MATLAB 简介

2017.10.12

崔哲

phy_cuiz10@163.com

目录

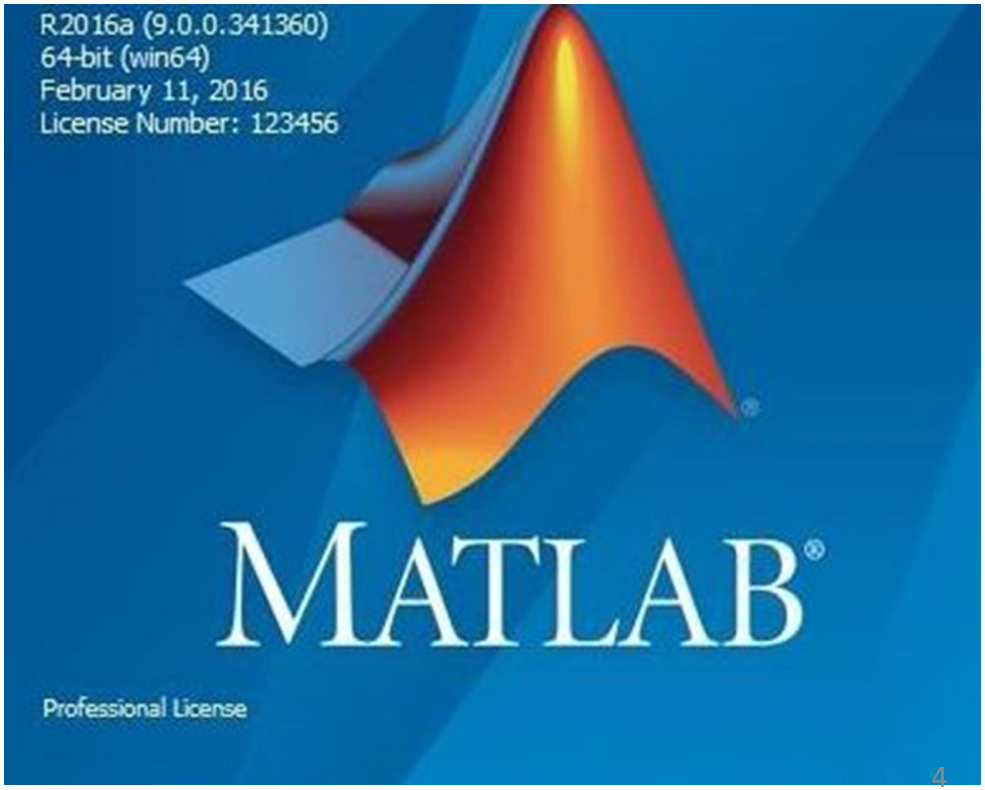
- MATLAB简介：安装/使用/调试
- 基本操作：变量/脚本/函数
- 图像处理：基础/滤波/分割
- 其他功能

简介

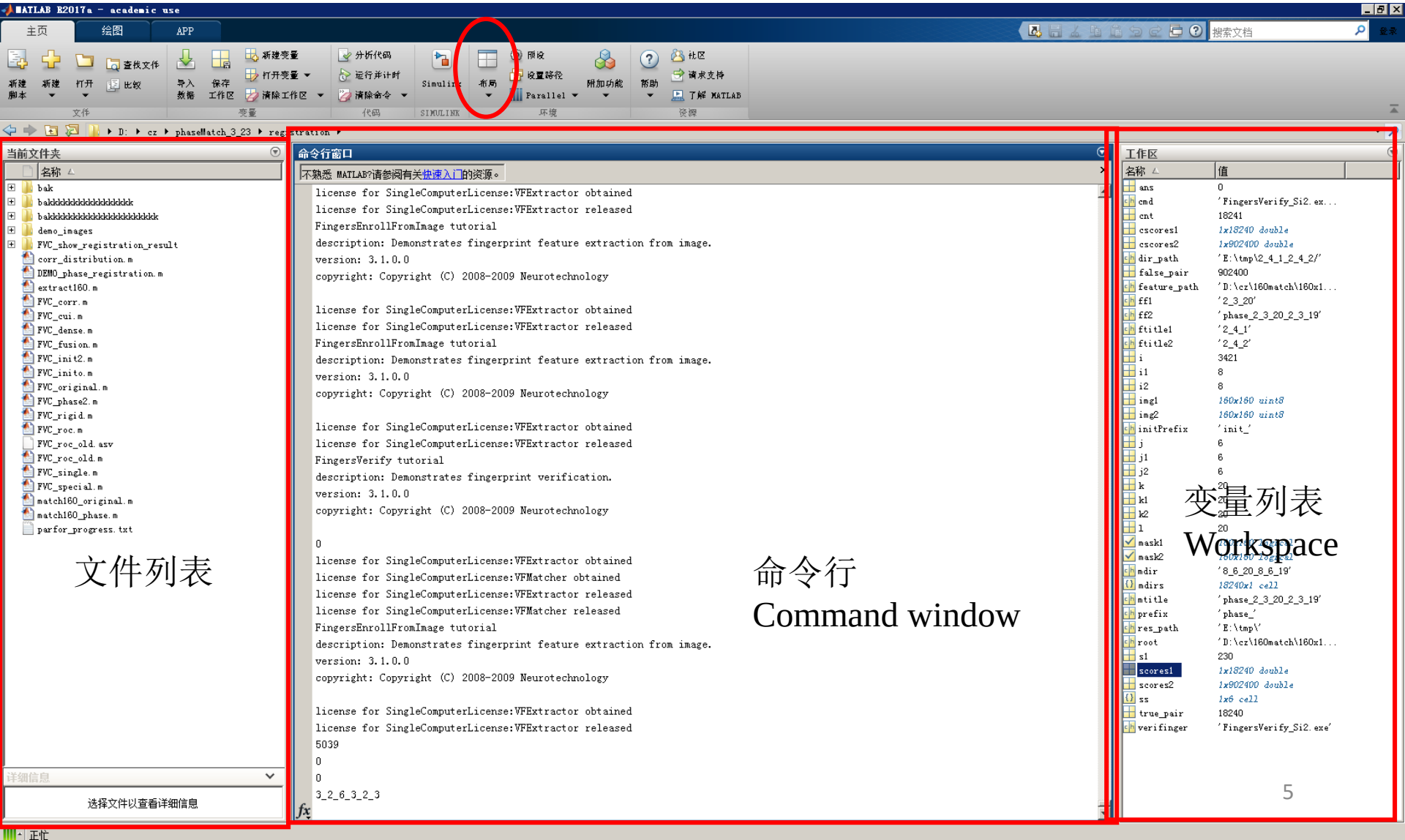
- MATLAB（matrix laboratory），是美国MathWorks公司出品的一款商业数学软件，包含MATLAB和Simulink两个部分，其主要优势在于强大的矩阵运算和数值计算功能。目前广泛应用于算法研发，数据可视化，仿真等领域；
- 高效的矩阵运算和数值计算功能；
- 强大的可视化（2D，3D）功能；
- 人机交互功能
- 表达语言自然化，接近数学表达式；

简介–安装

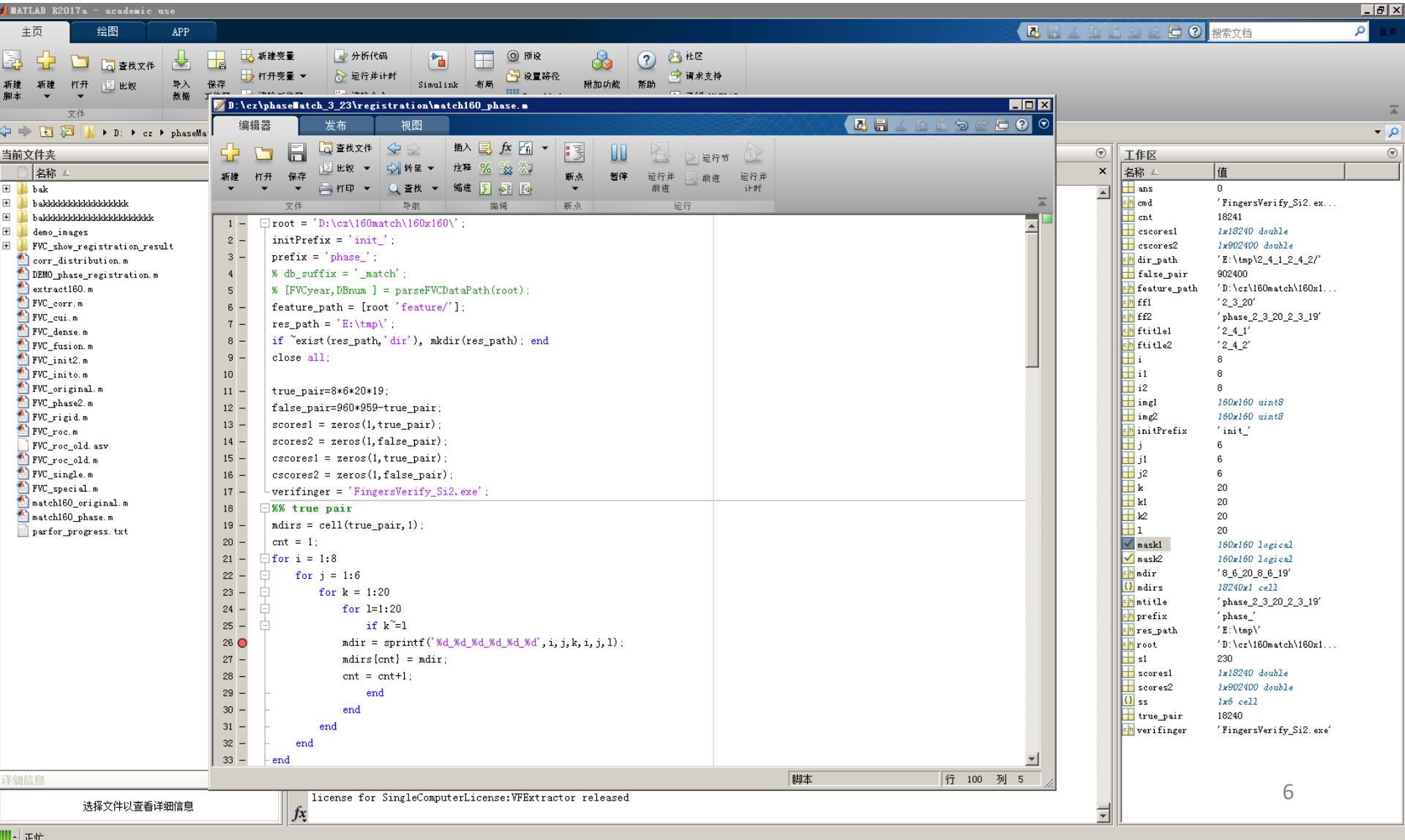
- 版本越新越好，功能越全(最新2017a)
- 学校免费提供教育版win/linux/mac
- 正版¥15000/3000



简介-使用


MATLAB R2017a - academic use
主页
绘图
APP
布局
当前文件夹
文件列表
选择文件以查看详细信息
命令行窗口
不熟悉 MATLAB?请参阅有关快速入门的资源。
license for SingleComputerLicense:VFExtractor obtained
license for SingleComputerLicense:VFExtractor released
FingersEnrollFromImage tutorial
description: Demonstrates fingerprint feature extraction from image.
version: 3.1.0.0
copyright: Copyright (C) 2008-2009 Neurotechnology
license for SingleComputerLicense:VFExtractor obtained
license for SingleComputerLicense:VFExtractor released
FingersEnrollFromImage tutorial
description: Demonstrates fingerprint feature extraction from image.
version: 3.1.0.0
copyright: Copyright (C) 2008-2009 Neurotechnology
license for SingleComputerLicense:VFExtractor obtained
license for SingleComputerLicense:VFExtractor released
FingersVerify tutorial
description: Demonstrates fingerprint verification.
version: 3.1.0.0
copyright: Copyright (C) 2008-2009 Neurotechnology
0
license for SingleComputerLicense:VFExtractor obtained
license for SingleComputerLicense:VFMatcher obtained
license for SingleComputerLicense:VFExtractor released
license for SingleComputerLicense:VFMatcher released
FingersEnrollFromImage tutorial
description: Demonstrates fingerprint feature extraction from image.
version: 3.1.0.0
copyright: Copyright (C) 2008-2009 Neurotechnology
license for SingleComputerLicense:VFExtractor obtained
license for SingleComputerLicense:VFExtractor released
5039
0
0
3_2_6_3_2_3
命令行
Command window
工作区
变量列表
Workspace
正忙

