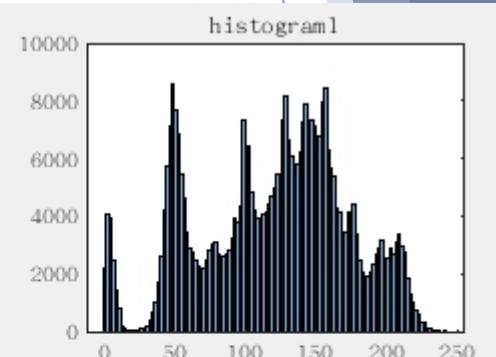
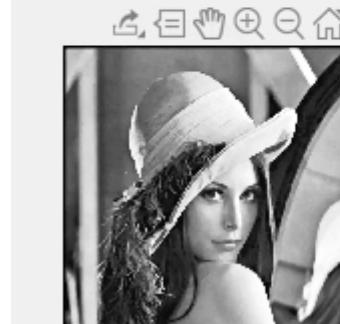
示例

```
1 %clear
2 - clc,clear,close all
3
4 - rgb = imread('/home/ubuntu/图片/2.jpeg')
5 - il = rgb2gray(rgb)
6 - i2 = imadjust(il,[0.3,0.7],[])
7
8 - f = figure()
9 - subplot(2,2,1)
10 - imshow(il)
11 - title('origin')
12 - subplot(2,2,3)
13 - imshow(i2)
14 - title('imadjust')
15 - subplot(2,2,2)
16 - histogram(il)
17 - title('histogram1')
18 - subplot(2,2,4)
19 - histogram(i2)
20 - title('histogram2')
```



- ▶ `histeq(I,hgram)` 变换灰度图像 I，以使输出灰度图像 J 具有 `length(hgram)` 个 bin 的直方图近似匹配目标直方图 hgram。
- ▶ `histeq(I,n)` 变换灰度图像 I，以使输出灰度图像 J 具有 n 个 bin 的直方图大致平坦。当 n 远小于 I 中的离散灰度级数时，J 的直方图更平坦。n 默认 64。
- ▶ 示例：

```
1 %clear
2 clc,clear,close all
3
4 rgb = imread('/home/ubuntu/图片/2.jpeg')
5 i1 = rgb2gray(rgb)
6 i2 = histeq(i1,256)
7
8 f = figure()
9 subplot(2,2,1)
10 imshow(i1)
11 title('origin')
12 subplot(2,2,2)
13 histogram(i1)
14 title('histogram1')
15 subplot(2,2,3)
16 imshow(i2)
17 title('histeq')
18 subplot(2,2,4)
19 histogram(i2)
20 title('histogram2')
```



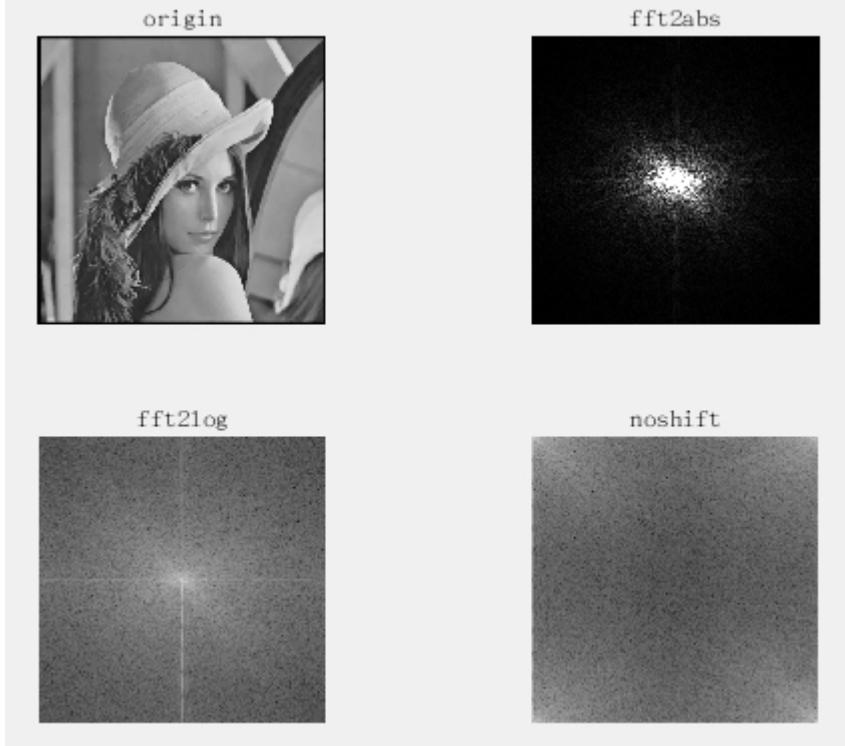
- ▶ $\text{fft}(X)$ 用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)。
 - ▶ 如果 X 是一个多维数组，则 $\text{fft}(X)$ 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回每个向量的傅里叶变换。
- ▶ $\text{fft}(X,n,\text{dim})$ 返回沿维度 dim 的傅里叶变换。例如，如果 X 是矩阵，则 $\text{fft}(X,n,2)$ 返回每行的 n 点傅里叶变换。
- ▶ $\text{fft2}(X,m,n)$ 将截断 X 或用尾随零填充 X ，以便在计算变换之前形成 $m \times n$ 矩阵。如果 X 是一个多维数组， fft2 将根据 m 和 n 决定 X 的前两个维度的形状。
- ▶ $\text{fft2}(X,m,n)$ 使用快速傅里叶变换算法返回矩阵的二维傅里叶变换，这等同于计算 $\text{fft}(\text{fft}(X,m,1),n,2)$ 。如果 X 是一个多维数组， fft2 将采用高于 2 的每个维度的二维变换。输出 Y 的大小与 X 相同。

