Image Processing CS 6640 : An Introduction to MATLAB Basics

Bo Wang and Avantika Vardhan

August 29, 2014

1 Getting Started with MATLAB

1.1 Resources

1) CADE Lab: Matlab is installed on all the CADE lab machines. If you wish to remotely work on the cade machines perform the following steps:
For Windows Mac Users:
CADE lab tutorial on Remote Access
For Linux Users:
Once you have your CADE username and password , you need to run the following command in your linux machine.
ssh -X username@lab1-2.eng.utah.edu
Once you see the welcome message just type matlab. If you still have problems you can try replacing the -X with -Y.
2) Marriott Library: Here is the link which provides you with detailed instructions :
Marriott Tutorial on Remote Access to MATLAB
On the above link (once you have installed the latest version of Java) , click on the red "Log in" tab.
You can then log in with your UNID. MATLAB appears on a menu on the left hand side.
It is compatible with OS X, Windows, and Linux. However, it might be a bit slow.

3) Student Version available online.

1.2 Starting MATLAB

Windows users need to navigate to the MATLAB install directory and execute the Matlab application matlab from the **MATLAB/bin/** directory.
Linux users can run the same by running **./matlab** from the **MATLAB/bin/** directory.

2 Getting Help for MATLAB

The best source of information is the MATLAB help which is installed with the original product. If you are using the graphical interface navigate to the help menu and click on "Product Help". Search for the command or keyword. If you are running the console version of MATLAB, and want the details regarding a particular function, use the command

help functionname

Another excellent source is the mathworks website `http://www.mathworks.com/help/techdoc/`. The website has a examples, tutorials, user guides and a lot more.

3 Running Commands and Writing Programs in MATLAB

1) Running commands from Command Line
Load images into the workspace.
2) Writing a basic program in MATLAB
clc; Clears Command window
clear all; Clears workspace
close all; Closes all figures
3) Save the program as a .m file, eg. "FirstMatlabCode.m"
4) Run the code by pressing "F5" or alternately clicking on the green pointer button on the programming window.

4 Basic MATLAB commands

1) Declaring a numeric variable
Var1 = 5; The numeric value 5 is stored in Var1.
2) Declaring a string variable
Str1 = $'string'$**;**
3) Displaying a string or text
disp($'$ **The value of the variable is** $'$**);**
4) Displaying a variable
disp(Var1)
Output : 5
5) Size of variable
size(Var1) , or it is only a vector **length(Var1)**
Output : 1
6) Details of a variable
whos Str1
Output :

Name	Size	Bytes	Class	Attributes
str1	1x6	12	char	

7) Maximum, Minimum, Mean, and Variance of a variable
Maximum : **max(VariableName);**
Minimum : **min(VariableName);**
Mean : **mean(VariableName);**
Variance : **var(VariableName);**

5 Example 1: MATLAB File using basic commands :

```
clc ;
close all;
clear all;
Var2 = [1,2,3,4];
disp('Variable is');
disp(Var2);
whos Var2 ;
size1 = size(Var2) ;
disp('Size of the Variable is');
disp(size1);
Max1 = max(Var2);
disp('Maximum Value is');
disp(Max1);
Min1 = min(Var2);
disp('Minimum Value is');
disp(Min1);
Mean1 = mean(Var2); disp('Mean Value is');
disp(Mean1);
Variance1 = var(Var2);
disp('Variance is');
disp(Variance1);
```

6 Reading and writing images

Images are treated as matrices in MATLAB. The first simple command that you need to know to know is:
imageMatrix = imread('fileName');
This command reads the image file and writes it out into the matrix "imageMatrix". If the image file is grayscale, the matrix has dimensionality $[n \times m]$. If the image is a 3 channel color image, then the dimensionality of the matrix is $[n \times m \times 3]$.