简介-调试


MATLAB R2017a - academic use
主页
绘图
APP
D:\cz\phaseMatch_3_23\registration\match160_phase.m
编辑器
发布
视图
root = 'D:\cz\160match\160x160\';
initPrefix = 'init_';
prefix = 'phase_';
% db_suffix = '_match';
% [FVCyear,DBnum] = parseFVCDataPath(root);
feature_path = [root 'feature/'];
res_path = 'E:\tmp\';
if ~exist(res_path,'dir'), mkdir(res_path); end
close all;
true_pair=8*6*20*19;
false_pair=960*959-true_pair;
scores1 = zeros(1,true_pair);
scores2 = zeros(1,false_pair);
cscores1 = zeros(1,true_pair);
cscores2 = zeros(1,false_pair);
verifinger = 'FingersVerify_Si2.exe';
%% true pair
mdirs = cell(true_pair,1);
cnt = 1;
for i = 1:8
for j = 1:6
for k = 1:20
for l=1:20
if k~=1
mdir = sprintf('%d_%d_%d_%d_%d_%d',i,j,k,i,j,l);
mdirs{cnt} = mdir;
cnt = cnt+1;
end
end
end
end
end
脚本
行 100 列 5
当前文件夹
名称
bak
FVC_show_registration_result
corr_distribution.m
DEMO_phase_registration.m
extract160.m
FVC_corr.m
FVC_cui.m
FVC_dense.m
FVC_fusion.m
FVC_init2.m
FVC_inito.m
FVC_original.m
FVC_phase2.m
FVC_rigid.m
FVC_roc.m
FVC_roc_old.asv
FVC_roc_old.m
FVC_single.m
FVC_special.m
match160_original.m
match160_phase.m
parfor_progress.txt
详细信息
选择文件以查看详细信息
工作区
名称
值
ans 0
cmd 'FingersVerify_Si2.ex...
cnt 18241
cscores1 1x18240 double
cscores2 1x902400 double
dir_path 'E:\tmp\2_4_1_2_4_2/'
false_pair 902400
feature_path 'D:\cz\160match\160x1...
ff1 '2_3_20'
ff2 'phase_2_3_20_2_3_19'
ftitle1 '2_4_1'
ftitle2 '2_4_2'
i 8
i1 8
i2 8
img1 160x160 uint8
img2 160x160 uint8
initPrefix 'init_'
j 6
j1 6
j2 6
k 20
k1 20
k2 20
l 20
mask1 160x160 logical
mask2 160x160 logical
mdir '8_6_20_8_6_19'
mdirs 18240x1 cell
mtitle 'phase_2_3_20_2_3_19'
prefix 'phase_'
res_path 'E:\tmp\'
root 'D:\cz\160match\160x1...
s1 230
scores1 1x18240 double
scores2 1x902400 double
ss 1x6 cell
true_pair 18240
verifinger 'FingersVerify_Si2.exe'
license for SingleComputerLicense:VFExtractor released

目录

- MATLAB简介：安装/使用/调试
- **基本操作**：变量/脚本/函数
- 图像处理：基础/滤波/分割
- 其他功能

基本操作–变量

- 数值矩阵/向量/数/逻辑(二值)
- 字符/字符串
- struct
- cell

基本操作-矩阵

- MATLAB的核心：矩阵
- MATLAB的设计初衷就是处理矩阵和线性代数问题
- 例如：对（二维）矩阵的定义和元素存取

矩阵$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

```
>> A = [1,2,3;4,5,6;7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9
```

基本操作-矩阵

- MATLAB提供的矩阵基本操作非常丰富，这里仅举图像处理中较常用的一些最基本例子

```
B =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> size(B)

ans =

     3     4
```

矩阵规模

```
>> C = B'

C =

     1     5     9
     2     6    10
     3     7    11
     4     8    12
```

矩阵转置

```
>> reshape(B, [2,6])

ans =

     1     9     6     3    11     8
     5     2    10     7     4    12
```

矩阵重排

MATLAB中矩阵元素按列排列!!!

基本操作-矩阵

- 对于矩阵A，典型的索引方式包括：

$A(i,j)$：获得矩阵A第i行j列的元素；分清楚下标与坐标

$A(i,:)$：获得矩阵A的第i行向量；

$A(:,j)$：获得矩阵A的第j列向量；

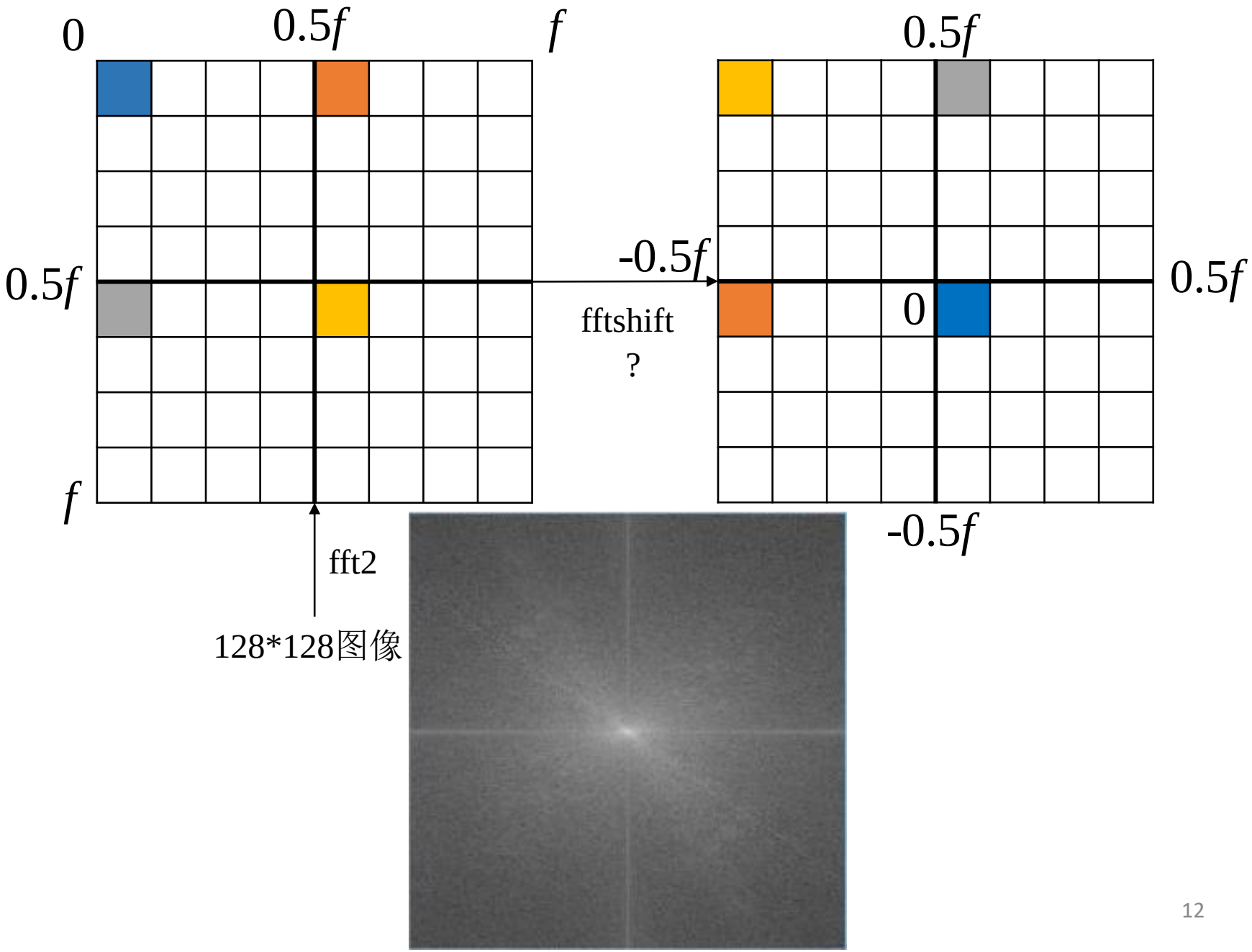
$A(i_1:i_2,:)$：获得矩阵A的第i_1至i_2行组成的子矩阵；

$A(:,j_1:j_2)$：获得矩阵A的第j_1至j_2列组成的子矩阵；(end)

$A(i_1:i_2,j_1:j_2)$：选择矩阵A的第i_1至i_2行，再选择其中的第j_1至j_2列组成的子矩阵；

$V=A(:)$：将矩阵A按逐列排列的方式“拉直”为向量V；

$A(ind)$：将A拉直为向量V，返回V的第ind个元素。


0
0.5f
f
0.5f
f
-0.5f
0.5f
0
fftshift
?
0.5f
-0.5f
fft2
128*128图像

基本操作-矩阵运算

- 分清矩阵运算与元素运算
- *, .*; ^2, .^2; /, ./
- /　\　^-1　inv()　pinv()
- sin,cos,exp,log等均是对元素运算(element-wise)
- tr(), det(), diag(), eig()
- 可用[A,B]或[A;B]来合并矩阵
- A(:,1)=[]来删除矩阵的某一行或某一列

基本操作-逻辑运算

- 针对逻辑(二值矩阵)0/1
- &, | , ~, xor()
- 逻辑矩阵作为索引使用
- A=[8,6,9,3,4,1]; B=[1,0,1,0,0,0]; C=[1,3];
- A(B)=A(C)=[8,9]
- A(A>=8)=[8,9]

基本操作-高维数组

- 通常我们所说的“矩阵”，实际上是指二维数组的概念，而“数组”则是C，FORTRAN等语言中对数据的连续存储和获取的结构，但是“数组”本质上是对矩阵概念的进一步抽象；
- 在很多问题中，我们需要考虑的是高维的数据，例如对图像处理而言，RGB图像是典型的三维数组，其他多维数组应用例如多元统计分析，数据采集，复杂姿态参数估计问题等。
- MATLAB中提供了对任意维数组的基本操作，K维数组，是指具有行、列、页……等K个维度的“超立方体”
- 一个典型的K维数组可以表示为A_D

$$D = d_1 \times d_2 \times \cdots \times d_K$$

基本操作–高维数组

- 例：声明高维数组，及其基本操作

```
Array5 = rand(2,3,4,5,6); %声明一个5-D数组
Nd = ndims(Array5);       %数组维数
S = size(Array5);         %数组各个维度的大小（规模）
S3d = size(Array5,3);     %数组指定维度的大小
L = length(Array5);       %数组各维度中的最长规模
Ne = numel(Array5);       %数组元素总数
```

基本操作-高维数组

- 对于每一个矩阵A，MATLAB本质上都视其为向量，这是很多操作（例如reshape）的基础，因此$A(i,j)$和$A(ind)$这两种索引方式本质上是等同的，对于二维矩阵，两种索引的对应关系很容易计算，而对高维的矩阵则比较麻烦，因此，MATLAB提供了两个转换的函数$ind2sub()$和$sub2ind()$来进行这种转换
- 例：对2D矩阵

IND = [3,4,5,6]; %待转换的index共4个，分别为3,4,5,6

sz = [3,3];　　%矩阵的size是3×3

[I,J] = ind2sub(sz,IND)

- 对高维矩阵

IND = [1,10,50];%待转换的index共3个，分别为1,10,50

sz = [3,3,4,5];　%矩阵的size是3×3×4×5

[I,J,K,L] = ind2sub(sz,IND)

基本操作–高维数组

- 反之，*sub*2*ind*()函数，将给定的矩阵各维度分量下标，转换为索引。
- 例如：

```
A = [1,2,3;4,5,6];
sub2ind(size(A),2,2);      %获取A(2,2)对应的索引
```

基本操作-高维数组

- MATLAB对矩阵元素的操作：自然
- 相比于C++等语言，MATLAB对于矩阵元素的读写是非常容易的，而这是最基本的操作;
- 对于高维矩阵具有很好的支持，这一点其他一些语言不能保证

$$A = zeros(3,4,4,3,2);$$

- 严谨性换来较高的灵活性，对于某些操作，C++等强类型语言会带来不便；以牺牲严谨性为代价，数组越界，矩阵维度等需要自己检查

	C++(基于openCV2)	Java(基于jama包)	MATLAB
读取RGB图像左上角蓝色分量	int b = img.at<Vec3b>(0,0)[0];	int b = img.get(0,0);	b = img(1,1,1);
获取矩阵行列信息	int r = img.rows; Int c = img.cols;	int r=img.getRowDimension(); int c=img.getColDimension();	r = size(img,1); c = size(img,2);
矩阵乘法	cv::Mat C = A * B;	Matrix C = A.times(B);	C = A * B ;
矩阵转置	cv::Mat imgT = img.t();	Matrix imgT = img.transpose();	imgT = img’;
矩阵复制	cv::Mat B(A);	Matrix B = (Matrix)(A.clone());	B = A ;
	适用于复杂图像处理和应用开发，不适用于快速的实验	不是完整的线性代数库，例如不包括系数矩阵，不包括复矩阵等	方便，简洁，严格性有所欠缺

基本操作-字符串

- MATLAB对字符串的声明很简单，由单引号及其内部的字符内容声明：
- >>a = ‘hello, MATLAB world!’
- >>length(a) %查看字符串长度
- >>a(1) = ‘H’ %通过索引读写字符串
- >>b = a %字符串赋值
- >>c = [a,b] %字符串连接
- >>a == b %字符串比较
- 通过比较、连接等操作，我们可以看出MATLAB对字符串同样视为数组，每个元素是一个字符

基本操作-字符串

- str2num(‘100’)=100;
- num2str(100)=‘100’;
- ASCII码：char(65)=‘A’;

例：实现按首字母排序

- eval()函数：将字符串按函数语句执行

例：v1=‘A’, eval([v1 ‘==0’])等价于A==0

基本操作-struct/cell

- struct 结构体
- cell 元胞
- 可以将不同类型的数据组合在一起，非常灵活

```
>> c = rand(5,5);   %创建随机矩阵c
>> f = @(x) x.^2;   %创建函数句柄f
>> x = 1 + 3j;      %创建复数x
>> G = {c, f, x};   %将三者组合构成单元数组
>> class(G)         %class，重要命令，查看对象的类型

ans =

cell
```

```
>> G{1}

ans =

    0.8147    0.0975    0.1576    0.
    0.9058    0.2785    0.9706    0.
    0.1270    0.5469    0.9572    0.
    0.9134    0.9575    0.4854    0.
    0.6324    0.9649    0.8003    0.
```

向量化操作

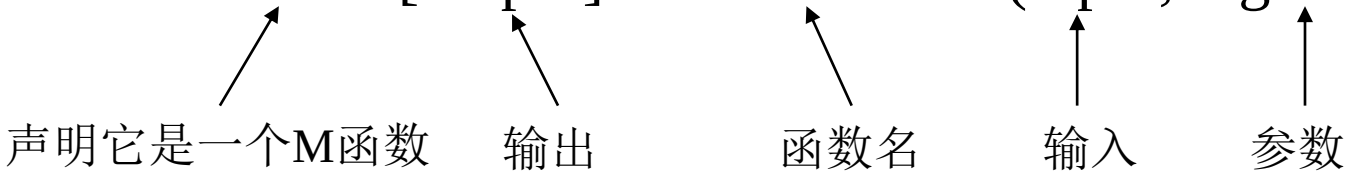
- 相比于C，MATLAB为人所“诟病”的问题之一在于其效率，但是这种认识是片面的，C的基本类型是数值，而MATLAB基本类型就是向量，因此用MATLAB应该抛弃“标量化”的思维方式。
- 在MATLAB中，for循环是一个非常耗时的过程！
- 因此，尽可能将标量的for循环替代为向量化操作，是一个很重要的并行提高MATLAB效率的技巧。

```
for i=1:N
        %do something
end
```

BAD!

