Introduction to MATLAB with Image Processing.

By: Sutanshu S. Raj, Divij Babbar & Palak Jain.

B.Tech – GGSIPU.

What is MATLAB.?

MATLAB is a numerical computing environment which allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages like C / C++ and Fortran.

VECTORS.

- A vector is defined by placing a sequence of numbers within square braces :

  >> v = [3 1]

  v = 3 1

- To define a set of numbers with a common increment using colons :

  >> v = [1:8]

  v = 1 2 3 4 5 6 7 8

- You can view individual entries in the vector.

  >> v(1)

  ans = 1

Vectors (Cont.)

- To simplify the creation of large vectors :

  >> v = [0:2:8]

  Matlab will automatically figure out how many entries you need and their values.

- You can look at specific parts of the vector:

  >> v(1:3)

  ans = 0 2 4

  >> v(1:2:4)

  ans = 0 4

  >>v(1, :) % First Row, all columns

  ans = 0 2 4 6 8

  >>v(: , 3) ; ans = ?

Vectors (cont.)

- You can perform mathematical operations on the vectors:

```
>> v(1:3)-v(2:4)
ans = -2  -2  -2
>> u = [0:-1:-4]
>> u+v
>> -2*u
>> v/3
>> u+v'
??? Error using ==> plus
Matrix dimensions must agree.
```

MATRICES.

- Defining a matrix:

  >> A = [1 2 3; 3 4 5; 6 7 8] % Row-major form.

- Work with different indexes of a matrix :

  >> A(1:2,2:3)

  >> A(1:2,3:4)

  ??? Index exceeds matrix dimensions.

- Perform many standard operations on the matrix. For e.g, you can find the inverse of a matrix :

  >> inv(A)

  >> inv(a) % Case – Sensitive.

  ??? Undefined function or variable a.

Matrices (cont.)

- You can find an approximation to the eigen values of a matrix:

  >> eig(A)

  >> [v,e] = eig(A)

  >> diag(e)

- To distinguish the difference between solving systems that have a right or left multiply, Matlab uses two different operators, "/" and "\".

  >> v = [1 3 5]'

  >> x = A\v

  >> x = B\v

  >> B*x

  >> x1 = v'/B

Vector Operations.

- You can do "element-wise" operations on the matrices :

  >> v.*b

  >> v./b
- You can pass a vector to a predefined math function :

  >> sin(v)

  >> log(v)
- Complex operations can be defined and can be done quickly :

  >> x = [0:0.1:100]

  >> y = sin(x).*x./(1+cos(x));
- Matlab will let you plot the graph of the results :

  >>plot(x , y)

  >> help plot

LOOPS.

- If you want to repeat some action in a pre-determined way, you can use the *for loop :*

```
>> for j=1:4,
   j
   end
```

- We want to perform operations on the rows of a matrix :

```
>> A = [ [1 2 3]' [3 2 1]' [2 1 3]']
>> B = A;
>> for j=2:3,
     A(j,:) = A(j,:) - A(j-1,:)
end
```

Loops (cont.)

- Gaussian Elimination can be performed using only two loops and one statement :

```
>> for j=2:3,
       for i=j:3,
          B(i,:) = B(i,:) - B(j-1,:)*B(i,j-1)/B(j-1,j-1)
       end
 end
```

- The *while loop* repeats a sequence of commands as long as some condition is met. This can make for a more efficient algorithm.
- The *for loop* can use up a lot of memory just creating the vector used for the index. A better way of implementing the algorithm is to repeat the same operations but only as long as the number of steps taken is below some threshold.

EXECUTABLE FILES.

- The easiest editor on our system is the built in matlab editor. It allows you to do some very simple file manipulations.
- Matlab executable files (called M-files) must have the extension ".m".
- Type in the following command to create the file, at the matlab prompt:
  >> edit simpleEuler.m
- To get Matlab to execute the commands in the m-file, simply type in "simpleEuler“ in the command prompt.
- NOTE : You must first specify all of the variables that are not defined in the file itself.

SUB-ROUTINES.

