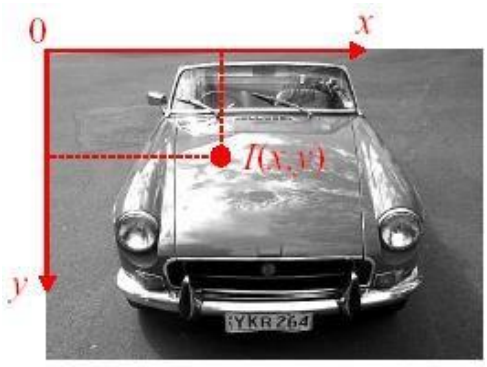
Student Name:_____ Roll No:_____ Section: _____

1. Images as Matrices

The coordinate system in and the preceding discussion lead to the following representation for a digitized image function:



- ✓ An image **I** is a *matrix* of pixel values
- ✓ Pixel value at $\mathbf{p} = [x,y]^T$ is $\mathbf{I}(\mathbf{p})$ or $\mathbf{I}(x,y)$
- ✓ Origin is top left corner
- ○ **I** is a matrix
- ○ **p** is a vector
- ○ x, y, $\mathbf{I}(\mathbf{p})$, $\mathbf{I}(x,y)$ are scalars

The image processing may be done simply by matrix calculation or matrix manipulation.

2. 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

- ✓ 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

3. Read and Display an Image

Clear the MATLAB workspace of any variables and close open figure windows.
clear, close all

To read an image, use the imread command. The example reads one of the sample images included with the Image Processing Toolbox, pout.tif, and stores it in an array named I.

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

Now display the image. The toolbox includes two image display functions: *imshow* and *imview*. You can use either one to display an image.

```
imshow(I)
```

4. Check How the Image Appears in the Workspace

You can also get information about variables in the workspace by calling the whos command.

```
Whos I
Name        Size              Bytes  Class

  I         291x240           69840  uint8 array

Grand total is 69840 elements using 69840 bytes
```

Student Name:_____Roll No:_____Section: _____

5. Write the Image to a Disk File

To write the newly adjusted image I2 to a disk file, use the imwrite function. If you include the filename extension '.png', the imwrite function writes the image to a file in Portable Network Graphics (PNG) format, but you can specify other formats.

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

6. Check the Contents of the Newly Written File

To see what *imwrite* wrote to the disk file, use the *imfinfo* function. This function returns information about the image in the file, such as its format, size, width, and height.

```
imfinfo('pout2.png')
ans =
                    Filename: 'pout2.png'
                FileModDate: '29-Dec-2003 09:34:39'
                    FileSize: 36938
                      Format: 'png'
               FormatVersion: []
                       Width: 240
                      Height: 291
                    BitDepth: 8
                   ColorType: 'grayscale'
```

7. Convert RGB image or Color map to grayscale

```
I = rgb2gray(RGB);
newmap = rgb2gray(map);
```

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. If you have Parallel Computing Toolbox™ installed, rgb2gray can perform this conversion on a GPU.
newmap = rgb2gray(map) returns a grayscale colormap equivalent to map.

Examples

```
RGB = imread('peppers.png');
imshow(RGB)
```

```
I = rgb2gray(RGB);
figure
imshow(I)
```

Student Name:_____Roll No:_____Section: _____

8. Convert RGB Colormap to HSV Colormap

```
cmap = rgb2hsv(M)
hsv_image = rgb2hsv(rgb_image)
```

cmap = rgb2hsv(M) converts an RGB colormap M to an HSV colormap cmap. Both colormaps are m-by-3 matrices. The elements of both colormaps are in the range 0 to 1.

The columns of the input matrix M represent intensities of red, green, and blue, respectively. The columns of the output matrix cmap represent hue, saturation, and value, respectively.

hsv_image = rgb2hsv(rgb_image) converts the RGB image to the equivalent HSV image. RGB is an m-by-n-by-3 image array whose three planes contain the red, green, and blue components for the image. HSV is returned as an m-by-n-by-3 image array whose three planes contain the hue, saturation, and value components for the image.