基本操作-函数

- 和C语言一样，函数是MATLAB编程的核心概念之一。
- 本质上，MATLAB是对我们输入的命令进行解释，然而，对于较大规模的问题，将其完全按照一条条命令来组织是很繁琐的，也是很不好的编程习惯，因此我们希望将功能进行封装，在需要时只需调用其接口。
- MATLAB中，对函数的基本定义方式是：
- function[output]=functionName(input, argument)



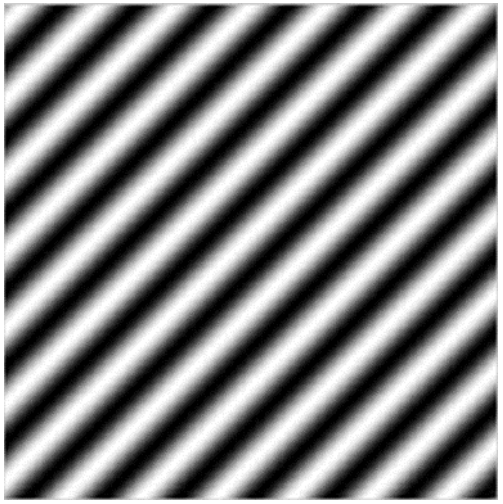
- 它的语法很灵活，允许多个输入/输出

基本操作-函数

例如，我们基于如下公式创建合成图像观察正弦函数的周期性：

$f(x,y)=Asin(u\_0\ x+v\_0\ y)$

可以创建如下的m函数：



```
function f = twodsin(A, u0, v0, M, N)
% twodsin 函数名
% A, u0, v0, M, N 输入参数
% f 函数的输出
f = zeros(M, N);          %预分配输出的数组可以提高效率
for c = 1:N
        v0y = v0 * (c - 1);
        for r = 1:M
                u0x = u0 * (r - 1);
                f(r,c) = A * sin(u0x + v0y);          %计算f的各个分量
        end
end
end
```

基本操作-函数

- 将函数保存在同名的m文件中，可以作为一个“函数文件”，直接在MATLAB命令窗口中通过函数名来调用。

```
function  f = twodsin(…)
% do something
end
```

保存在

>>f = twodsin(…)

基本操作-脚本

- 说明，m文件一般分为两种，除了对M函数的定义外，还可以作为若干命令的组合的“脚本”，例如：下面的函数实现了一个简单版本的logistic回归

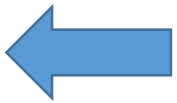
```
%随机构造一个数据集
x = rand(200,100);                          %100个200维向量
y = randi([0,1],1,100);          %100个类别标签
x(:,find(y==1)) = x(:,find(y==1)) + 0.1;
w = zeros(1,200);                           %初始化权值
for iter = 1:10000                                      %迭代次数
        g = w*x;
        pai = 1./(1+exp(-g));
        L = zeros(1,1000);
        for t = 1:100
                if y(t) == 1
                        L(t) = -log(pai(t));
                else
                        L(t) = -log(1-pai(t));
                end
        end
        loss = sum(L);                                  %计算误差损失
        disp(['for iteratoin: ',num2str(iter),' , loss: ',num2str(loss)])
        K = -y.*(1-pai) + (1-y).*pai;
        grad = sum(x*diag(K),2);
        w = w - grad‘*0.001;                            %负梯度下降
end
```

基本操作-函数

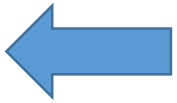
- 同一个m文件中可以定义多个M函数，其中与该文件同名的可以直接通过文件名调用
- 考虑前面的例子，计算2D的正弦，我们考虑分别从标量循环，和向量化的角度来给出计算方式，比较计算结果和时间。
- 建立m文件testTwoSin.m，加入如下函数：

```
function testTwoSin()
    clear all;
    disp(‘for-loop style method:’)                    %在命令窗口中显示提示消息
    tic;                                              %计时
    f1 = twodsin1(1, 1/(4*pi), 1/(4*pi), 2048, 2048);
                                                      %用方法1构造2048× 2048图像
    toc;
    disp('meshgrid style method:')
    tic;
    f2 = twodsin2(1, 1/(4*pi), 1/(4*pi), 2048, 2048);
                                                      %用方法2构造2048× 2048图像
    toc;
    subplot(1,2,1)      %在子图中分别显示
    imshow(f1, [])
    subplot(1,2,2)
    imshow(f2, [])
end
```

```
%与上一页ppt是同一个m文件中
%标量化循环
function f = twodsin1(A, u0, v0, M, N)
        f = zeros(M, N);
        for c = 1:N
                v0y = v0 * (c - 1);
                for r = 1:M
                        u0x = u0 * (r - 1);
                        f(r,c) = A * sin(u0x + v0y);
                end
        end
end
%向量化数组
function f = twodsin2(A, u0, v0, M, N)
        r = 0:M-1;
        c = 0:N-1;
        [C, R] = meshgrid(c,r);
        f = A * sin(u0*R + v0*C);
end
```



直观，易于理解

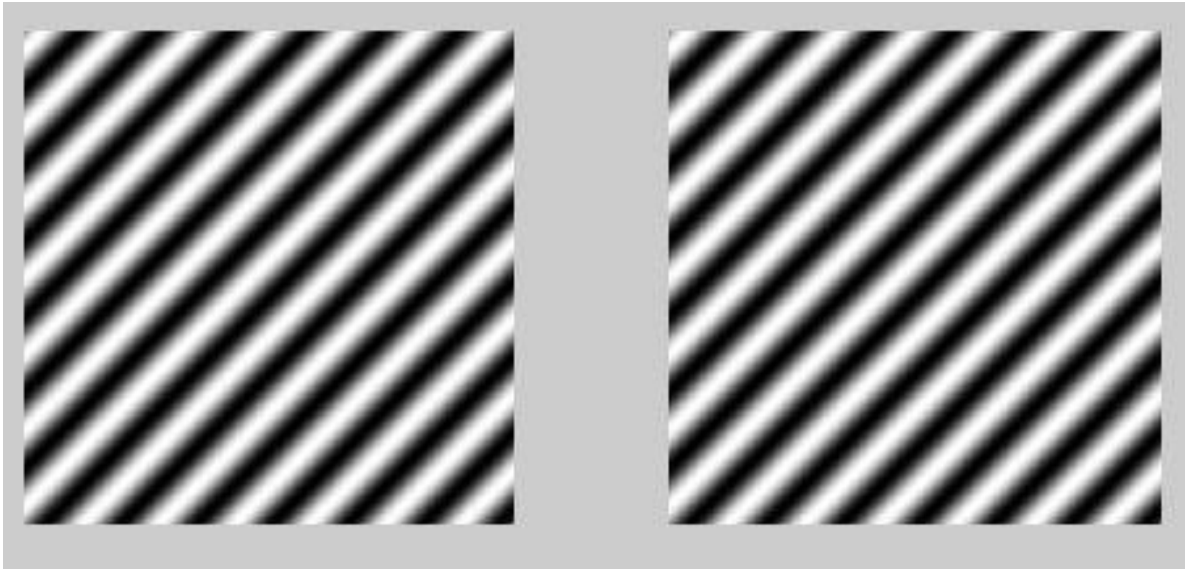


矩阵表述简洁

基本操作-函数

- 调用testTwoSin()函数，对比其时间：

>>testTwoSin()

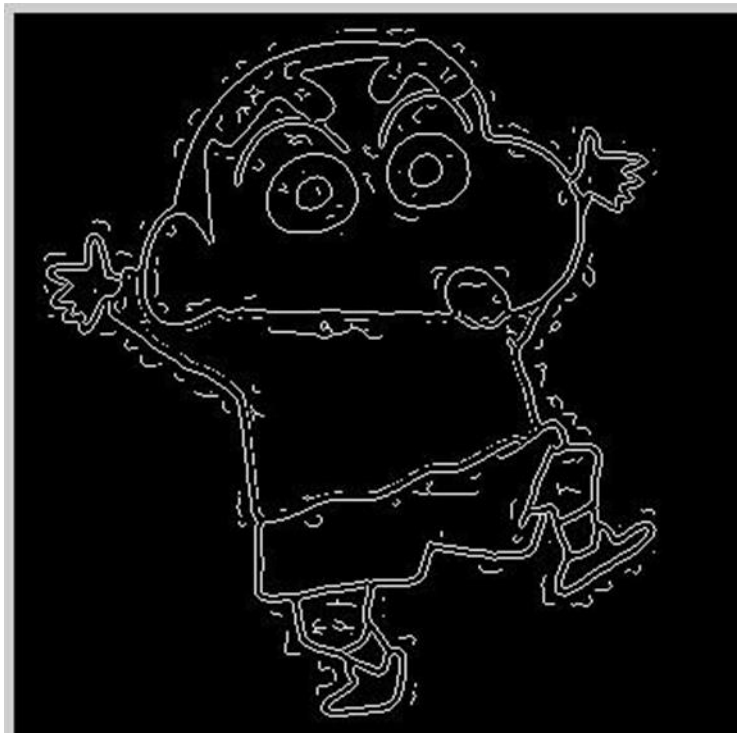


目录

- MATLAB简介：安装/使用/调试
- 基本操作：变量/脚本/函数
- **图像处理**：基础/滤波/分割
- 其他功能

图像处理

- 图像处理工具箱
- MATLAB提供了功能强大的图像处理工具箱，实现了众多图像处理的算法，简洁高效



图像处理基础

- 数字图像（单通道）定义为二维函数$f(r,c)$；
- 其中r,c为像素坐标，f为该像素的亮度（灰度）；
- 经采样、量化得到的数字图像，其坐标和灰度取离散整数，通常可以用矩阵来表示位置与灰度的对应关系：

$$f=\begin{pmatrix} f(1,1),\ f(1,2),\dots,\ f(1,N) \\ f(M,1),\ f(M,2),\dots,\ f(M,N) \end{pmatrix}$$

图像处理基础

- MATLAB中对图像灰度支持的类型：

double	64位双精度浮点数
single	32位单精度浮点数
uint8	8位无符号整数 0-255 （常见）
uint16	16位无符号整数 0-65535
uint32	32位无符号整数
int8	8位有符号整数
int16	16位有符号整数
int32	32位有符号整数
logical	0或1的逻辑值

图像处理基础

- 灰度图（uint8或unit16）：

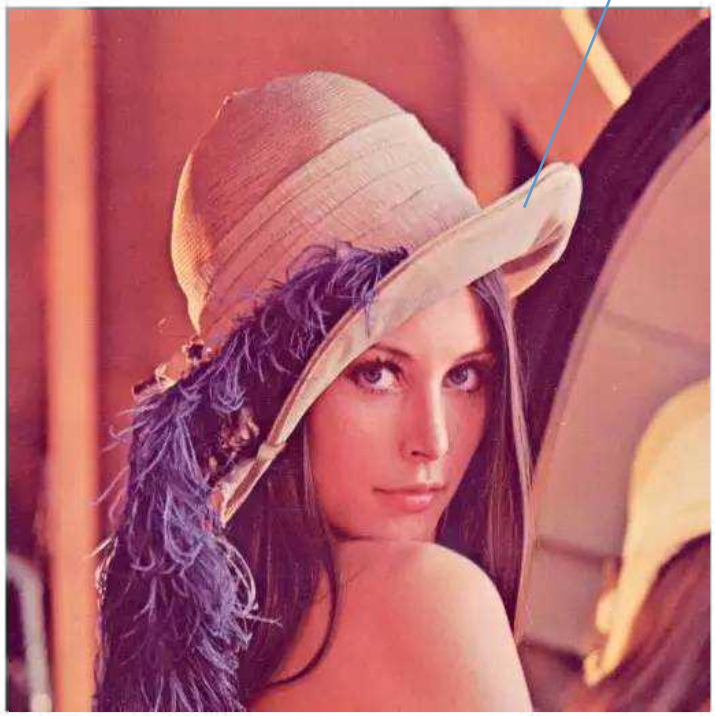
$$f(27,78) = 240$$

单通道矩阵，通常采用无符号整数，即像素值非负，以8位居多，像素值范围0-255

图像处理基础

$$f(27,78,1) = 250$$
$$f(27,78,2) = 243$$
$$f(27,78,3) = 240$$

- 彩色图



包括RGB（红绿蓝分量），HSV（色调、饱和度、明度空间）等类型，常见的为RGB

图像处理基础

- 二值图

logical类型，值为0或1，通常应用于图像分割及腐蚀，膨胀等运算

图像处理基础-输入输出

- 图像输入：f=$imread('filename')$
- $filename$是图像全名的字符串（绝对路径或相对路径），MATLAB支持读取.png,.jpg,.gif,.tif等各种文件类型。该函数返回的是图像对应的矩阵表示；
- 图像显示：$imshow(f)$
- 图像输出：$imwrite(f,'filename')$
- ps：对于jpg图像，可以在输出时指定压缩质量，例如：
- $imwrite(f,'src.jpg','quality',q)$