- Subroutines are just like executable files, but you can pass it different vectors and matrices to use.
- When we try to find *x* given that *Ax=b*, the routine needs matrix A and the vector b, and it will pass back the vector x. If the name of the file is called *gaussElim.m*, then the first line will look like this:
  function [x] = gaussElim(A , b)
- To execute above sub-routine, you give the commands :

  >> A = [1 2 3 6; 4 3 2 3; 9 9 1 -2; 4 2 2 1]

  >> b = [1 2 1 4]'

  >> [x] = gaussElim(A,b)

The Image Processing Toolbox (IPT).

- IPT 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 Spatial image transformations, Morphological operations, Neighborhood and block operations, Linear filtering and filter design, Transforms, Image analysis and enhancement, Image registration, Deblurring Region of interest operations, etc.
- Many of the toolbox functions are MATLAB M-files, a series of MATLAB statements that implement specialized image processing algorithms.

Basic Concepts.

- To read an image :

>> I = imread(‘cameraman.tif');

- Now display the image. The toolbox includes two image display functions: imshow and imtool.

>> imshow(I)

- The imtool provides all the image display capabilities of imshow but also provides access to several other tools for navigating and exploring images, such as scroll bars, the Pixel Region tool, Image Information tool, and the Contrast Adjustment tool.

Concepts (cont.) – Improving Image Contrast.

```
>> I = imread('pout.tif');
>> imshow(I)
>> figure; imhist(I)
```

• histeq function is used to spread the intensity values over the full range of the image.

```
>> I2 = histeq(I);
>> figure; imshow(I2)
```

• To write the newly adjusted image I2 to a disk file, use the imwrite function.

```
>> imwrite (I2, 'pout2.png');
```

Introduction to Images.

- MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image.
- Types of Images : *Binary* - Logical array containing only 0's and 1's, interpreted as black and white, respectively.

  Indexed - Array of class logical, uint8, uint16, single, or double whose pixel values are direct indices into a colormap.

  Intensity (or grayscale) - Array of class uint8, uint16, int16, single, or double whose pixel values specify intensity values.
- For a matrix of type uint8, uint16, or int16, the intensity *intmin(class(I))* represents black and the intensity *intmax(class(I))* represents white.

Introduction (contd.)

Converting Between Image Types : *gray2ind* - Create an indexed image from a grayscale intensity image.

grayslice - Create indexed image from a grayscale intensity image by thresholding.

ind2gray - Create grayscale intensity image from an indexed image.

mat2gray - Create grayscale intensity image from data in a matrix, by scaling the data.

- Image Arithmetic : >> I = imread('rice.png');

  >> I2 = imread('cameraman.tif');

  >> K = imdivide(imadd(I,I2), 2);

  >> K = imlincomb(.5,I,.5,I2); % Recommended.

Displaying Images.

```
>> [X1,map1]=imread('forest.tif');
>> [X2,map2]=imread('trees.tif');
>> subplot(1,2,1), imshow(X1,map1)
>> subplot(1,2,2), imshow(X2,map2)
```

- *subimage* converts images to RGB before displaying them and therefore circumvents the colormap sharing problem.
- Displaying multi-frame images : >>mri = uint8(zeros(128,128,1,27));

```
for frame=1:27
   [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end
montage(mri,map);
```

Spatial Transformation – Image Resizing, Rotation & Cropping.

```
>> I = imread('circuit.tif');
>> J = imresize(I,1.25);
>> imshow(I)
>> figure; imshow(J)
```

• You can specify the size of the output image by passing a vector that contains the number of rows and columns in the output image.

```
>> Y = imresize(X,[100 150])
```

• Image Rotation : `>> J = imrotate(I,35);  % counterclockwise.`

• Image Cropping :

```
>> imshow circuit.tif
>> I = imcrop;
>> imshow(I);
```

Linear Filtering & Filter Design.

• Filtering is a technique for modifying or enhancing an image. You can filter an image to emphasize certain features or remove other features.