- ▶ `fftshift(X)` 通过将零频分量移动到数组中心，重新排列傅里叶变换 X 。
 - ▶ 如果 X 是向量，则 `fftshift` 会将 X 的左右两半部分进行交换。
 - ▶ 如果 X 是矩阵，则 `fftshift` 会将 X 的第一象限与第三象限交换，将第二象限与第四象限交换。
 - ▶ 如果 X 是多维数组，则 `fftshift` 会沿每个维度交换 X 的半空间。
- ▶ `fftshift(X,dim)` 沿 X 的维度 dim 执行运算。例如，如果 X 是矩阵，其行表示多个一维变换，则 `fftshift(X,2)` 会将 X 的每一行的左右两半部分进行交换。



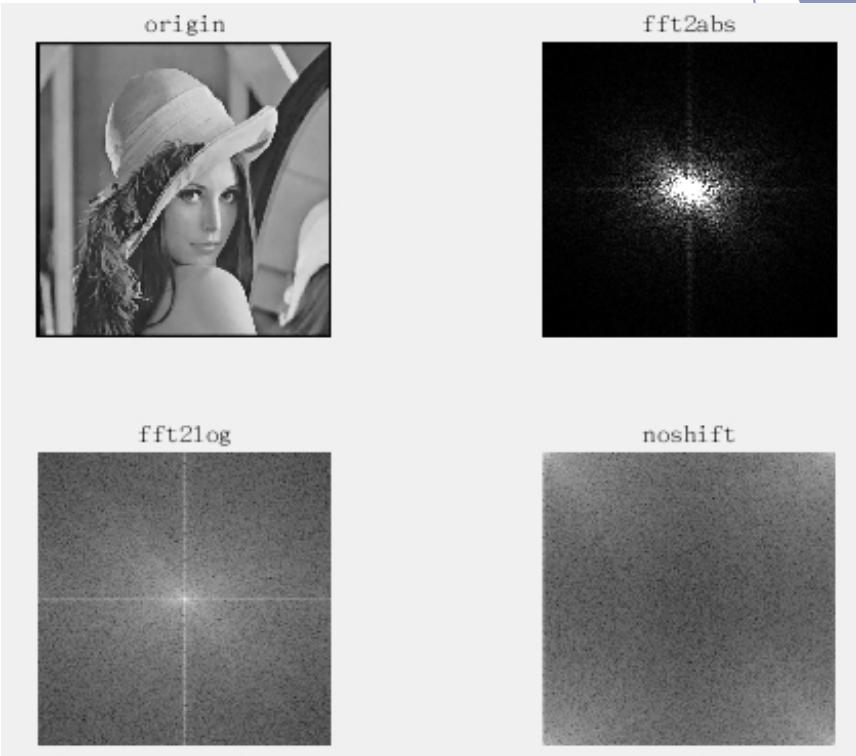
示例

```
1 %clear
2 clc,clear,close all
3
4 rgb = imread('/home/ubuntu/图片/2.jpeg')
5 i1 = rgb2gray(rgb)
6 i2 = fft2(i1)
7 i3 = fftshift(i2)
8 i4 = abs(i3)
9 i5 = log(i4 + 1)
10 i4 = uint16(i4)
11 i5 = uint8(i5)
12 max1 = double(max(i5(:)))/255
13 i5 = imadjust(i5,[0,max1],[])
14 i6 = uint8(log(abs(i2)+1))
15 max2 = double(max(i6(:)))/255
16 i6 = imadjust(i6,[0,max2])
17
18 f = figure()
19 subplot(2,2,1)
20 imshow(i1)
21 title('origin')
22 subplot(2,2,2)
23 imshow(i4)
24 title('fft2abs')
25 subplot(2,2,3)
26 imshow(i5)
27 title('fft2log')
28 subplot(2,2,4)
29 imshow(i6)
30 title('noshift')
```



