Image Processing in Matlab

By Ji Hui

References

- Inline help:

  >>help

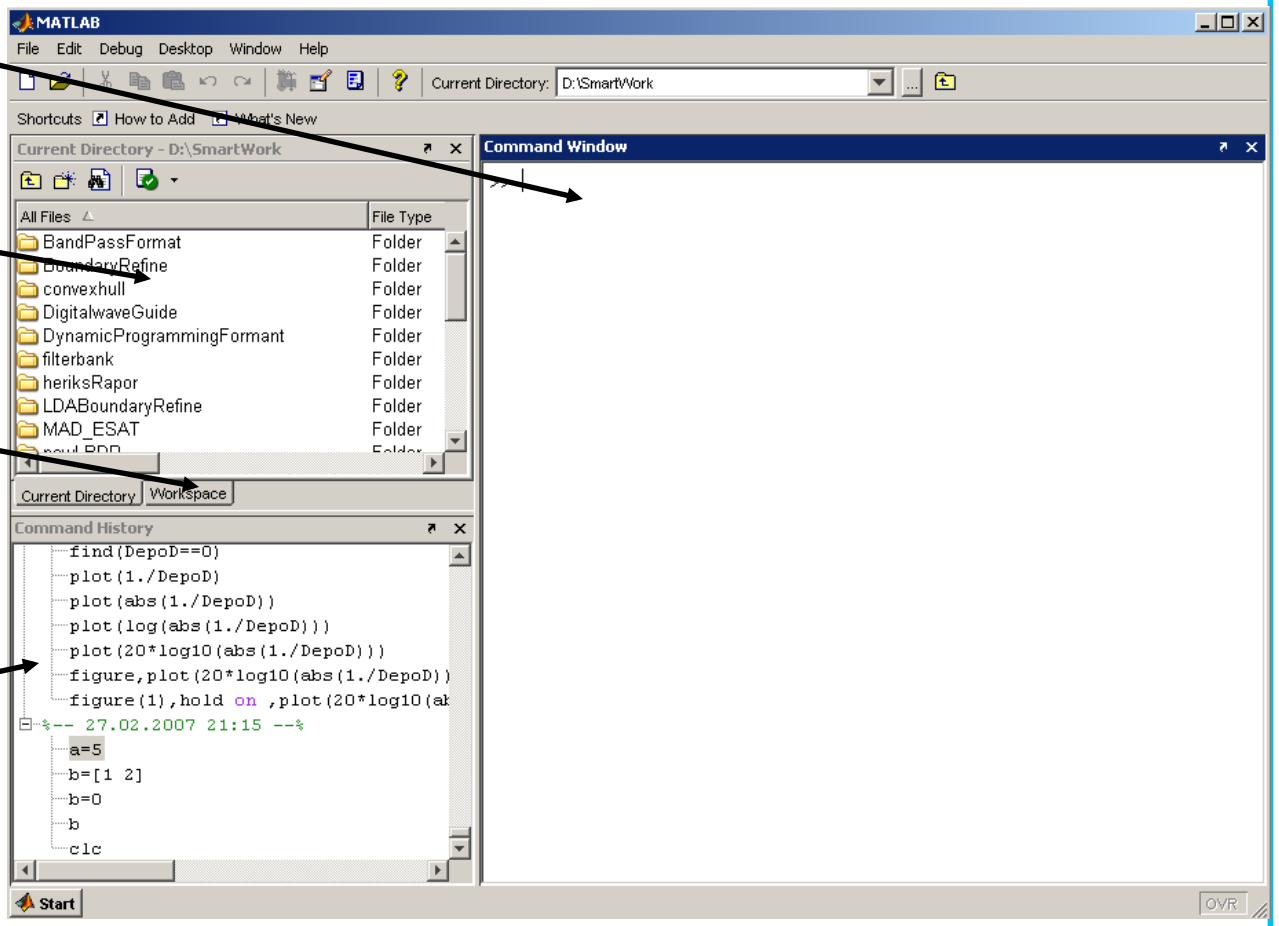
  >>help FUNCTION

  >>helpdesk

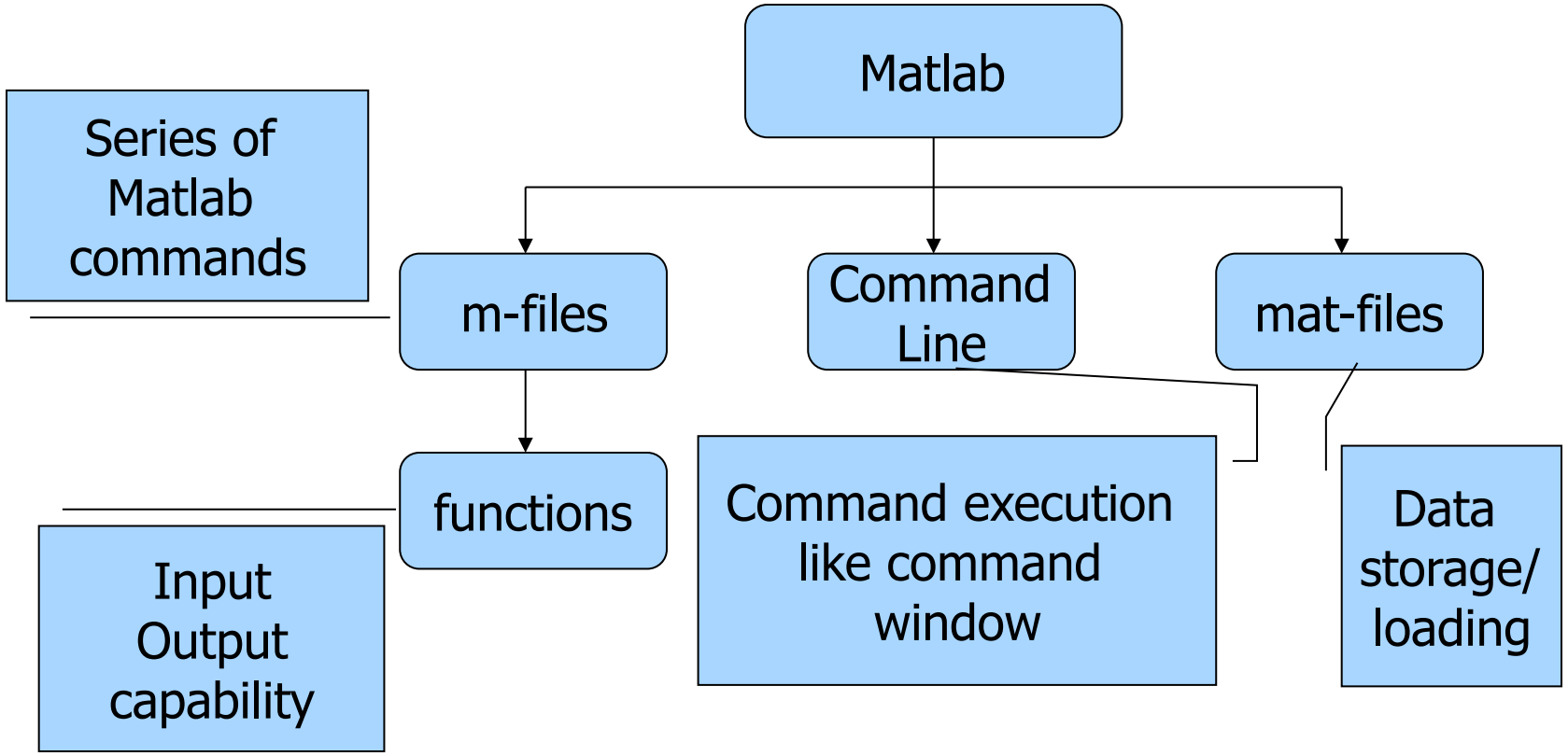
- Online help resources
  - http://www.mathworks.com/help
- Ebooks
  - http://www.mathworks.com/moler/index.html

Matlab Screen

- Command Window
  - type commands
- Current Directory
  - View folders and m-files
- Workspace
  - View program variables
  - Double click on a variable to see it in the Array Editor
- Command History
  - view past commands
  - save a whole session using diary



