Introduction to MATLAB (Basics)

Reference from:
Azernikov Sergei
mesergei@tx.technion.ac.il

MATLAB Basics

- Where to get help?

1) In MATLAB's prompt type:

   help, lookfor,helpwin, helpdesk, demos.

2) On the Web:

   http://www.mathworks.com/support

MATLAB's Workspace

- **who,whos** - current variables in workspace
- **save** - save workspace variables to *.mat file
- **load** - load variables from *.mat file
- **clear all** - clear workspace variables

Matrices in MATLAB

- Matrix is a main MATLAB's data type
- How to build a matrix?

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

Creates matrix A with size 3x3.

- Special matrices :

  zeros(n,m), ones(n,m),eye (n,m)

Basic Operations on Matrices

- All the operators in MATLAB defined on matrices : **+, -, *, /, ^, sqrt, sin, cos** etc.
- Element wise operators defined with preceding dot : **.*, ./, .^ .**
- size(A) - size vector
- sum(A) - columns sums vector
- sum(sum(A)) - all the elements sum

Logical Conditions

- **== , < , > , *(not equal)*~= ,*(not)*~**

- **find('condition') -** Returns indexes of A's elements that satisfies the condition.

Logical Conditions(cont.)

- Example:

```
>> A = [1 2; 3 4], I = find(A<4)

A =

    1    2
    3    4


I =

    1
    2
    3
```

Flow Control

- MATLAB has five flow control constructs:
  - if statements
  - switch statements
  - for loops
  - while loops
  - break statements

if

- **IF** statement condition.

  The general form of the **IF** statement is

  IF expression

  statements

  ELSEIF expression

  statements

  ELSE

  statements

  END

if(cont.)

- Example:

```
if I == J
  A(I,J) = 2;
elseif abs(I-J) == 1
  A(I,J) = -1;
else
  A(I,J) = 0;
end
```

switch

- **SWITCH -** Switch among several cases based on expression.
- The general form of the SWITCH statement is:

```
SWITCH switch_expr
  CASE case_expr,
    statement, ..., statement
  CASE {case_expr1, case_expr2, case_expr3,...}
    statement, ..., statement
 ...
  OTHERWISE,
    statement, ..., statement
END
```

switch (cont.)

- Note:

Only the statements between the matching
CASE and the next CASE, OTHERWISE, or END are executed.
Unlike C, the SWITCH statement does not fall through
(so BREAKs are unnecessary).

for

- **FOR** Repeat statements a specific number of times.

- The general form of a FOR statement is:

  FOR variable = expr, statement, ..., END

for (cont.)

- Example:

```
FOR I = 1:N,
        FOR J = 1:N,
            A(I,J) = 1/(I+J-1);
        END
END
```

while

- **WHILE** Repeat statements an indefinite number of times.
- The general form of a **WHILE** statement is:

```
WHILE expression
    statements
END
```

while (cont.)

- Example:

```
E = 0*A; F = E + eye(size(E)); N = 1;
while norm(E+F-E,1) > 0,
        E = E + F;
        F = A*F/N;
        N = N + 1;
end
```

Scripts and Functions

- There are two kinds of M-files:
  - **Scripts**, which do not accept input arguments or return output arguments. They operate on data in the workspace.
  - **Functions**, which can accept input arguments and return output arguments. Internal variables are local to the function.

Functions in MATLAB

- **FUNCTION** Add new function.

- New functions may be added to MATLAB's vocabulary if they are expressed in terms of other existing functions.

Functions in MATLAB (cont.)

- Example :

  The existence of a file

  on disk called STAT.M with:

```
function [mean,stdev] = stat(x)
%STAT Interesting statistics.
n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);
```

  defines a new function called STAT that calculates the mean and standard deviation of a vector.

Visualization and Graphics

- **plot(x,y), plot(x,sin(x))** - plot 1-D function
- **figure , figure(k)** - open a new figure
- **hold on, hold off** - refreshing
- **mesh(x_ax,y_ax,z_mat)** - view surface
- **contour(z_mat)** - view z as top. map
- **subplot(3,1,2)** - locate several plots in figure
- **axis([xmin xmax ymin ymax])** - change axes
- **title('figure title')** - add title to figure

Image Proc. with MATLAB

(Please refer to Matlab Demo for more details of Image Processing Tool Box)

What Is the Image Processing Toolbox?