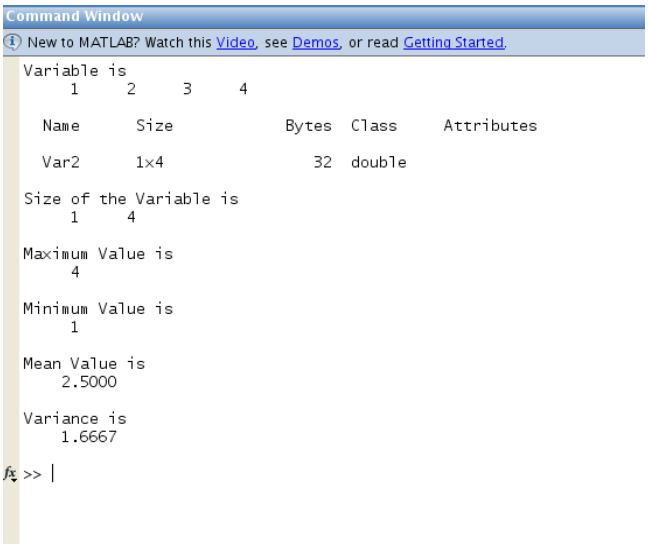

Figure 1: Output of MATLAB Basic Commands code.

n is the height of the image while m is the width.
If you wish to modify the value at a certain pixel location you need to access it by its index. Note that the matrix indexing starts from 1 and not 0. For example if you wish to change the pixel value at (10,10) of "imageMatrix" then we can do it as:
imageMatrix(10,10) = 30;
To access a sub-portion of the image (for example access pixels with x indices 10 to 20, y indices 10 to 15 of the image) and write it into a new image, the command is :
subMatrix = imageMatrix(10:20,10:15);
There are various ways of selecting subsets of matrices to operate upon. For a better description go to `http://www.mathworks.com/company/newsletters/articles/Matrix-Indexing-in-MATLAB/matrix.html` .

6.1 Read image from file and save to file

ImageMatrix = imread('image1.png'); Reads image.
size(ImageMatrix); Finds size of the image.
image(ImageMatrix); Displays image.
imshow(ImageMatrix); Displays image, image processing toolbox needed to use this.
imwrite(ImageMatrix, 'image1new.png'); Writes image into a png file.

6.2 Column and row operations

ImgC1 = ImageMatrix(:,:,1); Assigns first channel of color image to a second image.
size(ImgC1); Size of image is shown.
imagesc(ImgC1); Displays the image.
colormap(gray); The image colormap by default is color, and can be set to gray.
NOTE : More on image display in a later section.

7 Converting a color image to grayscale

On some occassions, you may require to convert a color image to grayscale. This can be done using the following two ways:
rgbImage = imread('someColorImage');
grayImage1 = rgb2gray(rgbImage); rgb2gray requires Image Processing Toolbox
grayImage2 = 0.2989*rgbImage(:,:,1) + 0.5870*rgbImage(:,:,2) + 0.1140*rgbImage(:,:,3);
This first method is an inbuilt matlab function. The second command we explicitly use the colorspace conversion transform. Depending upon the colorspace we are dealing with, we can use the relevant colorspace conversion transform.

8 Displaying images

MATLAB graphics are generally displayed in graphics object called "figure". The command to invoke a new object is simply
figure ;
To create a new figure with a specific index no (for example, 1) the command is :
figure(1);
The overloaded calls of the function allow you to access various properties and the handle to the graphics object. You can display various objects in a figure. Let us look at the methods to display images in the figure. There are two major ways of displaying images. The first simple way is:
image(imageMatrix);
OR
imshow(imageMatrix); (imshow requires the Image Processing Toolbox).
The above method does not scale the image data to occupy the entire colormap. If you wish to scale the data appropriately you can use:
imagesc(imageMatrix);
The above commands display a single image in a figure window. To display multiple objects in the same figure use:
figure(1);

subplot(n,m,i); imagesc(imageMatrix1);
subplot(n,m,j); imagesc(imageMatrix2);
subplot(n,m,l); imagesc(imageMatrix3);
The command "subplot" breaks the figure window into an $n \times m$ grid of axes. Each axis object is counted from left to right, top to bottom starting from the top left.

8.1 Scattering points on an already displayed image :

In order to add something more to an already displayed image it is necessary to use the command "hold on" after the figure is called. Any subsequent displayed commands will then be added to the already opened figure.
figure(1);
hold on;
imagesc(imageMatrix1);
scatter(x,y);
NOTE : The axis labeling when "scatter" is used.

9 Writing images to files

To write matrix data into common image file formats use the command:
imwrite(imageMatrix, 'fileName', 'fileFormat')
This command will write the image matrix to the file specified in the desired file format.

10 Read cursor position and values

[x,y] = ginput(n)
enables you to select n points from the current axes and returns the x- and y-coordinates in the column vectors x and y, respectively. You can press the Return key to terminate the input before entering n points.
[x,y] = ginput
gathers an unlimited number of points until you press the Return key.