示例

```
1 %clear
2 clc,clear,close all
3
4 -rgb = imread('/home/ubuntu/图片/2.jpeg')
5 -il = rgb2gray(rgb)
6 -i2 = fft(fft(il,1024,1),1024,2)
7 -i3 = fftshift(i2)
8 -i4 = abs(i3)
9 -i5 = log(i4 + 1)
10 -i4 = uint16(i4)
11 -i5 = uint8(i5)
12 -max1 = double(max(i5(:)))/255
13 -i5 = imadjust(i5,[0,max1],[])
14 -i6 = uint8(log(abs(i2)+1))
15 -max2 = double(max(i6(:)))/255
16 -i6 = imadjust(i6,[0,max2])
17
18 -f = figure()
19 -subplot(2,2,1)
20 -imshow(il)
21 -title('origin')
22 -subplot(2,2,2)
23 -imshow(i4)
24 -title('fft2abs')
25 -subplot(2,2,3)
26 -imshow(i5)
27 -title('fft2log')
28 -subplot(2,2,4)
29 -imshow(i6)
30 -title('noshift')
--
```





中国科学技术大学
University of Science and Technology of China

快速傅里叶逆变换

- ▶ `ifft(Y)` 使用快速傅里叶变换算法计算 Y 的逆离散傅里叶变换。 X 与 Y 的大小相同。
 - ▶ 如果 Y 是多维数组，则 `ifft(Y)` 将大小不等于 1 的第一个维度上的值视为向量，并返回每个向量的逆变换。
- ▶ `ifft(Y,n,dim)` 返回沿维度 dim 的傅里叶逆变换。例如，如果 Y 是矩阵，则 `ifft(Y,n,2)` 返回每一行的 n 点逆变换。
- ▶ $X = \text{ifft2}(Y,m,n)$ 在计算逆变换之前截断 Y 或用尾随零填充 Y ，以形成 $m \times n$ 矩阵。如果 Y 是一个多维数组，`ifft2` 将根据 m 和 n 决定 Y 的前两个维度的形状。
- ▶ 同理，`ifft2` 可以用 2 个 `ifft` 函数代替



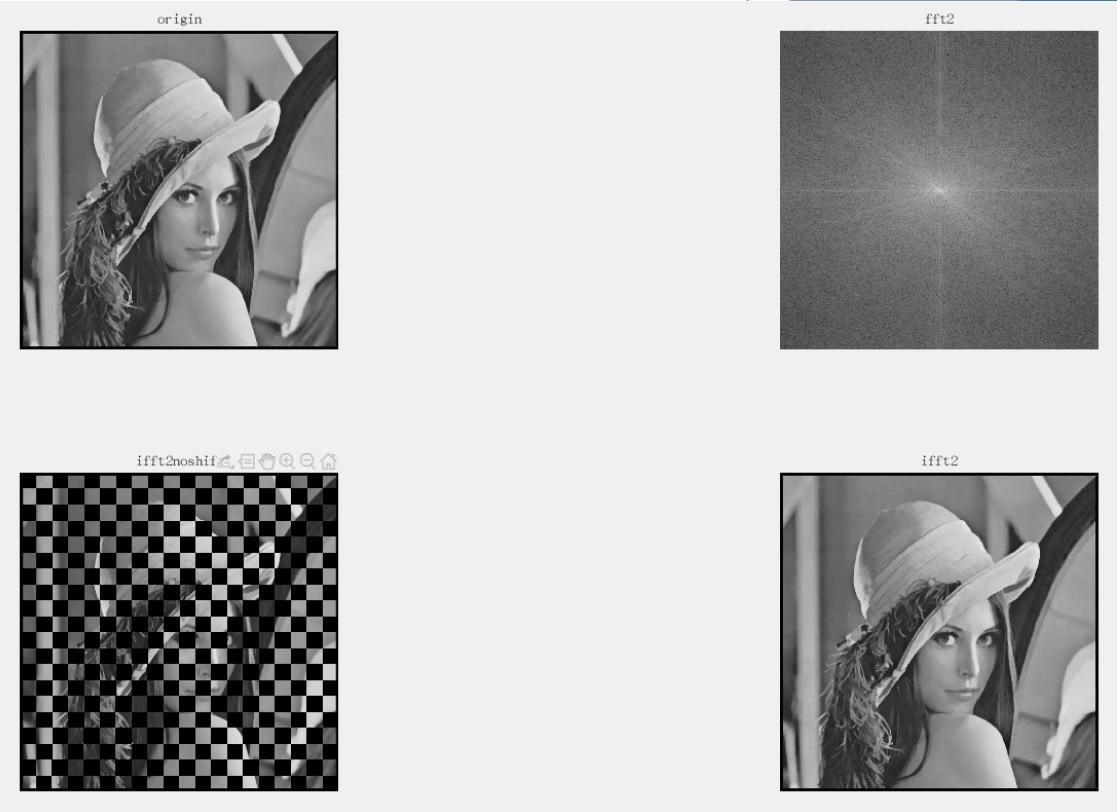
中国科学技术大学
University of Science and Technology of China