图像处理基础-显示

- imshow显示灰度图像，
- *imshow*(*I*,[*low*,*high*])
  注：imshow(I,[])用法仅仅对单通道灰度图适用。
- 自定义调色板范围，MATLAB将自动对灰度图像进行标度，使得调色板范围合适；
- image显示RGB图像，对double型，uint8型，uint16型都能正确显示，它是MATLAB中最基本的图像显示函数；
- 注：image对于double型的三通道彩色图，要求其取值范围为[0,1]
- imagesc将图像矩阵中的值先缩放到合适的范围，再使用image，优点是图像矩阵中的值过大/过小/过于集中时，仍然可以保证显示的视觉效果

图像处理基础-显示

- A = imread(‘CA.jpg’); %读取原图像并获得各个通道分量
- x1 = A(:,:,1);
- x2 = A(:,:,2);
- x3 = A(:,:,3);

- figure(1)
- subplot(2,2,1),imshow(A) %imshow函数显示
- subplot(2,2,2),imshow(x1)
- subplot(2,2,3),imshow(x2)
- subplot(2,2,4),imshow(x3)

- figure(2)
- subplot(2,2,1),image(A) %image函数显示
- subplot(2,2,2),image(x1)
- subplot(2,2,3),image(x2)
- subplot(2,2,4),image(x3)

- figure(3)
- subplot(2,2,1),imagesc(A) %imagesc函数显示
- subplot(2,2,2),imagesc(x1)
- subplot(2,2,3),imagesc(x2)
- subplot(2,2,4),imagesc(x3)

图像处理基础-转换图像类型

- MATLAB对各种支持的图像类型，提供了方便的转换接口，包括图像类型，图像格式。
- 对于图像类型，一个常见的转换是将RGB彩色图像转换为灰度图，以便于后续操作，处理，MATLAB提供了相应的$rgb2gray()$函数；
- $I=imread(filename)$;
- $grayI=rgb2gray(I)$;
- 灰度图与原图尺寸相同。
- 而对于图像格式的调整，可以读取图像后，直接选择新的格式进行保存。

图像处理基础-转换图像类型

- 例：图像类型调整

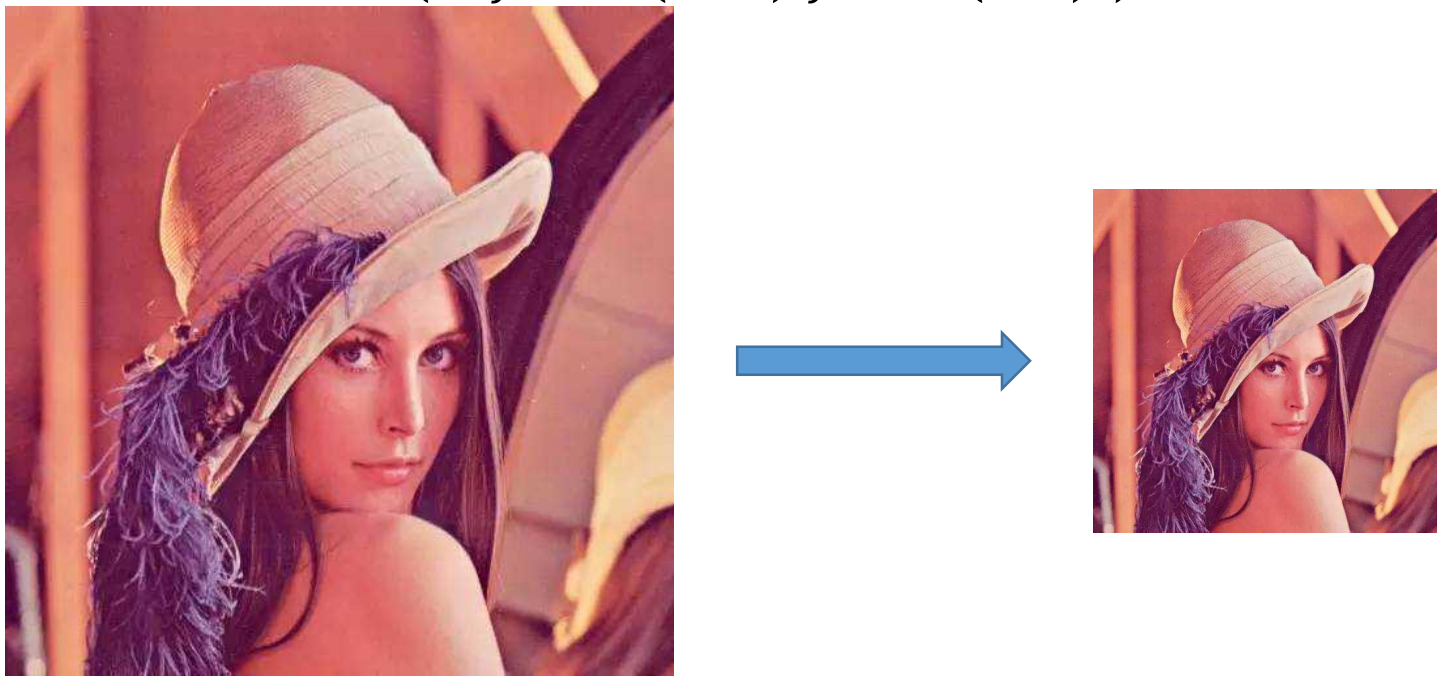
```
I = imread('lena.jpg');                    %读取图像
grayI = rgb2gray(I);                       %转换为灰度图
imwrite(I, 'lena.png')      %将原图保存为png图像
imwrite(grayI, 'grayLena.jpg')             %保存灰度图
```

图像处理基础-图像大小

- 在计算机视觉或机器学习相关研究中，常常需要对图像进行批处理，图像的尺寸差异会带来不便，我们希望将其统一；
- 图像过大，也会对其显示、存储、传递带来麻烦；
- 图像处理工具箱提供了imresize函数，可以对图像进行尺寸放缩调整（插值问题）。
- imresize的调用格式主要分为：
- $imresize(img, scale, METHOD)$;
- $imresize(img, [numRows, numCols], METHOD)$;
- img为输入图像， scale为伸缩比例，[numRows,numCols]为指定的目标尺寸，METHOD为制定的差值方法的字符串，如’nearest’,’bicubic’等

图像处理基础-图像大小

- 例如，希望将图像调整至高、宽各为50%，即4:1缩小
- $I=imread('lena.jpg')$;
- $[m,n,c]=size(I)$;
- $small=imresize(I,[floor(m/2),floor(n/2)])$;



图像处理基础-图像大小

- 实例：图像缩小和放大重建

```
I = imread('filename.jpg');
smallI = imresize(I,0.5);  %图像尺寸缩小
reconI = imresize(smallI, 2.0, 'nearest');
                           %最近邻插值恢复图像
subplot(1,2,1)
imshow(I);                 %比较原图和复原图
subplot(1,2,2)
imshow(reconI)
```

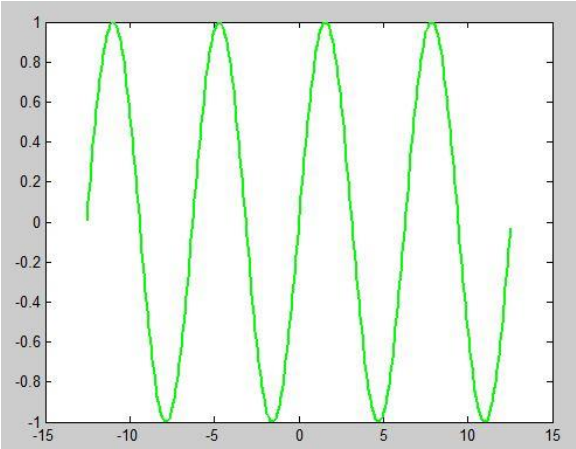
图像处理基础

- MATLAB提供了一个很强大的命令print，它可以将当前程序中的命令窗口输出，以我们想要的方式进行保存，可以保存图像，曲线，模型等多种数据类型。
- 举例来看一下print命令的功能。

图像处理基础

- 例.保存MATLAB中绘制的曲线

```
minbd = -4*pi;                          %变量下界
maxbd = 4*pi;                           %变量上界
t = minbd:0.1:maxbd;                    %生成0.1等间距的向量
plot(t, sin(t), ‘g’, ’Linewidth’, 2);   %plot，绘制其正弦曲线
print(‘-dpng’, ‘sin.png’);              %print，输出为png图像
```

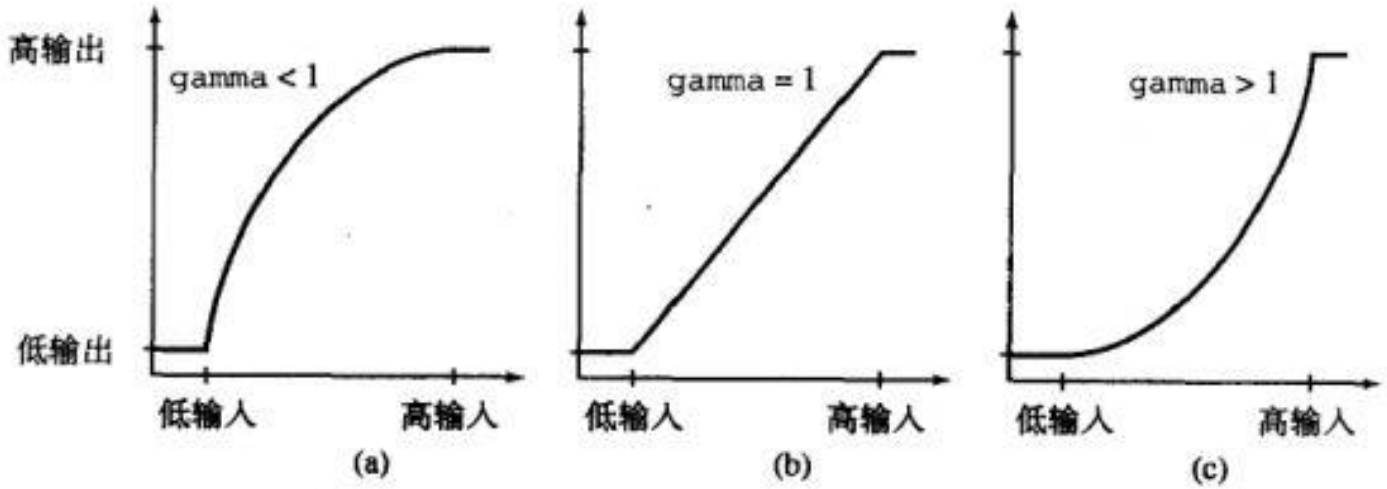


图像处理-滤波

- 图像处理是一门很“杂”的学科，内容丰富，专题众多，这里无法给出详尽介绍，这里主要以空域和频域滤波，及其相关应用给一些简单示例介绍

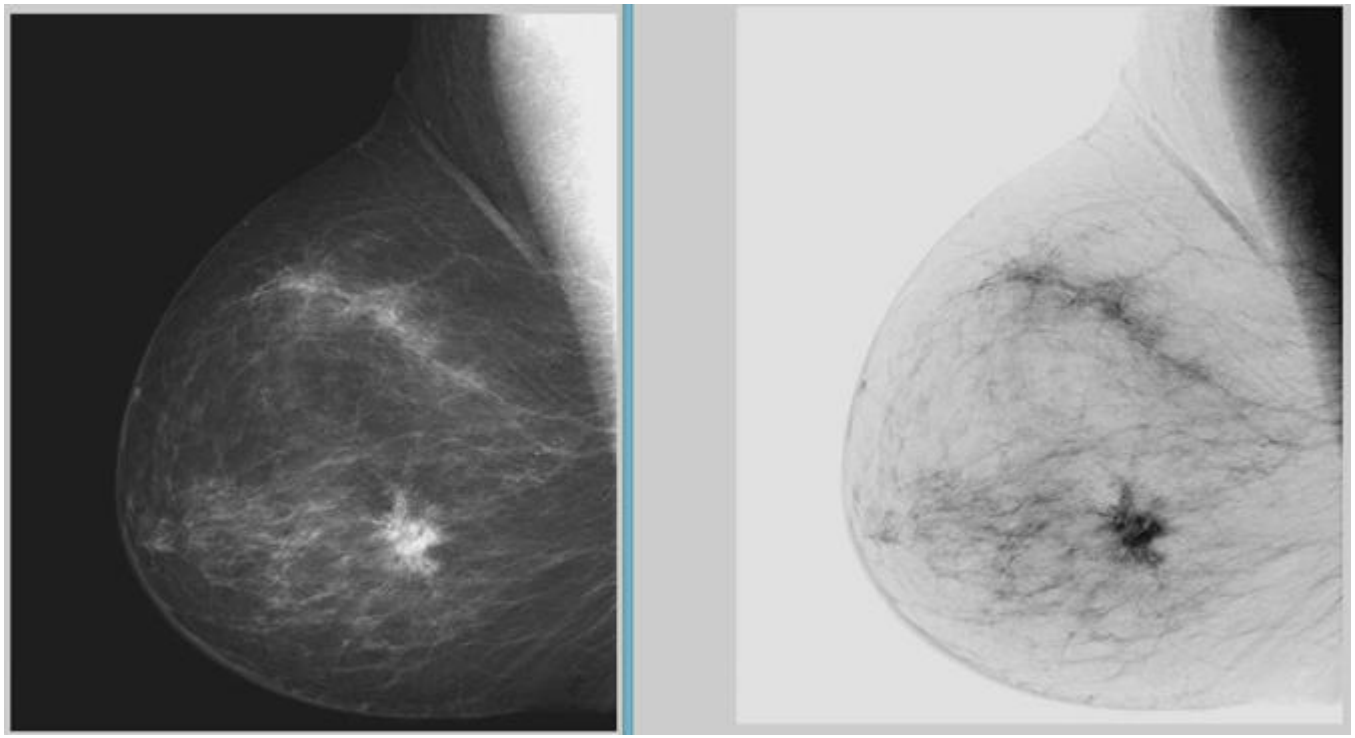
图像处理-空域滤波

- 空间域指的是图像平面本身，通过对图像像素直接操作进行：
- $g(x,y)=T[f(x,y)]$
- 操作的核心是灰度变换函数T
- MATLAB中提供了函数$imadjust$，对全图像进行整体灰度变换：
- $g=imadjust(f,[low\_in,high\_in\ ],[low\_out,high\_out\ ],gamma)$
- 将输入的区间对应映射到输出区间