11 Pixel operation

11.1 Thresholding an image (binarize) with specified threshold

Example: Suppose we have a image which size is 3×5, we want to threshold this image using 0.5. The following code does it and we set all pixels which are larger than the threshold as 1, the pixels which are smaller or equal to the threshold as 0.

```
I = rand(3,5);
I(I>0.5) = 1;
I(I<=0.5) = 0;
```

11.2 Scaling intensity values.

Example: Suppose we have a image which size is 4×7, the range of the image is $[0, 1]$, we want to rescale the image range to $[0, 255]$. The following is the code to do this.

```
I = rand(4,7);
Inew = I*255;
```

12 Example 2: MATLAB File using basic Image functions :

```
clc;
clear all;
close all;
ImageMatrix = imread('puppy.jpg');
figure(1); imagesc(ImageMatrix);
ImageMatrixGray = 0.2989*ImageMatrix(:,:,1) + 0.5870*ImageMa-
trix(:,:,2) + ImageMatrix(:,:,3);
SubImageMatrix = ImageMatrix(400:600,100:500);
figure(2);
subplot(2,2,1); imagesc(ImageMatrix);
subplot(2,2,2); imagesc(ImageMatrixGray); colormap(gray);
subplot(2,2,3);imagesc(SubImageMatrix);
subplot(2,2,4); imagesc(ImageMatrix); hold on; scatter(480,680,'y','o');
imwrite(ImageMatrix,'halfpuppy.jpg','jpg');
xinput = ginput(1);
disp('Pixel clicked on is');
disp(xinput);
```

NOTE : The directory in which you store images, files, functions, etc. that you are using in your text needs to be the same as the directory you are running your code from. Else, a path to the files needs to be explicitly added (The command used is **addpath** - MATLAB Help has more details).

13 Data Types

When performing any type of operation on variables be aware of its data type. Many issues in MATLAB scripts arise due to incorrect usage of a specific data type. imread outputs the data type uint8 or uint16. As a general rule try to

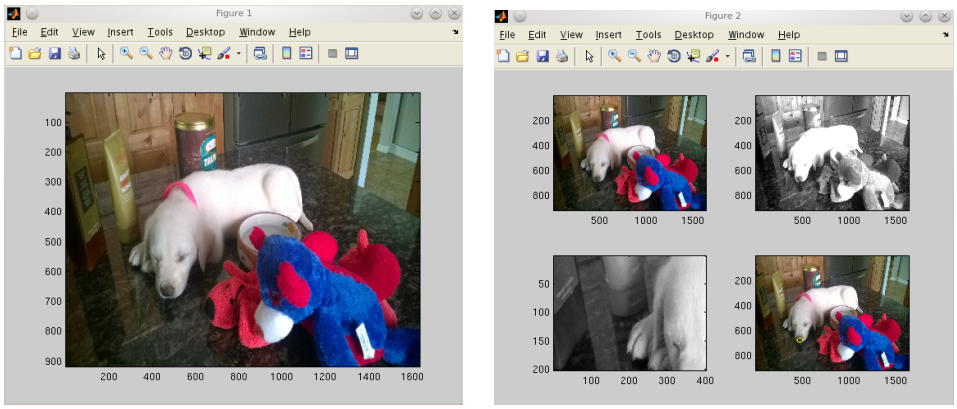

Figure 2: Output of MATLAB Basic Image functions code.

convert your data to a double precision floating point before performing any operations on it. To convert a "uint8" matrix to a "double":
imageMatrix = double(imageMatrix);
To convert it back:
imageMatrix = uint8(imageMatrix);
NOTE : The default variable type in MATLAB is double.
Therefore, if you type :
x = 25.873;
This would mean that x is stored as a double variable automatically.