- The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB ® numeric computing environment. The toolbox supports a wide range of image processing operations, including:
  - Geometric operations
  - Neighborhood and block operations
  - Linear filtering and filter design
  - Transforms
  - Image analysis and enhancement
  - Binary image operations
  - Region of interest operations

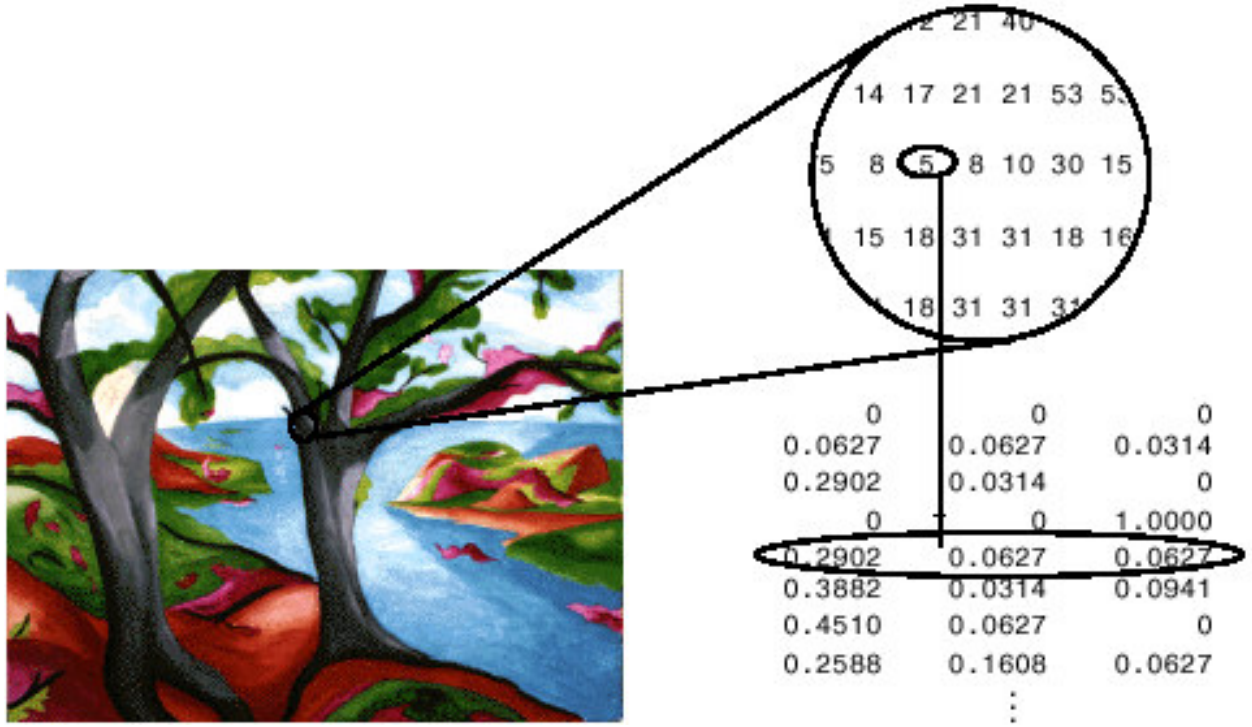
MATLAB Image Types

- Indexed images : m-by-3 color map
- Intensity images : [0,1] or uint8
- Binary images : {0,1}
- RGB images : m-by-n-by-3

Indexed Images

» [x,map] = imread('trees.tif');