图像处理-空域滤波

- 例：$g=imadjust(f,[0,255],[255,0])$，灰度线性反转



利用人肉眼对明暗区域的不同感受，实现图像增强

图像处理-空域滤波

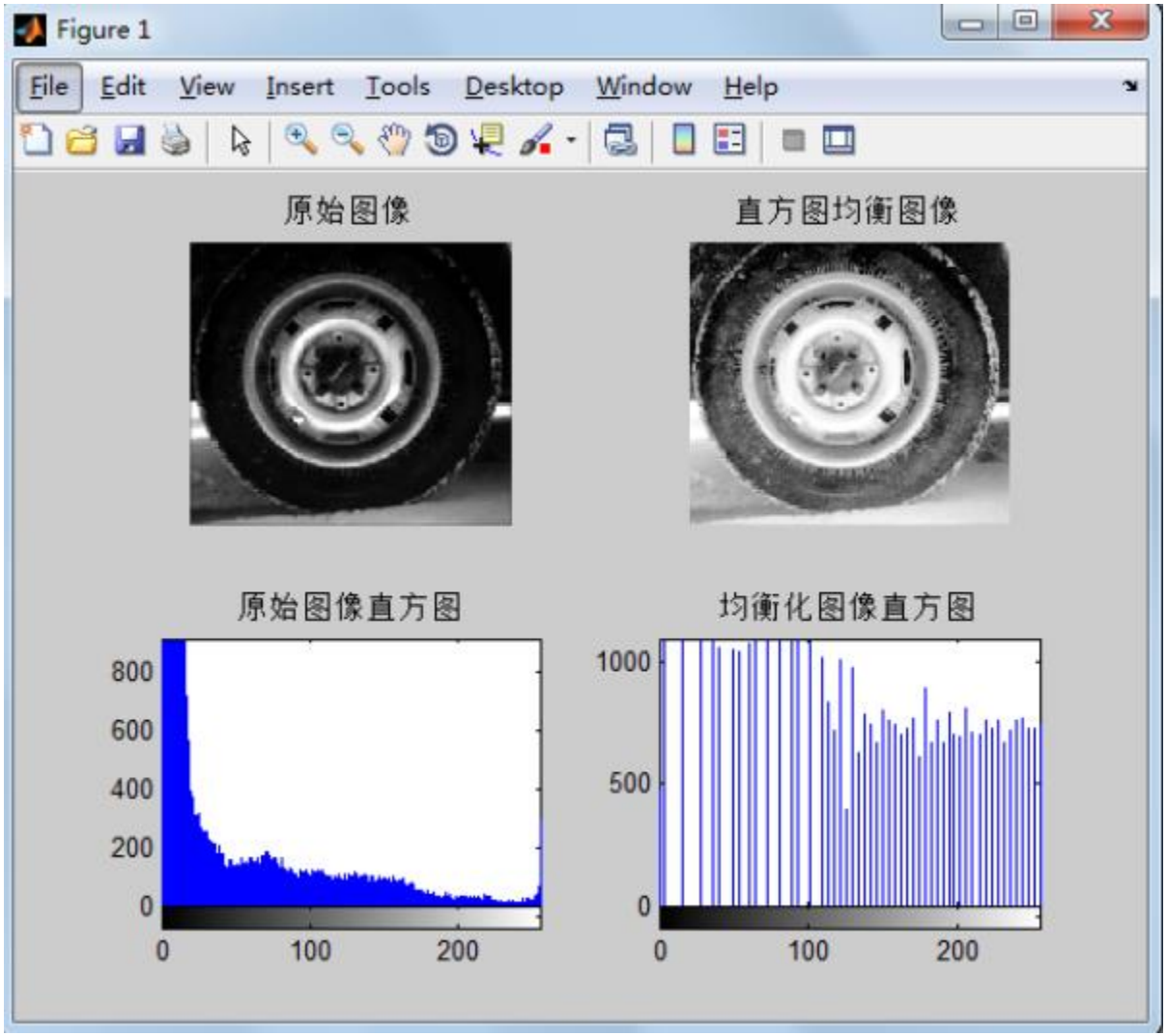
- 空间滤波的典型方法之一是直方图均衡化，原始输入图像的灰度分布可能很不平衡，导致对比度不能满足要求，通过调节灰度使得直方图均衡化，可以增强对比度。
- MATLAB中提供了函数$histeq$对图像进行直方图均衡化处理：
- $q=histeq(f,hspec)$
- 其中$hspec$为希望输出直方图逼近的目标直方图；
- PS：调用函数$imhist(f)$可以得到图像的灰度直方图

图像处理-空域滤波

- 示例：直方图均衡化图像增强

```
I = imread('filename');                    %读取图像
f = rgb2gray(I);                           %转为灰度图
imshow(f);
figure, imhist(f)                          %显示灰度分布直方图
g = histeq(f, 256);                        %直方图均衡化
figure, imshow(g)
figure, imhist(g)
```

图像处理-空域滤波

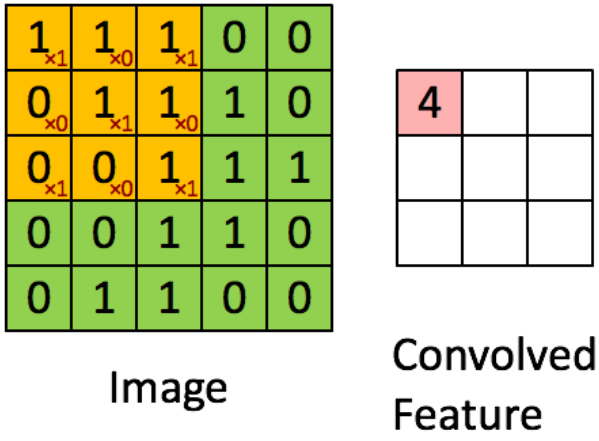


通过直方图均衡化的处理，可以观察到一些原本难以观察的纹理

图像处理-空域滤波

- 线性空间滤波
- 定义$m \times n$的滤波模板（卷积核），与原图像灰度映射做卷积（或相关）运算：

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$



图像处理-空域滤波

- MATLAB提供函数支持线性空间滤波：
- $g=imfilter(f,w,filteringMode,boundaryOptios,sizeOptions)$
- 例如，其常用的调用方式为
- $g=imfilter(f,w,'replicate')$
- 这是将图像的大小首先通过复制图像边界的值来扩展，在扩展图像上进行卷积操作，保证输出图像大小与原图像相同。

图像处理-空域滤波

- 例：利用空间滤波实现边缘检测算子

```
>> f = imread('fig0314(a)(moon).tif');
>> w4 = fspecial('laplacian', 0);
>> w8 = [1 1 1; 1 -8 1; 1 1 1];
>> f = im2double(f);
>> g4 = f - imfilter(f, w4, 'replicate');
>> g8 = f - imfilter(f, w8, 'replicate');
>> imshow(f)
>> figure, imshow(g4)
>> figure, imshow(g8)
```



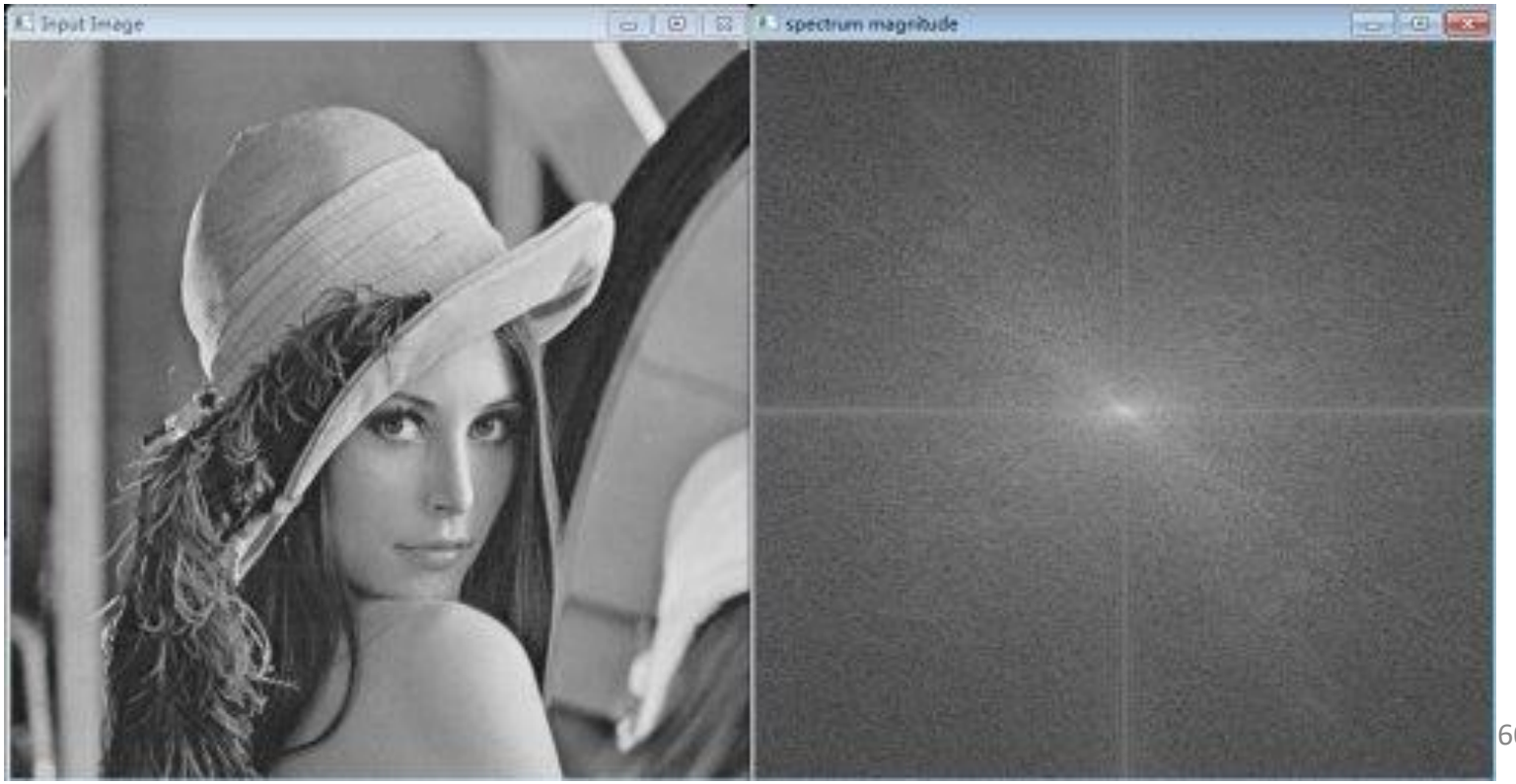
图像处理-空域滤波

- MATLAB图像处理工具箱提供了一些标准的空间滤波器，可以调用函数来生成：
- $w=fspecial('type',parameters)$

type	parameter
average	fspecial('average',[r c]) $r \times c$的矩形平均滤波器
disk	fspecial('disk',r) 半径为r的圆形平均滤波器
gaussian	fspecial('gaussian',[r c], sig) 大小为$r \times c$，标准差为sig的高斯低通滤波器
laplacian	fspecial('laplacian',alpha) 大小为3×3的拉普拉斯滤波器，形状由alpha决定
log	高斯-拉普拉斯（LoG）滤波器
motion	fspecial('motiaon',len,theta) 输出滤波器计算len个像素的线性运动，方向为以水平方向逆时针旋转theat
prewitt	fspecial('prewitt') 输出3×3的Prewitt滤波器，近似计算垂直梯度
sobel	fspecial('sobel') 输出3×3的Sobel滤波器，近似计算垂直梯度
unsharp	fspecial('unsharp',alpha) 输出3×3的非尖锐滤波器，alpha控制形状