14 Simple Arithmetic Operations on Matrices

A+B Adds A and B
A-B Subtraction of B from A
A*B Matrix multiplication, the sizes of A, B may not be the same, we just need to make sure A is $m \times n$ and B is $n \times l$
A.*B Array Multiplication (element by element) of 2 matrices, the matrices A and B should be of the same size
A/B Matrix Right Division
A./B Array Right Division
A′ - Transpose of A

15 Relational Operations

A == B Equality
A >= B Greater than equal to

A > B Greater than
A <= B Lesser than equal to
A < B Lesser than
A ~ = B Not equal to
tf = isequal(A,B), tf returns 1 if A,B are equal

16 Logical Operations

expr1 & **expr2 ;** AND operator
expr1 | **expr2 ;** OR operator
~**expr1** NOT operator

17 Example 3: MATLAB File using the Basic Matrix Arithmetic operations :

```
clc;
clear all;
close all;
A = [4, 10 ; 6, 8;];
B = [3, 2; 6, 7;];
disp('Matrix A is');
disp(A);
disp('Matrix B is');
disp(B);
AddMatrix = A + B;
disp('Addition of Matrices A and B is');
disp(AddMatrix);
SubMatrix = A - B;
disp('Subtraction of Matrix B from A is');
disp(SubMatrix);
MultMatrix = A*B ;
disp('Matrix Multiplication of Matrix B with A is');
disp(MultMatrix);
ArrMultMatrix = A.*B ;
disp('Array Multiplication of Matrix A and B is');
disp(ArrMultMatrix);
DivMatrix = A/B ;
disp('Matrix division of Matrix A by B is');
disp(DivMatrix);
ArrDivMatrix = A./B ;
disp('Array Division of Matrix A by B is');
disp(ArrDivMatrix);
```

GreaterThan = (A(1,1)>B(1,1));
disp(' Is element 1,1 of A greater than the element 1,1 of B ? ');
disp(GreaterThan);
LogicalANDOperator = (A(1,1)>B(1,1) A(1,1)>0);
disp('Logical Operation : Is element 1,1 of A greater than element 1,1 of B, and is element 1,1 of A greater than 0 ?');
disp(LogicalANDOperator);

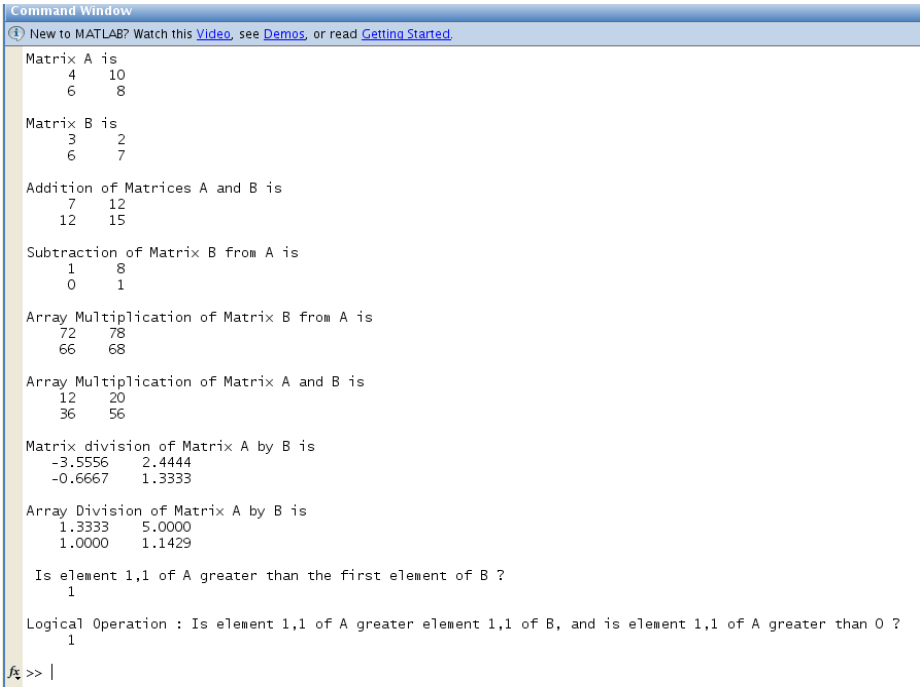

Figure 3: Output of MATLAB Basic Matrix Arithmetic operations code.