Organization of Matlab



Variable

- Variable declaration and assignment

```
>> t = 3.2
t =
  3.2000
>> x = 2 * t
x =
  6.4000
```

- Variable name is case sensitive
- Variable name consist only of the letters a–z, A—Z, the digits 0–9 and the underscore, starting with a letter
- Using semicolon in the end of expression will suppress the output

Data type

- Double precision floating-point number (default)
- Unsigned 8-bit integer
  - [0..255]
- Conversion between two types

```
>> uint8(3.1415)
ans =
     3
>> uint8(20)*20
ans =
     255
>> double(20)*20
ans =
     400
```

(cont')

- Complex number

```
>> 1+2i
ans =
     1.0000 + 2.0000i
```

  - `real();imag()`
- Char and string
  - Char: e.g. 'z'
  - String: e.g. 'zebra'

Array and matrix

- A vector

```
>> x = [1 2 5 1]

x =
          1    2    5    1
```

- A matrix

```
 >> x = [1 2 3; 5 1 4; 3 2 -1]

x =
    1     2     3
    5     1     4
    3     2    -1
```

Long array and matrix

- `t = 1:10`

```
t =
     1     2     3     4     5     6     7     8     9    10
```

- `k = 2:-0.5:-1`

```
k =
    2.0   1.5   1.0   0.5   0.0  -0.5  -1.0
```

- `B = [1:4; 5:8]`

```
B =
     1     2     3     4
     5     6     7     8
```

Generating Vectors from functions

- zeros(M,N) MxN matrix of zeros

```
>>x = zeros(1,3)
x =
   0     0     0
```

- ones(M,N) MxN matrix of ones

```
>>x = ones(1,3)
x =
   1     1     1
```

- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
>>x = rand(1,3)
x =
  0.9501   0.2311   0.6068
```

Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be positive integer

```
A =

     3     5     3
     6     8     2
     2     7     3
```

```
>> A(3, 2)

ans =

     7
```

```
>> A(2, :)

ans =

     6     8     2
```

```
>> A(1:2, 2)

ans =

     5
     8
```

```
>> A(6)

ans =

     7
```

Concatenation of Matrices

- x = [1 2], y = [4 5], z=[0 0]

```
A = [ x y]

    1    2    4    5

B = [x ; y]

    1  2
    4  5
```

Matrices Operations

Given A and B:

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

A =

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

```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B

Y =

    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

```
>> Z = A * B

Z =

    22    27    45
    55    66   102
    88   105   159
```

Transpose

```
>> T = A'

T =

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

Using "." for Element-wise Operation

- .* element-by-element multiplication
- ./ element-by-element division
- .^ element-by-element power

```
A = [1 2 3; 5 1 4; 3 2 1]
    A =
          1     2     3
          5     1     4
          3     2    -1
```

```
x = A(1,:)

x=
     1   2   3
```

```
y = A(3 ,:)

y=
     3   4   -1
```

```
b = x .* y

b=
      3   8  -3
```

```
c = x . / y

c=
   0.33   0.5   -3
```

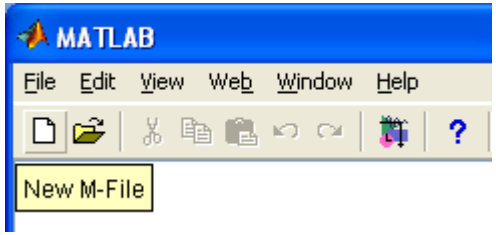
```
d = x .^2