图像处理-频域滤波

- 图像的频率分布：通常一幅图像的频率域分布主要集中在低频，而高频分量表征了边缘纹理的信息，往往具有更高价值。



图像处理-频域滤波

- MATLAB中计算和观察二维DFT：

  MATLAB中提供函数$fft2()$，采用快速傅里叶变换来计算DFT

$$F = fft2(f)$$

  并得到傅里叶频谱

$$S = abs(F)$$

  需要注意的是，相比于8位的图像，频谱值的范围变化很大，高频和低频分量对比不利于观察，因此通常由对数变换来解决这一问题

$$S = \log(1 + abs(F))$$

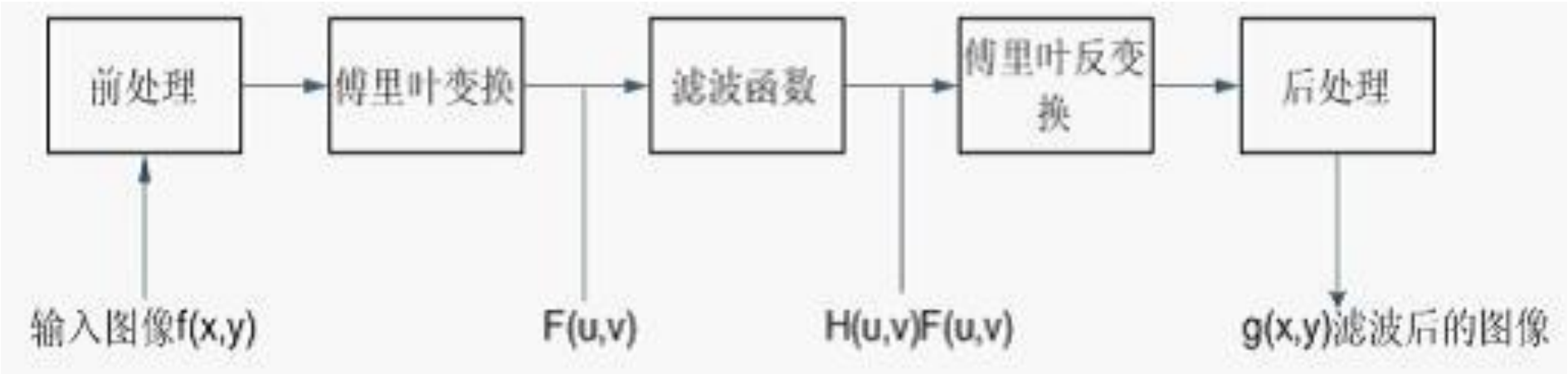
  $fftshift()$可以将频谱居中显示。

  傅里叶反变换为

$$f = ifft2(F)$$

图像处理-频域滤波

- 频域滤波步骤



图像处理-频域滤波

- 实例：Butterworth滤波

```
clc; clear all; close all;

I = imread('cleve_moler.png');
I = rgb2gray(I);
figure('units','normalized','position',[0.1,0.1,0.7,0.7]);

subplot(2, 2, 1);
imshow(I); title('原图像');

J1 = imnoise(I, 'salt & pepper');   % 叠加椒盐噪声
subplot(2, 2, 2);
imshow(J1); title('加噪声图像');

g = fft2(double(J1));       % 傅里叶变换
g = fftshift(g);            % 傅里叶频谱中心化
[M, N]=size(g);
nn = 2;                     % 二阶巴特沃斯低通滤波器
d0 = 20;
m = fix(M/2);
n = fix(N/2);
for i = 1:M
    for j = 1:N
        d = sqrt((i-m)^2+(j-n)^2);
        h = 1/(1+0.414*(d/d0)^(2*nn));   % 滤波器传递函数
        result(i,j) = h*g(i,j);
        T(i, j) = h;
    end
end
result = ifftshift(result);
J2 = ifft2(result);             %傅里叶反变换得到滤波后输出
J3 = uint8(real(J2));
subplot(2, 2, 3);
 mesh(T); title('滤波器示意图')

box on;
% 显示滤波处理后的图像
subplot(2, 2, 4); imshow(J3); title('滤波结果')
```

图像处理-分割

- 图像分割是图像处理和计算机视觉中最重要，也是困难的任务之一。将图像分为各个不同区域，细分程度取决于问题，例如在医疗图像中，对器官的分割结果要求精度非常高。
- 图像分割算法通常基于图像中的不连续性，和像素（或区域）之间的相似性；前者通常基于边缘检测，后者基于规则和区域生长。

图像处理-分割

- 图像中的不连续性检测，针对点、线、边缘这些不同层次的类型，类似于空间滤波的做法，可以用预先定义模板计算响应来寻找指定类型的边缘。
- 例如，点检测，希望找到区域中孤立的点，可以采用如下滤波器模板：

-1	-1	-1
-1	8	-1
-1	-1	-1

点检测规则可以表示为：$g = abs(imfilter(f, w)) \geq T$

图像处理-分割

- 不同类型的线检测模板：

水平

-1	-1	-1
2	2	2
-1	-1	-1

45度

2	-1	-1
-1	2	-1
-1	-1	2

垂直

-1	2	-1
-1	2	-1
-1	2	-1

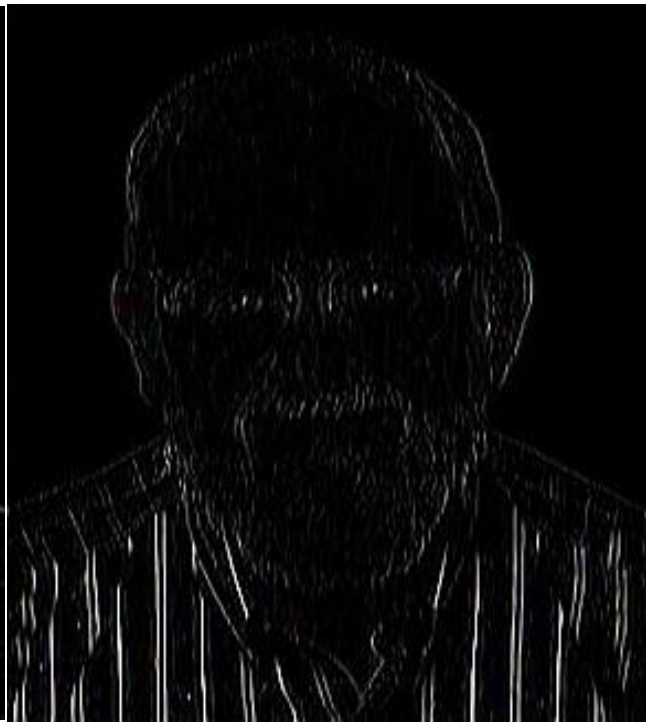
基于空间域滤波的方法，非常直观，更容易理解

图像处理-分割

- 不同的边缘检测结果

水平

45度



竖直

图像处理-分割

- MATLAB图像处理工具箱提供了$edge()$函数进行图像的边缘检测：

➢ $[g,t]=edge(f,'sobel',T,dir)$

➢ $[g,t]=edge(f,'prewitt',T,dir)$

➢ $[g,t]=edge(f,'roberts',T,dir)$

➢ $[g,t]=edge(f,'log',T,sigma)$

➢ $[g,t]=edge(f,'zerocross',T,H)$

➢ $[g,t]=edge(f,'canny',T,sigma)$

参数含义：T为阈值，dir决定方向，$sigma$为标准差

图像处理工具箱

- MATLAB对图像的处理功能主要集中在它的图像处理工具箱(Image Processing Toolbox)中
- 图像处理工具箱是由一系列支持图像处理操作的函数组成，可以进行诸如几何操作、线性滤波和滤波器设计、图像变换、图像分析和图像增强、数学形态学操作等图像处理操作。
- https://cn.mathworks.com/help/images/index.html

图像处理工具箱

- Import, Export, and Conversion
- Display and Exploration
- Geometric Transformation and Image Registration
- Image Filtering and Enhancement
- Image Segmentation and Analysis
- Deep Learning for Image Processing
- 3-D Volumetric Image Processing
- Code Generation
- GPU Computing

图像处理工具箱

- Viewing images is fundamental to image processing. The toolbox provides a number of image processing apps to view and explore images.
- Using the Image Viewer app, you can view pixel information, pan and zoom, adjust contrast, and measure distances.
- The toolbox also provides visual tools for creating your own apps.

图像处理工具箱

- Image Viewer App

- Pixel Information tool — for getting information about the pixel under the pointer
- Pixel Region tool — for getting information about a group of pixels
- Distance tool — for measuring the distance between two pixels
- Image Information tool — for getting information about image and image file metadata
- Adjust Contrast tool and associated Window/Level tool — for adjusting the contrast of the image displayed in the Image Viewer and modifying the actual image data. You can save the adjusted data to the workspace or a file.
- Crop Image tool — for defining a crop region on the image and cropping the image. You can save the cropped image to the workspace or a file.
- Display Range tool — for determining the display range of the image data
- Overview tool — for determining what part of the image is currently visible in the Image Viewer and changing this view.
- Pan tool — for moving the image to view other parts of the image
- Zoom tool — for getting a closer view of any part of the image.
- Scroll bars — for navigating over the image.

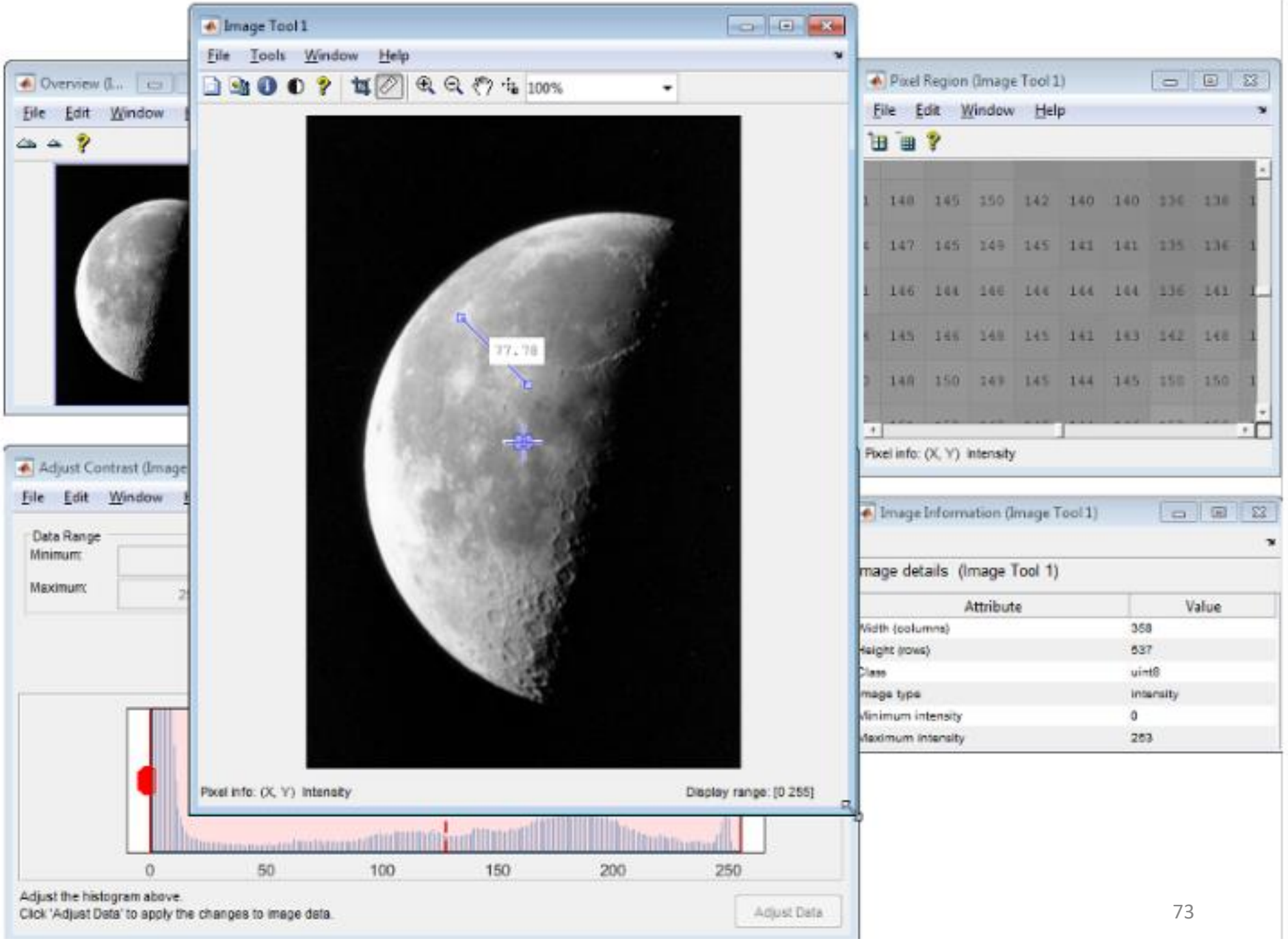


Image Tool 1
File Tools Window Help
100%
77.78
Pixel info: (X, Y) Intensity
Display range: [0 255]
Overview (I...
File Edit Window
Pixel Region (Image Tool 1)
File Edit Window Help
Pixel info: (X, Y) Intensity
Image Information (Image Tool 1)
Image details (Image Tool 1)
Attribute
Value
Width (columns)
358
Height (rows)
537
Class
uint8
Image type
intensity
Minimum intensity
0
Maximum intensity
253
Adjust Contrast (Image
File Edit Window
Data Range
Minimum:
Maximum:
0
50
100
150
200
250
Adjust the histogram above.
Click 'Adjust Data' to apply the changes to image data.
Adjust Data