18 Generating Random Numbers, Array Creation

Generate a single random number based on the uniform random distribution (by default between 0 and 1):
RandNum = rand;
Generate X × X matrix of random numbers based on the uniform random distribution :
RandSquareMatrix = rand(X) ;
Generate an array of X × Y random numbers based on the uniform random

distribution :
RandMatrix = rand(X,Y) ;
Generate a random number based on the Normal distribution (ranges from 0 to 1) :
RandNumNormal = randn;
Generate a random number based on Gaussian with Mean μ and Standard Deviation σ :
RandNumGaussian = randn*σ + μ ;
Create an Array of ones of dimensions X, Y
ArrOnes = ones(X,Y);
Create an Array of zeros of dimensions X, Y
ArrZeros = zeros(X,Y);

19 Advanced Matrix Operations :

1) Matrix Convolution :
A = rand(3);
B = rand(4);
C = conv2(A,B);
2) Getting Eigen Values and Eigen Vectors of Matrices :
lambda = eig(A);
3) SVD :
[U,S,V] = svd(A);
NOTE : SVD and Eig give results that are sorted differently. In SVD the sorting of eigenvalues is done from high to low and hence it is more convenient to use.
4) Matrix Determinant :
MatDet = det(A);
5) Matrix Inverse :
MatInv = inv(A);

20 Loops and Conditional Statements

1) If, else, elseif :
if expression
statements;
elseif expression
statements;
else
statements ;
end
2) For :
for index = values

program statements
:
end
NOTE : The semicolon is not used to end loops and conditional statements eg. **if (a>b)** does not end with a semicolon.

21 Example 4: MATLAB File using Basic Loops and Conditional statements :

```
clc;
clear all;
close all;
ImageMatrix = imread('puppy.jpg');
figure(1); imagesc(ImageMatrix);
sizeX = size(ImageMatrix,1);
sizeY = size(ImageMatrix,2);
ImageNew = zeros(sizeX, sizeY) ;
for i = 1:1:sizeX
for j = 1:1:sizeY
if(ImageMatrix(i,j,1)>100)
ImageNew(i,j,1) = 1;
elseif(ImageMatrix(i,j,1)<100)
ImageNew(i,j,1) = -1;
else
ImageNew(i,j,1) = 0;
end
end
end
figure(2); imagesc(ImageNew);
```

22 Miscellaneous MATLAB commands :

1) Convert a matrix into a vector :

```
MatrixA = [1 , 2 ; 3 , 4 ;] ;
disp('Matrix A is');
disp(MatrixA);
VecValues = MatrixA(:);
disp('Vector Form of Matrix A is');
disp(VecValues) ;
```

Output :
MatrixA is

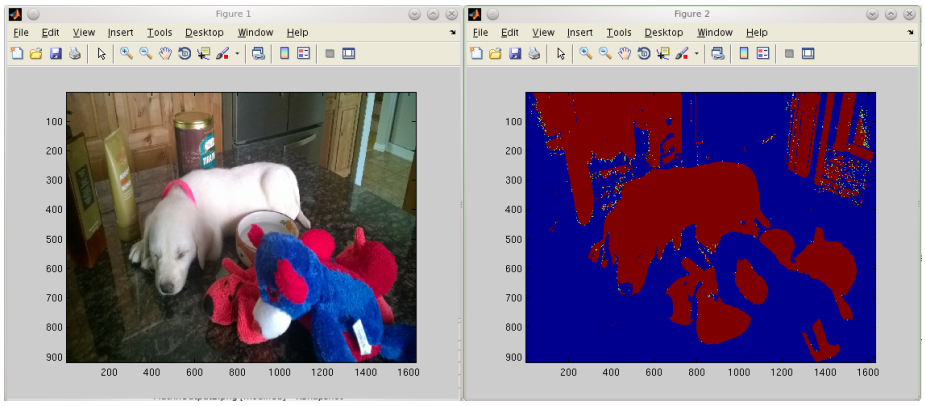

Figure 4: Output of MATLAB Basic Loops and Conditional Statements code.

MatrixA =
1 2
3 4
VecValues is
VecValues =
1
2
3
4

2) Reshape a Vector into a matrix of dimensions X × Y

X = 2;
Y = 2;
MatrixReturns = reshape(VecValues,X,Y) ;
disp('New Matrix is');
disp(MatrixReturns);
Output :
New Matrix is
MatrixReturns =
1 2
3 4

3) Plot a result :

figure;
plot(x,y); Plot the x and y points
plot(x,y,'*'); Plot the x and y points, with asterix as the line type.
plot(x,y,'*','r'); Plot the x and y points, line type asterix and red color.
title('First Plot in Image Processing Course'); Adds title to the plot.
xlabel('X Axis'); Label for the X axis.
ylabel('Y Axis'); Label for the Y axis.
legend('x','y ') Legends are added .
bar(x,y); This draws bars for each column in y for every x value.