d=
      1     4     9
```

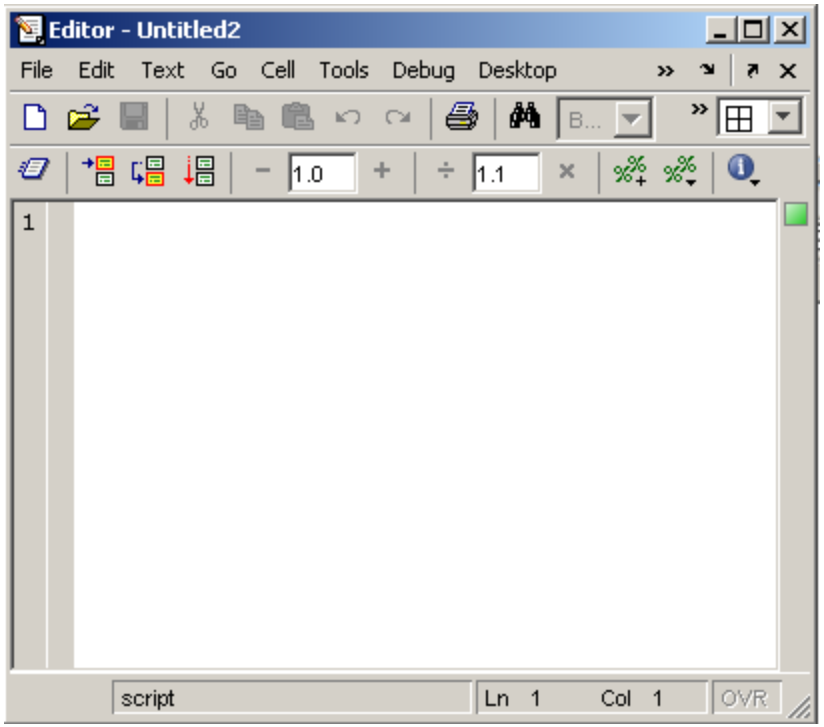
Logical operation and Flow Control

- Logical operation
  - ==, ~=, <, > ,<=, >=, |, &
- Flow structure
  - if
  - for
  - while
  - until

Use of M-File

Click to create a new M-File

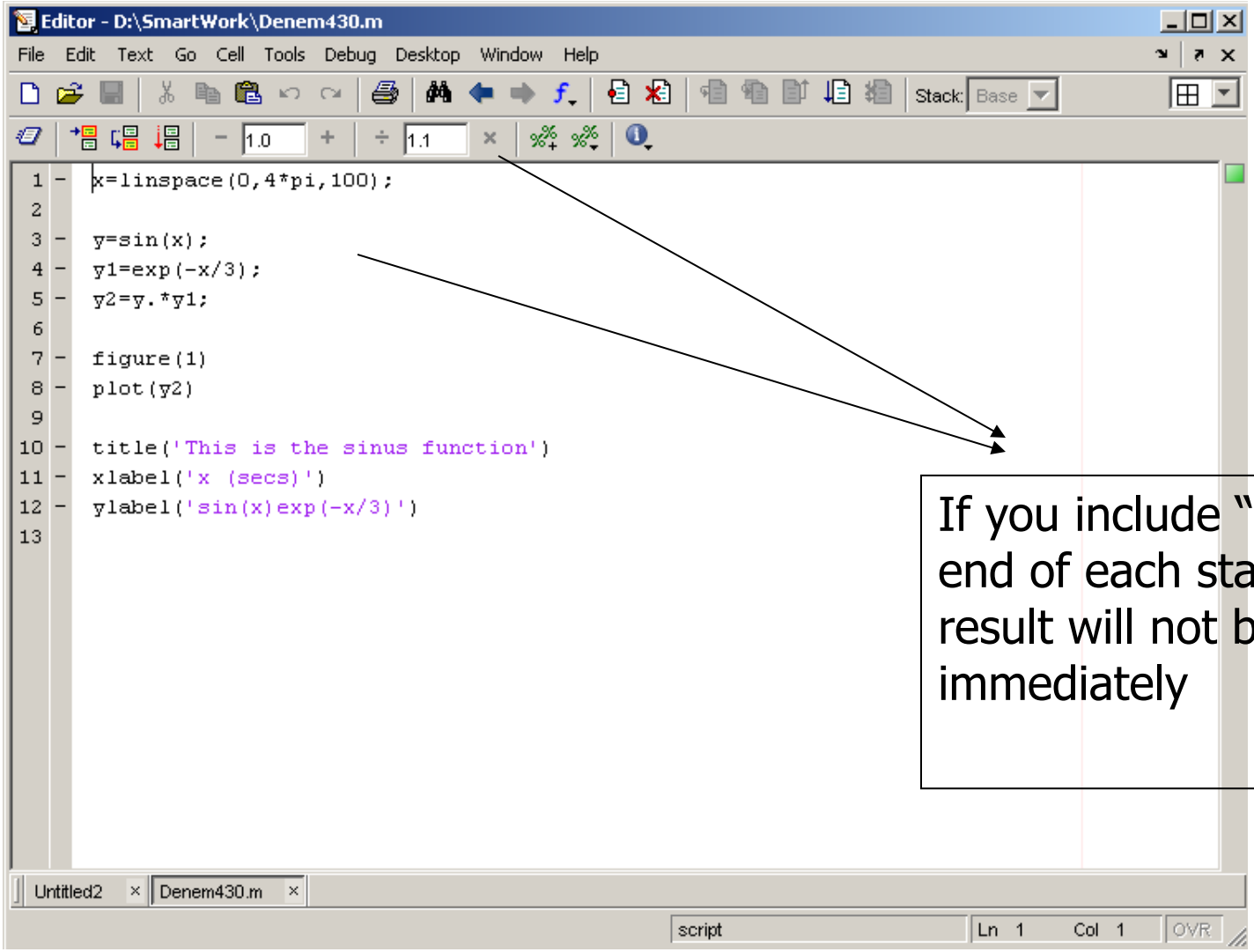




- Extension ".m"
- A text file containing script or function or program to run

Using M-File as script file

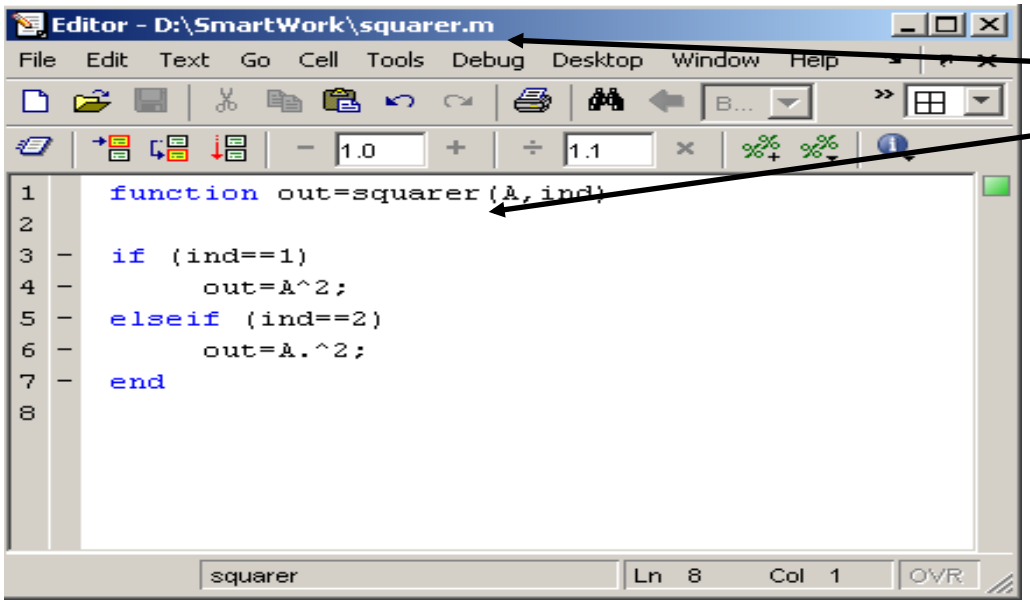
Save file as *demo*.m



If you include ";" at the end of each statement, result will not be shown immediately

Using m.file to define functions

- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input matrix if the input indicator is equal to 1