其他功能

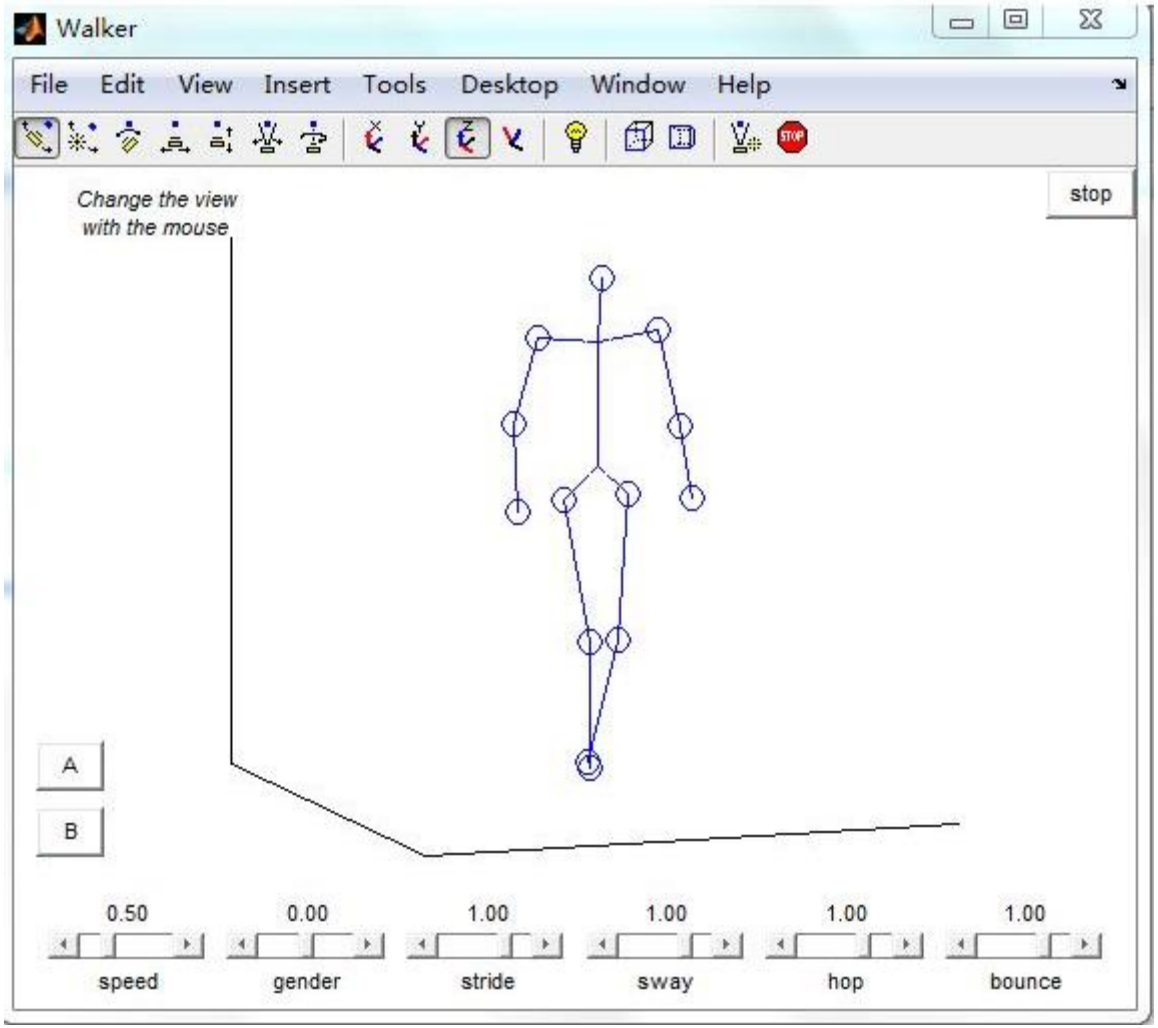
- GPU加速(不推荐)
- parfor加速(CPU)
- 符号运算
- GUI
- mex

其他功能-GUI

- MATLAB支持图形用户界面（Graphical User Interface, GUI），相比于单纯的命令行处理，图形界面对于用户而言有更好的视觉体验，在某种意义上增强了实用性。

其他功能-GUI

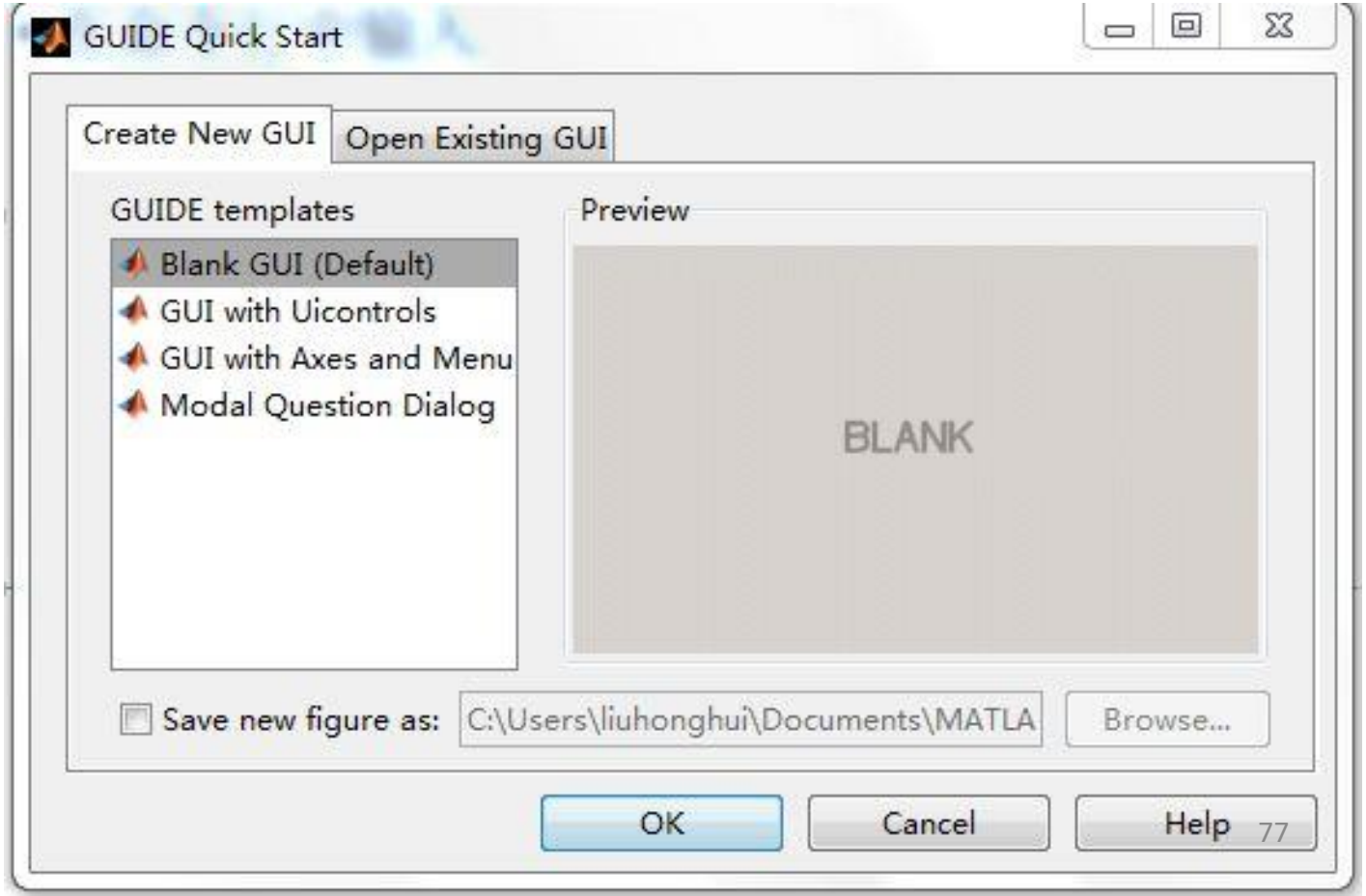
- 例：人体步态模型的MATLAB GUI实现



其他功能-GUI

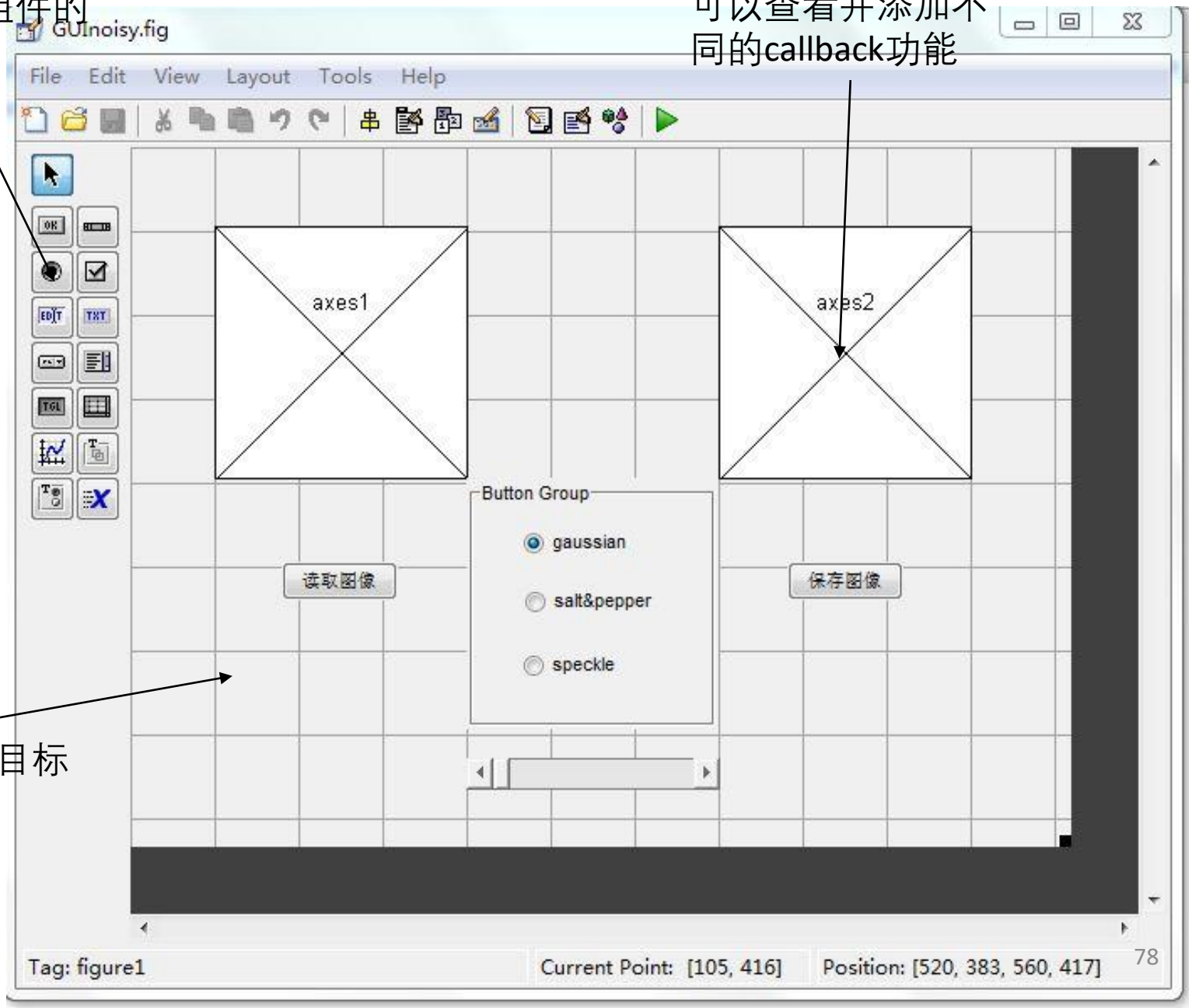
- MATLAB的GUI可以完全基于代码自主实现，也可以利用MATLAB提供的GUI生成功能，快速上手。
- 在命令行中输入guide命令，打开GUIDE Quick Start窗口：

>>guide



将各个基本GUI组件拖曳到右侧窗口区域中，完成组件的添加

右键单击各个组件，选择view callbacks可以查看并添加不同的callback功能



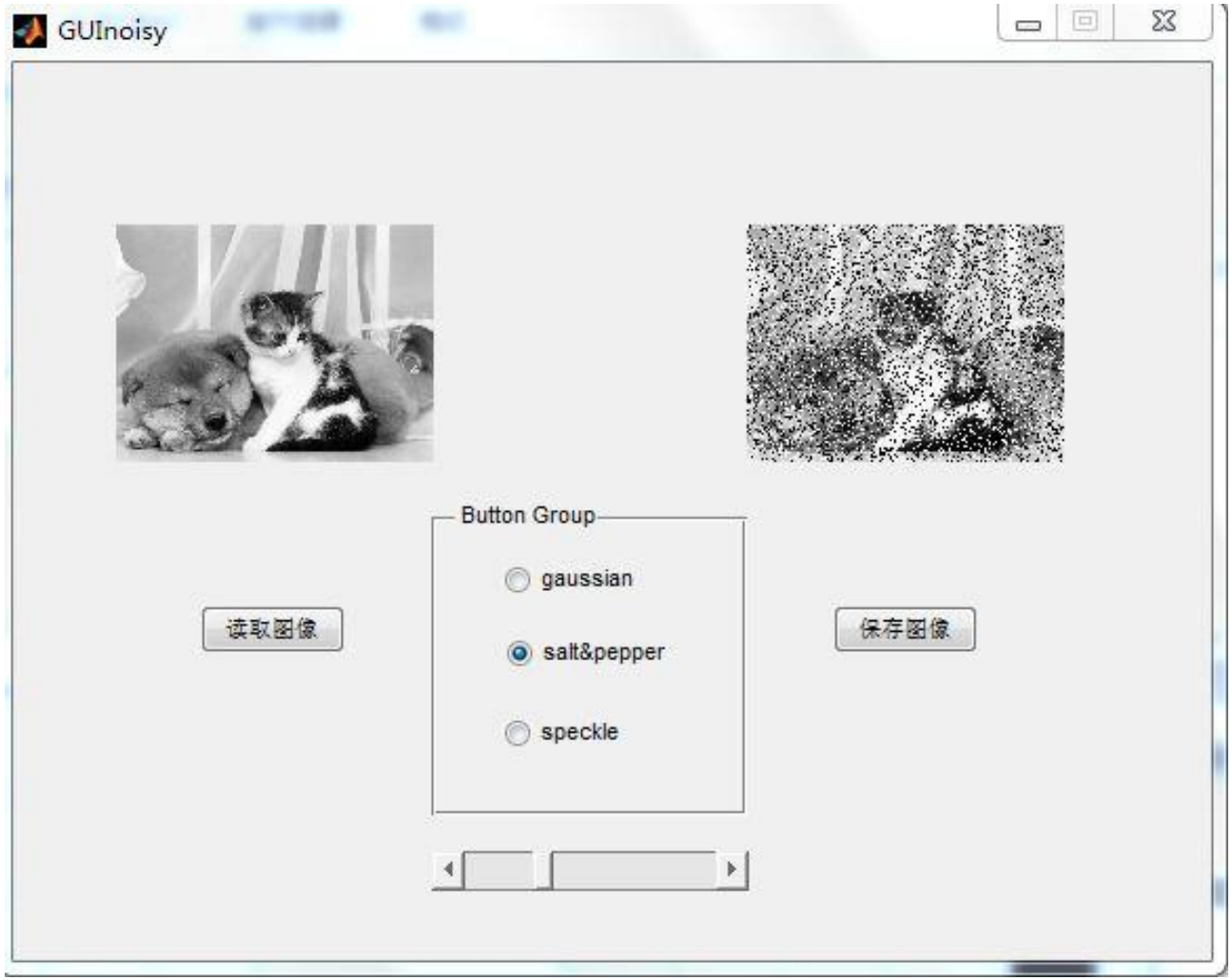
生成的GUI的目标窗口

其他功能-GUI

- GUI编程的基础：callback
- Callback功能（回调函数），是指通过函数指针调用的函数，即把函数指针作为参数传递给其他函数，使得其他函数可以调用它。
- 回调函数在特定事件发生，或触发特殊条件时被调用，这种机制很适用于GUI编程；
- 例如，当用户鼠标左键单击某按钮时，触发该按钮的pushdown callback，执行相应的函数所定义的操作