9. Convert image to binary image, based on threshold

```
BW = im2bw(I, level)
BW = im2bw(X, map, level)
BW = im2bw(RGB, level)
```

BW = im2bw(I, level) converts the grayscale image I to a binary image. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). Specify level in the range [0,1]. This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 is midway between black and white, regardless of class. To compute the level argument, you can use the function graythresh. If you do not specify level, im2bw uses the value 0.5.

BW = im2bw(X, map, level) converts the indexed image X with colormap map to a binary image.

BW = im2bw(RGB, level) converts the truecolor image RGB to a binary image.

If the input image is not a grayscale image, im2bw converts the input image to grayscale, and then converts this grayscale image to binary by thresholding.

Examples

```
load trees
BW = im2bw(X,map,0.4);
imshow(X,map), figure, imshow(BW)
```

10. Convert image, increasing apparent color resolution by dithering

```
X = dither(RGB, map)
X = dither(RGB, map, Qm, Qe)
BW = dither(I)
```

X = dither(RGB, map) creates an indexed image approximation of the RGB image in the array RGB by dithering the colors in thecolormap map. The colormap cannot have more than 65,536 colors.

Student Name:_____Roll No:_____Section: _____

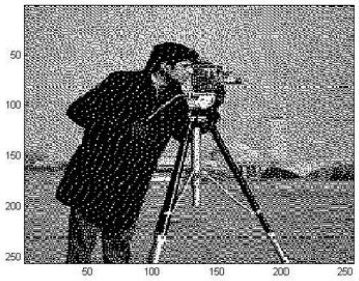
X = dither(RGB, map, Qm, Qe) creates an indexed image from RGB, where Qm specifies the number of quantization bits to use along each color axis for the inverse color map, and Qe specifies the number of quantization bits to use for the color space error calculations. IfQe < Qm, dithering cannot be performed, and an undithered indexed image is returned in X. If you omit these parameters, dither uses the default values Qm = 5, Qe = 8.
BW = dither(I) converts the grayscale image in the matrix I to the binary (black and white) image BW by dithering.

Examples

```
I = imread('cameraman.tif');
figure; imagesc(I);
colormap(gray);
```

Apply dithering to get an indexed image as,

```
BW = dither(I);
figure; imagesc(BW);
colormap(gray);
```



11. uint8, uint16, uint32, uint64

Convert to unsigned integer

Syntax

- I = uint8(X)
- I = uint16(X)
- I = uint32(X)
- I = uint64(X)

Description

I = uint*(X) converts the elements of array X into unsigned integers. X can be any numeric object (such as a double). The results of a uint* operation are shown in the next table.

Operation	Output Range	Output Type	Bytes per Element	Output Class
uint8	0 to 255	Unsigned 8-bit integer	1	uint8
uint16	0 to 65,535	Unsigned 16-bit integer	2	uint16
uint32	0 to 4,294,967,295	Unsigned 32-bit integer	4	uint32
uint64	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer	8	uint64

double and single values are rounded to the nearest uint* value on conversion. A value of X that is above or below the range for an integer class is mapped to one of the endpoints of the range. For example,

- uint16(70000)
- ans =
  65535

A particularly efficient way to initialize a large array is by specifying the data type (i.e., class name) for the array in the zeros, ones, or eye function. For example, to create a 100-by-100 uint64 array initialized to zero, type