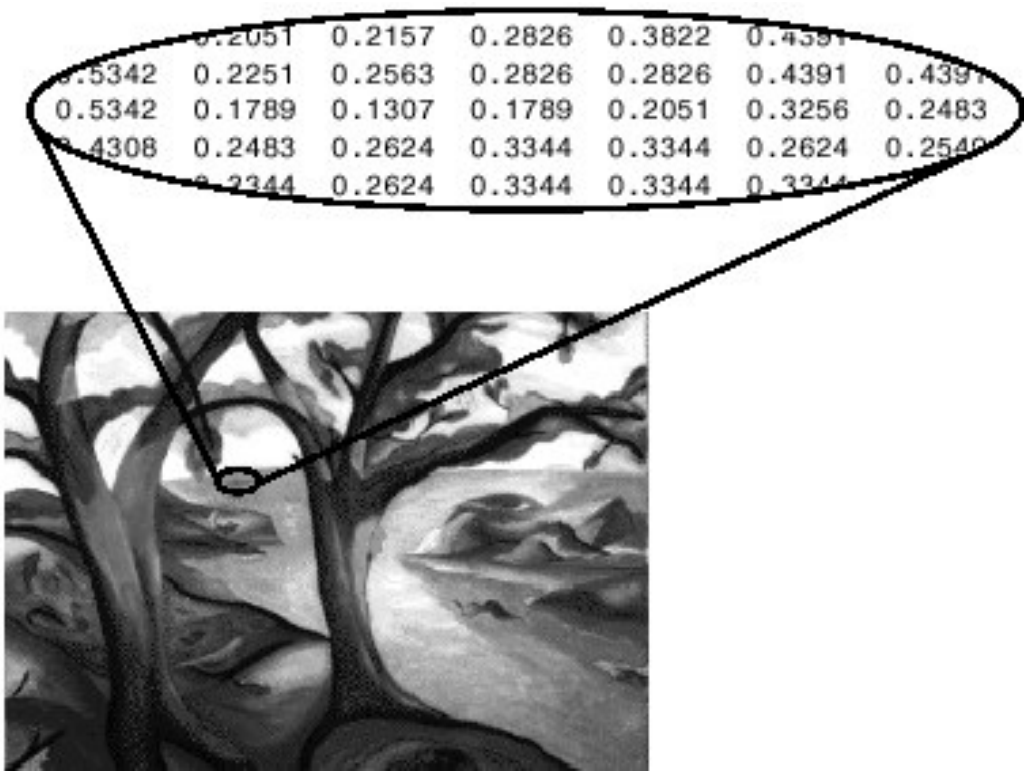
» imshow(x,map);



Intensity Images

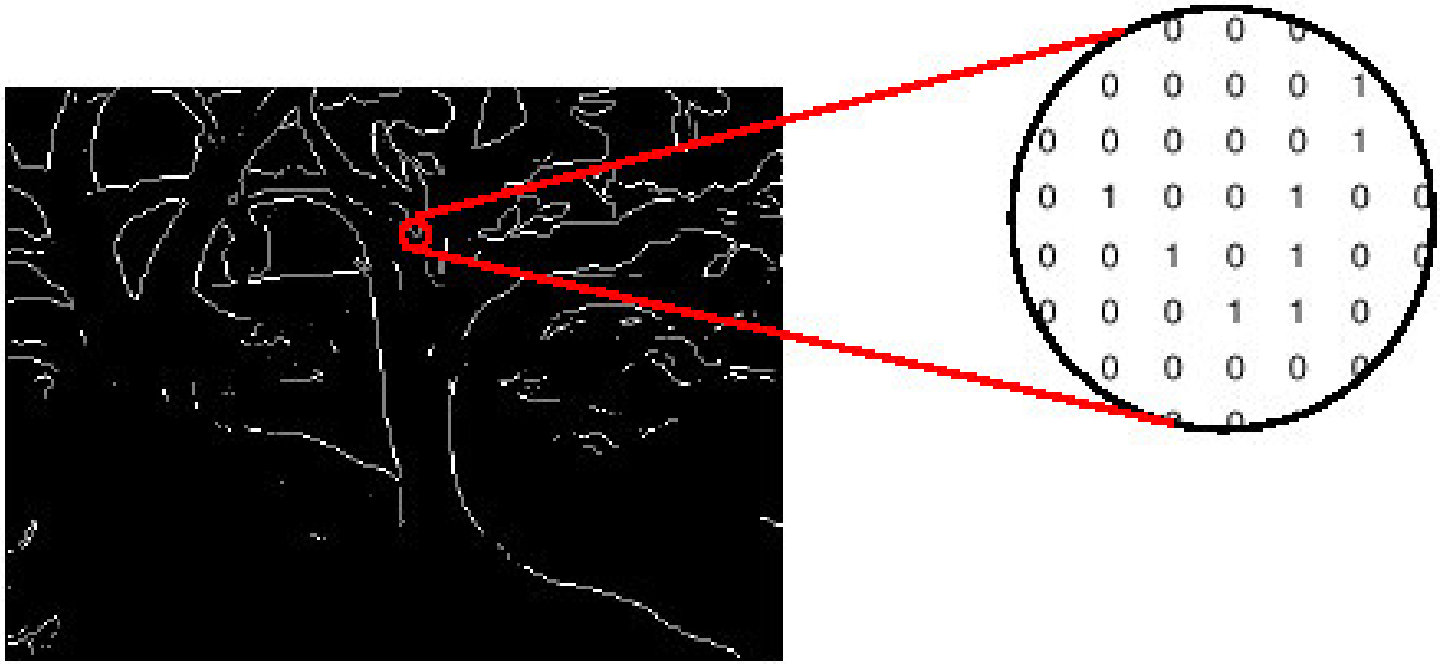
» image = ind2gray(x,map);