    - And takes the element by element square of the input matrix if the input indicator is equal to 2

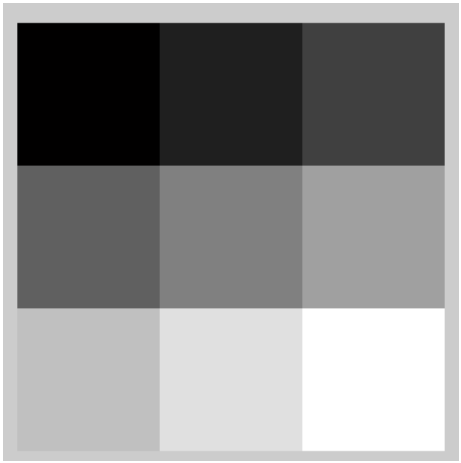


File name same
As function name

Images in Matlab

- Digital images is composed of pixels
- Pixels
  - small dots on the screen
- A digital image is an instruction of how to color each pixel
- Images are (2-dim or 3-dim) matrices in Matlab

$$\begin{pmatrix} 28 & 57 & 85 \\ 113 & 142 & 170 \\ 198 & 227 & 255 \end{pmatrix}$$



Input and output of images

- MATLAB can import/export several image formats
  - BMP, GIF, JPEG, PNG, TIFF, etc
- Read and write images in Matlab