快速傅里叶反变换

- ▶ `ifftshift(Y)` 将进行过零频平移的傅里叶变换 Y 重新排列回原始变换输出的样子。换言之，`ifftshift` 就是撤销 `fftshift` 的结果。
 - ▶ 如果 Y 是向量，则 `ifftshift` 会将 Y 的左右两半部分进行交换。
 - ▶ 如果 Y 是矩阵，则 `ifftshift` 会将 Y 的第一象限与第三象限交换，将第二象限与第四象限交换。
 - ▶ 如果 Y 是多维数组，则 `ifftshift` 会沿每个维度交换 Y 的半空间。
- ▶ `ifftshift(Y,dim)` 沿 Y 的维度 dim 执行运算。例如，如果 Y 是矩阵，其行表示多个一维变换，则 `ifftshift(Y,2)` 会将 Y 的每一行的左右两半部分进行交换。

示例

```
1 %clear
2 clc,clear,close all
3
4 -rgb = imread('/home/ubuntu/图片/2.jpeg')
5 -i1 = rgb2gray(rgb)
6 -i2 = fft2(i1)
7 -i3 = fftshift(i2)
8 -i4 = abs(i3)
9 -i5 = log(i4 + 1)
10 -i5 = uint8(i5)
11 -max1 = double(max(i5(:)))/255
12 -i5 = imadjust(i5,[0,max1],[])
13 -i6 = uint8(ifft2(i3))
14 -i7 = uint8(ifft2(ifftshift(i3)))
15
16 -f = figure()
17 -subplot(2,2,1)
18 -imshow(i1)
19 -title('origin')
20 -subplot(2,2,2)
21 -imshow(i5)
22 -title('fft2')
23 -subplot(2,2,3)
24 -imshow(i6)
25 -title('ifft2noshift')
26 -subplot(2,2,4)
27 -imshow(i7)
28 -title('ifft2')
```



- ▶ `dct2(A,m,n)` 和 `dct2(A,[m n])` 用 0 对矩阵 A 进行填充，使其大小为 $m \times n$ 。如果 m 或 n 小于 A 的对应维度，则 `dct2` 在变换前对 A 进行裁切。
- ▶ `idct2(A,m,n)` 和 `dct2(A,[m n])` 用 0 对矩阵 A 进行填充，使其大小为 $m \times n$ 。如果 m 或 n 小于 A 的对应维度，则 `2dct2` 在变换前对 A 进行裁切。

```
1 %clear
2 clc,clear,close all
3
4 rgb = imread('/home/ubuntu/图片/2.jpeg')
5 il = rgb2gray(rgb)
6 i2 = dct2(il)
7 i3 = log(abs(i2)+1)
8 i3 = uint8(i3)
9 max1 = double(max(i3(:)))/255
10 i4 = imadjust(i3,[0,max1],[])
11 i5 = uint8(idct2(i2))
12
13 f = figure()
14 subplot(2,2,1)
15 imshow(il)
16 title('origin')
17 subplot(2,2,2)
18 imshow(i4)
19 title('dct2')
20 subplot(2,2,3)
21 imshow(i5)
22 title('idct2')
```