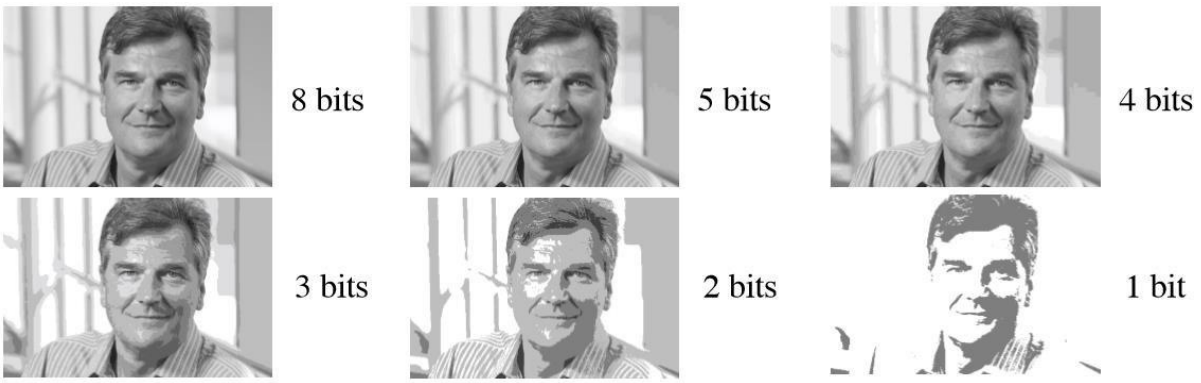
- I = zeros(100, 100, 'uint64');

An easy way to find the range for any MATLAB integer type is to use the intmin and intmax functions as shown here for uint32:

- intmin('uint32') intmax('uint32')
- ans = ans =
- 0 4294967295

12. Image Quantization

Quantization refers to number of bits per pixel of an image. 1:8 bits are available to quantized an image.



„Contouring“

13. Image Brightness and Contrast Adjustment

Student Name:_____Roll No:_____Section: _____

14. Relation of Quantization with gray level resolution:

Gray level resolution of an image can be formed using formula,

$$\square = 2^{\square}$$

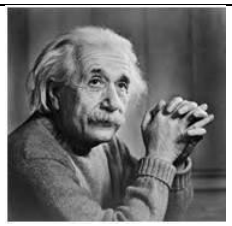
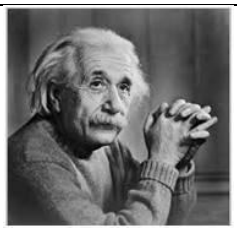
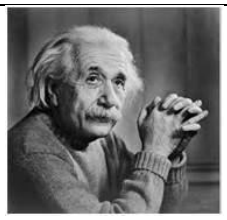
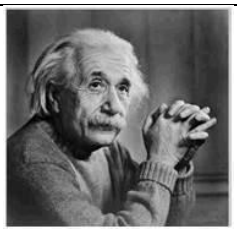
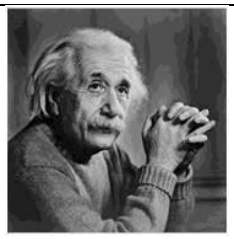
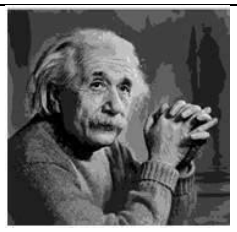
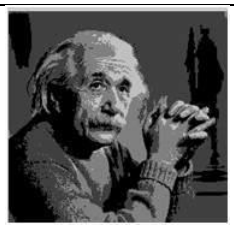
If gray level is equal to 256, we have 256 different shades of gray and 8 bits per pixel, hence the image would be a gray scale image.

Reducing the gray level

Now we will reduce the gray levels of the image to see the effect on the image.

For example

Lets say you have an image of K=8bpp, that has L=256 different levels. It is a grayscale image and the image look something like this:

L=256 Gray Levels K=8		L=128 Gray Levels K=7	
L=64 Gray Levels K=6		L=32 Gray Levels K=5	
L=16 Gray Levels K=4		L=8 Gray Levels K=3	
L=4 Gray Levels K=2		L=2 Gray Levels K=1	

Now before reducing it, further levels, we can easily see that the image has been distorted badly by reducing the gray levels. Now we will reduce it to 2 levels, which is nothing but a simple black and white level.

Student's Tasks:

1. Select any two grayscale/color images and implement the functions you have just practiced
2. Write MATLAB code to show 'face.jpg' for 1: 8 level of quantization
3. Write MATLAB code to show 'parrot.jpg' as increase brightness and 50% contrast using gamma correction
4. Write MATLAB code to show gray levels of gray scale image for $\square = 1:8$