```
>> I=imread('cameraman.tif');
>> imshow(I)
>> size(I)
ans =
     256   256
>> J=uint8(double(I)+10);
>> imshow(J);
>> imwrite(J, 'newcameraman.png');
```

Image types and matrices

- Binary image:
  - pixel value is either black or white: {0,1}
- Greyscale image:
  - pixel value is a shade of grey: {0 ,1,…, 255}
- Color image: (R,G,B)
  - Here each pixel has a particular colour (~ six millon colours)

Greyscale image

- Value of each pixel ranging from 0 to 255, can be represented by eight bits (one byte)
- Data type: m-by-n matrices of **uint8** number

230	229	232	234	235	232	148
237	236	236	234	233	234	152
255	255	255	251	230	236	161
99	90	67	37	94	247	130
222	152	255	129	129	246	132
154	199	255	150	189	241	147
216	132	162	163	170	239	122

Color images (RGB images)

- each pixel has a particular colour; that colour being described by the amount of red, green and blue in it.
- Bits required for each pixel is 3*8=24,
- Number of colours: $2\uparrow 24 \approx 6\ Millions$
- Data type: m-by-n-by-3 matrices of uint8 number

(cont')

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue

Data conversion

- Data read from images is of type **uint8**, convert it to double before applying some operations