» imshow(image);



Binary Images

» **imshow(edge(image));**



RGB Images

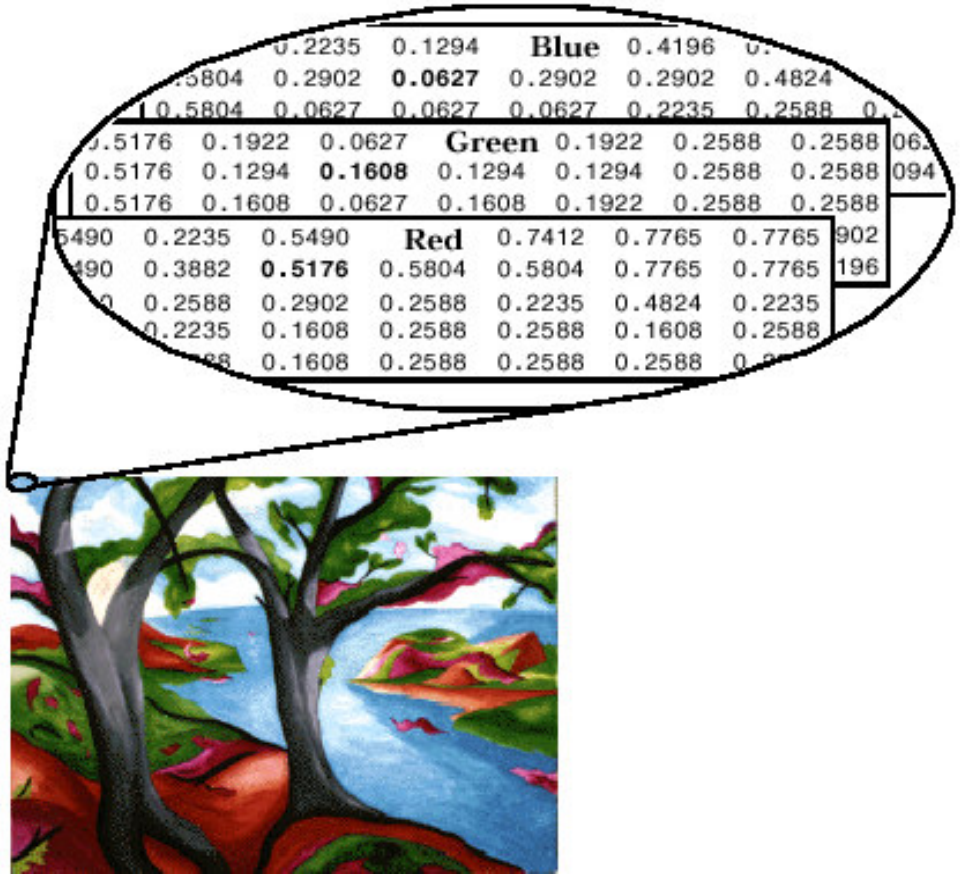

0.2235 0.1294 Blue 0.4196
.5804 0.2902 0.0627 0.2902 0.2902 0.4824
0.5804 0.0627 0.0627 0.0627 0.2235 0.2588
.5176 0.1922 0.0627 Green 0.1922 0.2588 0.2588
0.5176 0.1294 0.1608 0.1294 0.1294 0.2588 0.2588
0.5176 0.1608 0.0627 0.1608 0.1922 0.2588 0.2588
5490 0.2235 0.5490 Red 0.7412 0.7765 0.7765
490 0.3882 0.5176 0.5804 0.5804 0.7765 0.7765
0.2588 0.2902 0.2588 0.2235 0.4824 0.2235
0.2235 0.1608 0.2588 0.2588 0.1608 0.2588
0.1608 0.2588 0.2588 0.2588