• Filtering is a *neighborhood operation.*

• *fspecial* function produces several kinds of predefined filters, and filtering of images, either by correlation or convolution, can be performed using *imfilter.*

```
>> I = imread('moon.tif');
>> h = fspecial('unsharp');
>> I2 = imfilter(I,h);
>> imshow(I), title('Original Image')
>> figure; imshow(I2), title('Filtered Image')
```

Linear Filtering & Filter Design (cont.)

- h = fspecial(*type*) creates a two-dimensional filter h of the specified type (average, gaussian, log, motion, unsharp, etc)
- Same concept can be applied to signals (1-D).

```
>> I = imread('cameraman.tif');
>> subplot(2,2,1);  imshow(I);  title('Original Image');
>> H = fspecial('motion',20,45);
>> MotionBlur = imfilter(I,H,'replicate');
>> subplot(2,2,2); imshow(MotionBlur);title('Motion Blurred’);
>> H = fspecial('unsharp');
>> sharpened = imfilter(I,H,'replicate');
>> subplot(2,2,4); imshow(sharpened); title('Sharpened Image');
```

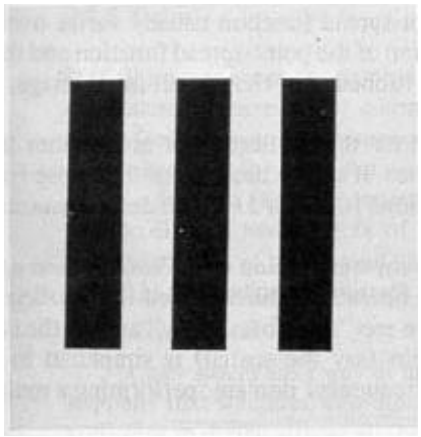
TRANSFORMS.

- The Fourier transform is a representation of an image as a sum of complex exponentials of varying magnitudes, frequencies, and phases.
- The Fourier transform plays a critical role in a broad range of image processing applications, including enhancement, analysis, restoration, and compression.

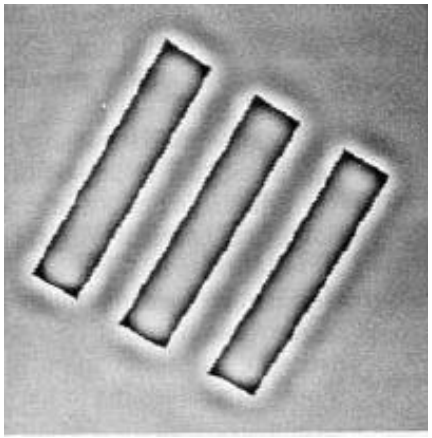
```
>> f = zeros(120, 120);
>> f(25:64, 13:57) = 1;
>> imshow(f, 'notruesize')
>> F = fft2(f);
>> F2 = log(abs(F));
>> imshow(F2,[-1 5],'notruesize');
```

Transforms (cont.)

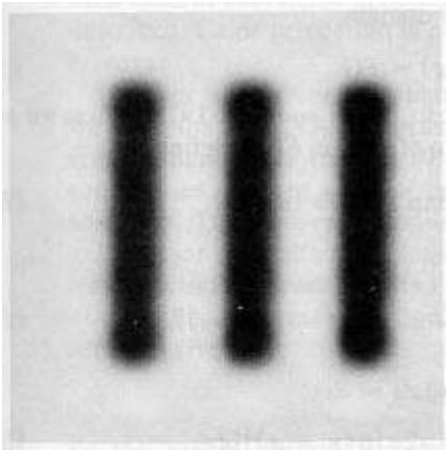
- High frequencies correspond to quickly varying information (e.g., edges).
- Low frequencies correspond to slowly varying information (e.g., continuous surface).



Original Image



High - Passed



Low-Passed

Discrete Cosine Transform.

• It is a separable linear transformation; that is, the two-dimensional transform is equivalent to a one-dimensional DCT performed along a single dimension followed by a one-dimensional DCT in the other dimension.

```
>> RGB = imread('autumn.tif');
>> I = rgb2gray(RGB);
>> J = dct2(I);
>> imshow(log(abs(J)),[])
>> J(abs(J) < 10) = 0;
>> K = idct2(J);
>> imshow(I)
>> figure, imshow(K,[0 255])
```


The End