```
>>cameraman = imread('cameraman.tif');
>>cameraman = double(cameraman);
```

- Before writing data to image files or for display, convert it to uint8.

```
>>newman = cameraman /2;
>>imshow(uint8(newman));
>>imwrite(uint8(newman),'newcameraman.png');
```

Basic image manipulations in Matlab

- Image resize

```
>> im =imread('cameraman.tif');
>> whos im
  Name         Size         Bytes  Class  Attributes

  im         256x256        65536  uint8

>> new_im = im(1:2:end,1:2:end);
>> whos new_im
  Name           Size       Bytes  Class Attributes

  new_im       128x128      16384  uint8

>> imshow(new_im)
```

(cont')

- Image cropping

```
>> im = imread('peppers.png'); imshow(im);
>> figure; imshow((im(201:end-100,201:end-200,:)));
```

Converting color image to greyscale image

- Brightness = 0.2989 * R + 0.5870 * G + 0.1140 * B

```
>>im = double(imread('peppers.png'));
>>greyim = 0.2989*im(:,:,1) + 0.5870*im(:,:,2)+
0.1140*im(:,:,3);
```

2D Fourier transform in Matlab

- 2D DFT and iDFT
  - fft2(x): two-dim discrete Fourier Transform.
  - ifft2(x): two-dim inverse discrete Fourier Transform.
- Implementation of FFT in Matlab is not centering at zero frequency
  - fftshift(x): Shift zero-frequency component to center of spectrum
  - ifftshift(x): IFFTSHIFT undoes the effects of FFTSHIFT

The end

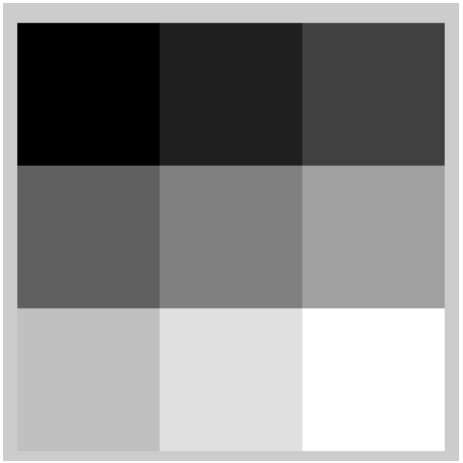
Image Processing using MATLAB
Part 2

By Ji Hui

Images in Matlab

- Digital images is composed of pixels
- Pixels
  - small dots on the screen
- A digital image is an instruction of how to color each pixel
- Images are (2-dim or 3-dim) matrices in Matlab

$$\begin{pmatrix} 28 & 57 & 85 \\ 113 & 142 & 170 \\ 198 & 227 & 255 \end{pmatrix}$$



Read an Image

- Read in an image
- Accepted image format
  - bmp, hdf, jpeg, pcx, png, tiff
- Store it in an matrix

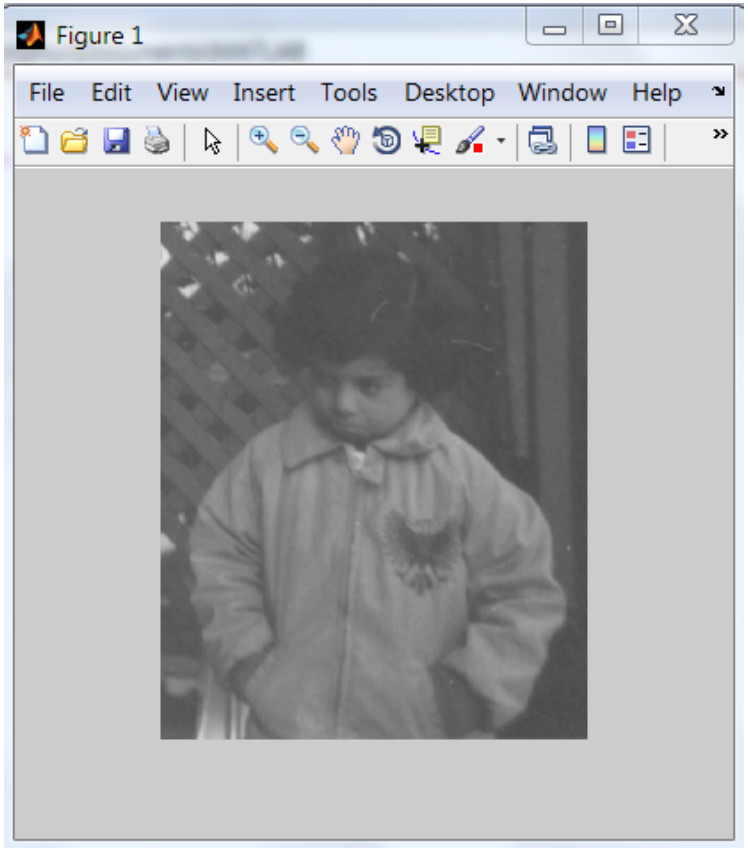
```
>>clear;
>>I = imread('pout.tif');
```

- Check it in memory

```
>> whos I
Name        Size            Bytes  Class    Attributes
I         291x240           69840  uint8
```

Display an Image

>>imshow(I)



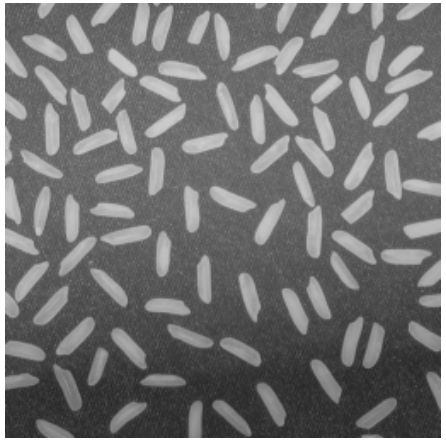
Write the Image

- Validates the extension
- Writes the image to disk