其他功能-GUI

- 示例：图像噪声的GUI示范



其他功能

- MATLAB具有接口简单，操作方便的优点，“节省开发时间”是其核心优势，然而MATLAB程序经常受到对其性能的批判，相比于编译型语言，“运行时间”是其劣势；
- 对于这一问题，可以解释为：首先脚本语言相比于编译型语言在速度上具有固有的差距；而且，并不是所有人都能够很熟练地用MATLAB写“MATLAB风格”的程序。
- MATLAB本质上是处理矩阵，而非标量的语言！

其他功能

- 现在很多工程问题，要处理庞大的数据量，计算的复杂度之大，对程序性能提出很高要求；
- 因此，MATLAB提供了C/C++的混合编程功能，可以在其中编译并调用基于C++的函数，极大地提高了其性能。
- 理想目标：综合利用MATLAB的“方便”和C++的速度，最大化工作效率。

其他功能-mex

- MATLAB提供了mex编译器，来编译C++程序
- 配置mex编译器（简单）：
- 　在MATLAB命令窗口输入
- >>mex -setup
- 会自动检测系统中已经安装的C++编译器，如果检测到合适的版本，则自动安装

```
>> mex -setup

Welcome to mex -setup.  This utility will help you set up
a default compiler.  For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2012a/win64.html

Please choose your compiler for building MEX-files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Microsoft Software Development Kit (SDK) 7.1 in C:\Program Files (x86)\Microsoft Visual Studio 10.

[0] None

Compiler: 1

Please verify your choices:

Compiler: Microsoft Software Development Kit (SDK) 7.1
Location: C:\Program Files (x86)\Microsoft Visual Studio 10.0

Are these correct [y]/n? y
```

其他功能-mex

- C++程序示例：
- 创建mexTest.cpp，在其中写入如下代码：

```
//编译mex需要的头文件
#include “mex.h”
//其他一些头文件
#include <iostream>
#include <vector>
using namespace std;
```

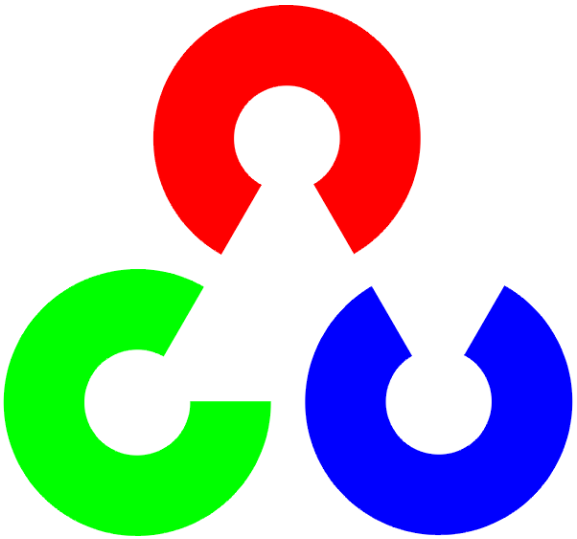
```
void mexFunction(int nlhs, mxArray * plhs[], int nrhs, const mxArray * prhs[])
//mexFunction相当于mex中的main函数
{
        //plhs输出矩阵的指针数组；prhs输入矩阵的指针数组；
        //nlhs输出矩阵的个数；      nrhs输入矩阵的个数；
        //mxArray是mex文件中的矩阵数据结构，是交互的核心
        double * dataCursor = NULL;   //通过对double的指针来处理矩阵元素
        vector<vector<double> > parms;

        dataCursor = mxGetPr(prhs[0]); //获得prhs的第一个元素，即第一个输入矩阵
        int mrows = mxGetM(prhs[0]);  //获得矩阵的行数
        int mcols = mxGetN(prhs[0]);                    //获得矩阵的列数

        parms.resize(mrows);
        for(int i = 0; i < mrows; i++)
                parms[i].resize(mcols);
        for(int i = 0; i < mrows; i++)
                for(int j = 0; j < mcols; j++)
                        parms[i][j] = dataCursor[j*mrows + i];

        plhs[0] = mxCreateDoubleMatrix(mrows, mcols, mxREAL);
        //创建一个新的矩阵，作为输出的第一个元素
        double * outCursor = mxGetPr(plhs[0]);
        for(int i = 0; i < mrows; i++)
                for(int j = 0; j < mcols; j++)
                        outCursor[j*mrows + i] = -1.0*parms[i][j];
}
```

其他功能-mex

- 在MATLAB命令窗口中，调用mex进行编译：
- >>mex mexTest.cpp
- 编译成功后，会生成mexTest.mex文件（或.mexw64文件，根据系统差异有所不同！）
- 这种.mex格式的文件本质上也是M函数文件，调用时执行其中定义的mexFunction函数，调用方式与.m文件相同，例如，对于mexTest，我们知道它有一个输入和一个输出，因此调用方式为：
- >>B = mexTest(A);

其他

- 更多MATLAB图像处理相关资源：
- Command: help XXX
- 官网教程：
  http://cn.mathworks.com/products/image/
- R.Gonzalez, R.Woods, S.Eddins,‘Digital Image Processing Using MATLAB’（有中文版，并有专门的本科教学版）



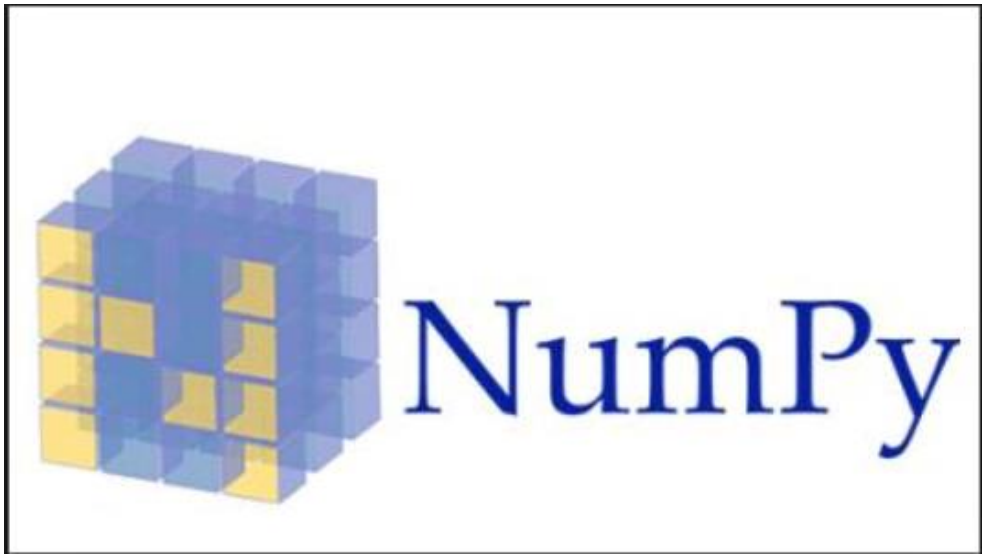
官网：http://opencv.org/

OpenCV（Open Source Computer Vision Library），是开源的跨平台计算机视觉库，可以运行在Linux、Windows和Mac OS操作系统上。它包含一系列 C 函数和少量 C++ 类，非常高效，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV早期用C语言编写，新版本的OpenCV用C++语言编写，它的主要接口也是C++语（但保留了大多数的C语言接口）。该库也有大量的Python, Java and MATLAB的接口。这些语言的API接口函数可以通过在线文档获得。如今也提供对于C#, Ruby的支持。

最新的OpenCV支持CUDA架构和GPU编程。

人生苦短
我用
python™



Python是近年来广受欢迎的语言，具有丰富和强大的库，被称为“胶水语言”，适用于测试算法，快速生成程序原型，python提供PIL库进行图像处理，配合科学计算库NumPy的矩阵计算功能，是图像处理的强大工具

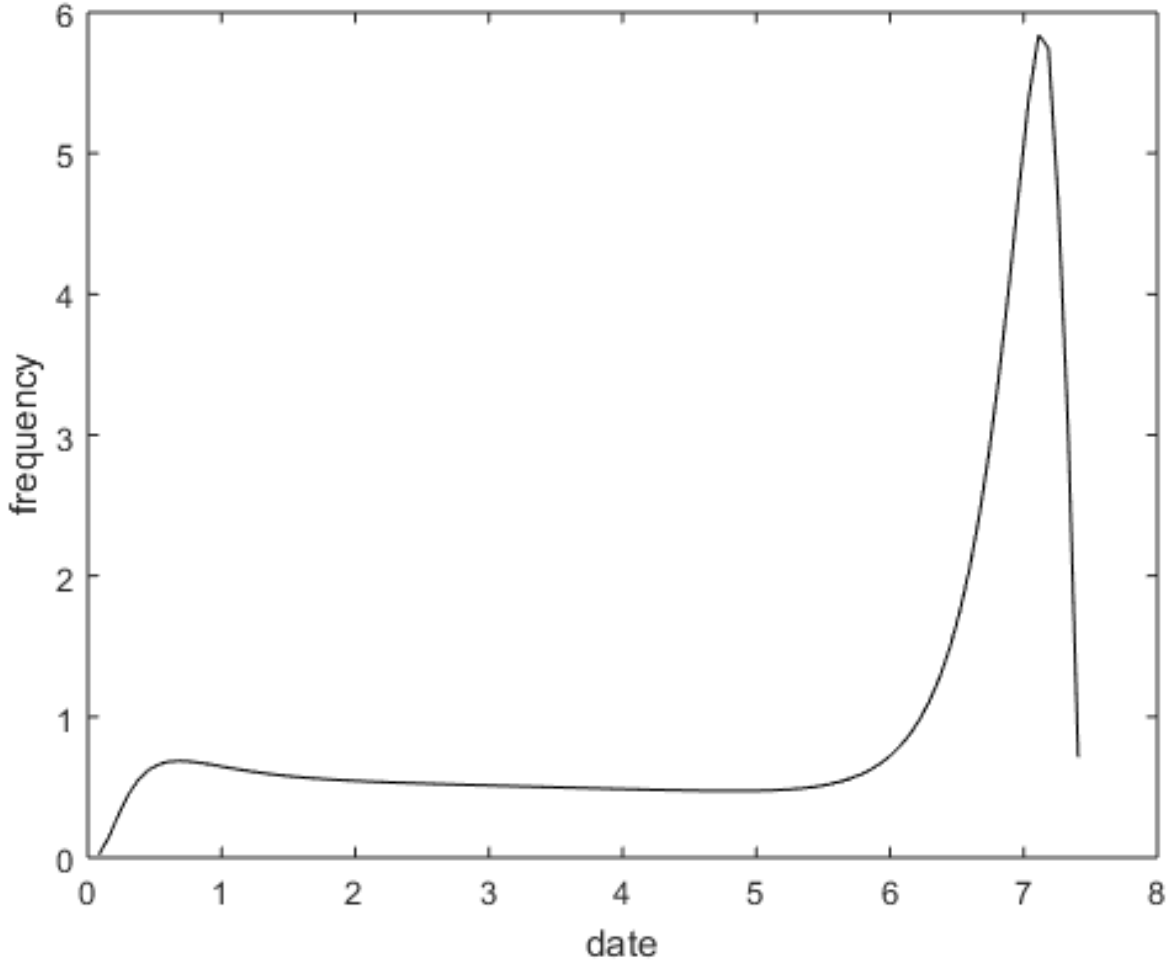
编程注意

- MATLAB的基础是矩阵运算，尤其是大规模数组时，一定想办法用矩阵运算，不要写循环，否则效率令人发指。
- 代码要有注释，比如函数的功能，输入输出，参数设置，一些关键语句。不要指望自己的记忆力，过几个月全忘了，改都没法改。
- 大规模程序，尤其是并行程序，一定要有容错机制，即使部分程序bug，不至于整体中断。(try…catch…)
- 不要着急写代码，写之前先想好程序框架流程，哪里会出bug，脑子里（或纸上）打好草稿。一个好的程序员只有1/3的时间在写代码，2/3的时间在思考程序架构。

作业要求

- 按时提交，独立完成，禁止抄袭，但鼓励互相交流；
- 代码清晰，加必要的注释是良好的习惯；
- 在编辑过程中被注释掉的大段“伪注释”代码不要保留；
- 一般不允许直接调用MATLAB工具箱实现好的函数（工作量很大的综合性作业除外）；

作业要求



交作业最高峰发生在28-Sep-2017 02:40:41

finally

- MATLAB图像处理工具箱内容极为丰富，通常我们只能用到其冰山一角；
- 当然，任何工具都不是完美的，MATLAB也是如此，相比于其他平台有其显著优势，也有其劣势；
- 因此，最重要的不是学习语言，或者平台，对原理本身的掌握和运用，才是图像处理学习的精髓。
- 鼓励大家开拓思维，积极学习！
- 学好图像处理，玩转MATLAB，祝你们成为好朋友！