Image Display

- **image** - create and display image object
- **imagesc** - scale and display as image
- **imshow** - display image
- **colorbar** - display colorbar
- **getimage**- get image data from axes
- **truesize** - adjust display size of image
- **zoom** - zoom in and zoom out of 2D plot

Some Points to Note

Pixel values are accessed as matrix elements.

2D Image with intensity values: `I(row,col)`

2D RGB images `I(row,col,color)`

- Color : Red = 1; Green = 2 ; Blue = 3

Displaying images

```
figure, imshow(I)
```

Displaying pixel position and intensity information

```
pixval on
```

Points to Note

- All arithmetic operations performed on matrices may be performed on images
- After processing, an image matrix can be written to an output image file with the `imwrite` function
  - **`imwrite(I,map,'filename','fmt')`**
- Without the map argument, the image data is supposed to be grayscale or RGB.
- The format 'fmt' needs to support the particular type of image

Image Conversion

- **gray2ind** - intensity image to index image
- **im2bw** - image to binary
- **im2double** - image to double precision
- **im2uint8** - image to 8-bit unsigned integers
- **im2uint16** - image to 16-bit unsigned integers
- **ind2gray** - indexed image to intensity image
- **mat2gray** - matrix to intensity image
- **rgb2gray** - RGB image to grayscale
- **rgb2ind** - RGB image to indexed image

% Working with Images (example)

```
[I,map]=imread('trees.tif');                    % read a TIFF image
figure, imshow(I,map)                           % display it as indexed image
I2=ind2gray(I,map);                             % convert it to grayscale
figure
colormap('gray')                                % use gray colormap
imagesc(I2,[0 1])                               % scale data to use full colormap
                                                % for values between 0 and 1
axis('image')                                   % make displayed aspect ratio %proportional
                                                % to image dimensions
I=imread(‘moon.jpg'); % read a JPEG image into 3D %array
figure
imshow(I)
rect=getrect;                                   % select rectangle
I2=imcrop(I,rect);                              % crop
I2=rgb2gray(I2);                                % convert cropped image to grayscale
imagesc(I2)                                     % scale data to use full colormap
```

```
% between min and max values in I2
colormap('gray')
colorbar                    % turn on color bar
pixval                      % display pixel values interactively
truesize                    % display at resolution of one %screen pixel
                            % per image pixel
truesize(2*size(I2))        % display at resolution of two %screen pixels
                            % per image pixel
I3=imresize(I2,0.5,'bil');  % resize by 50% using bilinear
                            % interpolation
I3=imrotate(I2,45,'bil');   % rotate 45 degrees and crop to
                            % original size
I3=double(I2);              % convert from uint8 to double, to %allow
% math operations
imagesc(I3.^2)              % display squared image (pixel-wise)
imagesc(log(I3))            % display log of image
```

MATLAB Resources on the Internet

http://www.mathworks.com/products/demos/#

http://www.math.siu.edu/MATLAB/tutorials.html

http://math.ucsd.edu/~driver/21d -s99/MATLAB-primer.html

http://www-cse.ucsd.edu/~sjb/classes/MATLAB/MATLAB.intro.html

http://www.mit.edu/~pwb/cssm/

http://www.mathworks.com

Interesting and very complete tutorials in:
http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

http://www.mathworks.com/matlabcentral/fileexchange

Getting started with MATLAB

http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.shtml

MATLAB tutorial

http://www.math.mtu.edu/~msgocken/intro/intro.html

http://amath.colorado.edu/scico/tutorials/matlab/

MATLAB helpdesk

http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml

MATLAB Primer

ftp://ftp.eng.auburn.edu/pub/sjreeves/matlab_primer_40.pdf