```
>> imwrite(I, 'pout2.png');
>> dir
.          ..          pout2.png
```

Adding images

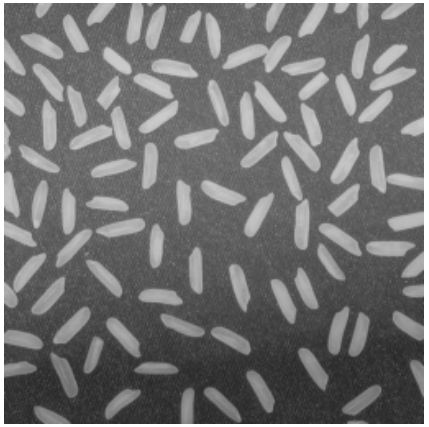
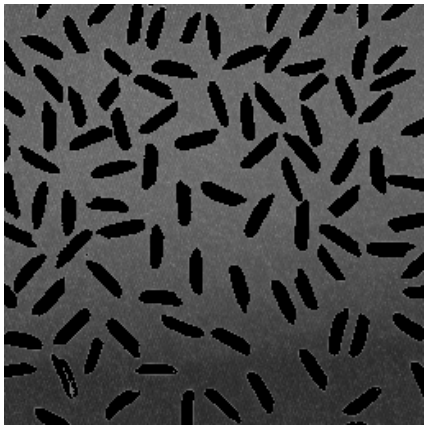
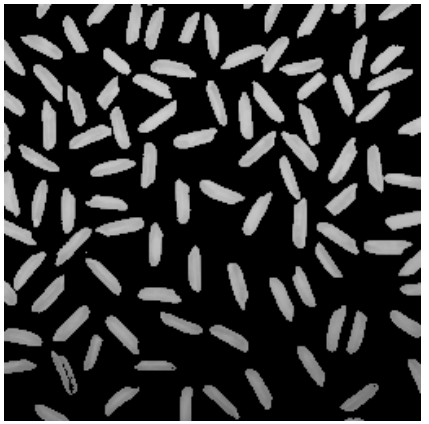
```
>> I = imread('rice.png');
>> J = imread('cameraman.tif');
>> K = uint8(1/2*double(I)+1/2*double(J));
```



Subtracting Images

- Subtract background of a scene

```
rice = double(imread('rice.png'));
thres = diag(linspace(160,110,size(rice,1))) * ones(size(rice));
idx = find(rice < thres);
bg = zeros(size(rice)); bg(idx) = rice(idx);
rice2 = uint8(rice – bg);
```

Downsize image

- Shrinkage image

```
I = imread('cameraman.tif');
rate = 0.5;
step = 1 / rate;
X = 1:step:size(I,1); Y = 1:step:size(I,2);
I2 = I (X,Y);
```

Anti-aliasing

- Removing aliasing effect

```
I = double(imread('cameraman.tif'));
rate = 0.5; step = 1 / rate;
X = 1:step:size(I,1); Y = 1:step:size(I,2);
K = 1/16 * [1 2 1;2 4 2; 1 2 1];
I2 = conv2(I,K,'same');
I2 = uint8(I2(X,Y));
```

What happen is step is not intege

- Modified code

```
I = double(imread('cameraman.tif'));
rate = 0.3; step = 1 / rate;
X = round(1:step:size(I,1)); Y = round(1:step:size(I,2));
K = 1/16 * [1 2 1;2 4 2; 1 2 1];
I2 = conv2(I,K,'same');
I2 = uint8(I2(X,Y));
```

Project package for image downsampling

- List of items
  - imdowsize.m
  - demo.m
  - cameraman.png

imdownsize.m

```
function im2 = imdownsize(im, rate)
% IMDOWNSIZE downsize image.
% IM2 = IMRESIZE(IM, RATE) returns an image that is
% RATE times the size of IM, which is a grayscale image.

S = size(im);
if length(S)==3, disp('Error: color image is not supported'); return 0; end
if rate >1, disp('Error: upsize image is not supported'); return 0; end
step = 1 / rate;
X = round(1:step:S[1]);
Y = round(1:step:S[2]);
K = 1/16 * [1 2 1;2 4 2; 1 2 1];
im2 = conv2(I,K,'same');
im2 = uint8(im2(X,Y));
```

demo.m

```
I = imread('cameraman.png');
I2 = imdownsize(I,0.5);
imwrite(I2,'cameraman_0.5.png')
imshow(I2);
```