Digital Image Processing

Presented by:

Dr. Moe Moe Myint

moemoemyint@moemyanmar.ml

http://www.slideshare.net/MoeMoeMyint

Information Technology Department

Technological University (Kyaukse)

- Only Original Owner has full rights reserved for copied images.
- This PPT is only for fair academic use.

Dr. Moe Moe Myint
Information Technology Department
Technological University (Kyaukse)

Converting images to other image types (Lab 2)

- The Image Processing Toolbox supports four basic types of images:
  - Indexed images
  - Grayscale images
  - Binary images
  - True Color images

Indexed images

- An indexed image consists of an array and a colormap matrix.
- The pixel values in the array are direct indices into a colormap.
- The colormap matrix is an m-by-3 array of class double containing floating-point values in the range [0,1]. Each row of map specifies the red, green, and blue components of a single color.

Pixel Values Index to Colormap Entries in Indexed Images

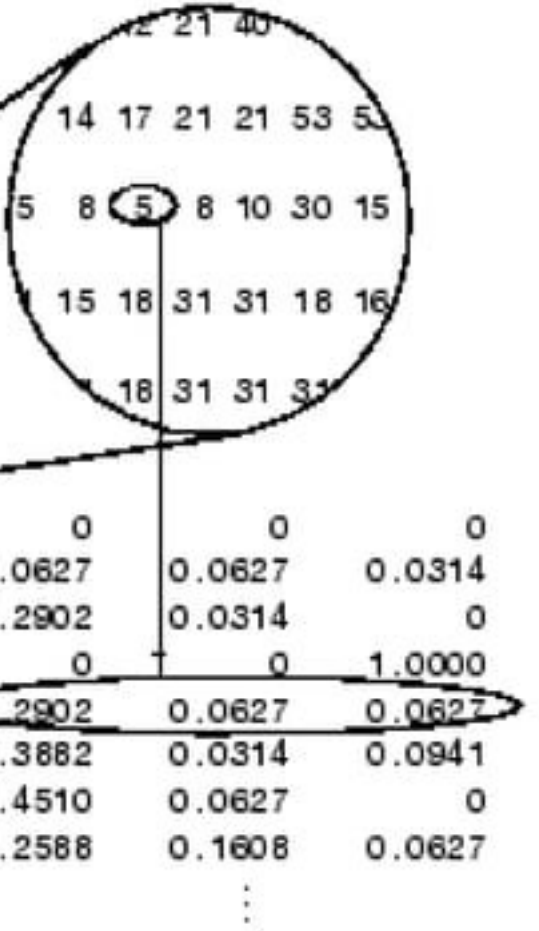
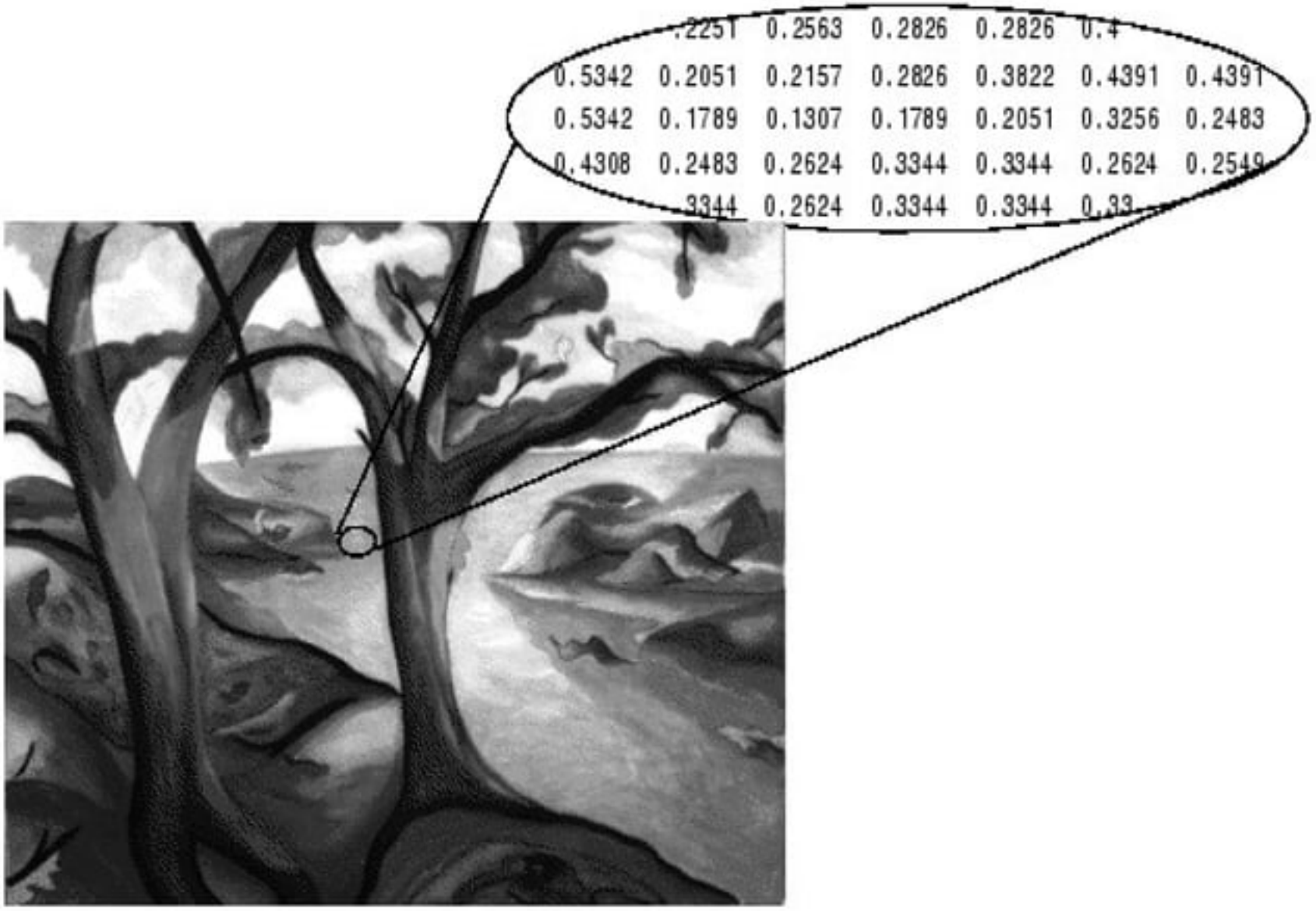


Image Courtesy of Susan Cohen

Grayscale Images

- MATLAB stores a grayscale image as a single matrix, with each element of the matrix corresponding to one image pixel.
- The matrix can be of class uint8, uint16, int16, single, or double in which case it contains values in the range [0,1], i.e, the intensity '0' represents black and the intensity '1' represents white.

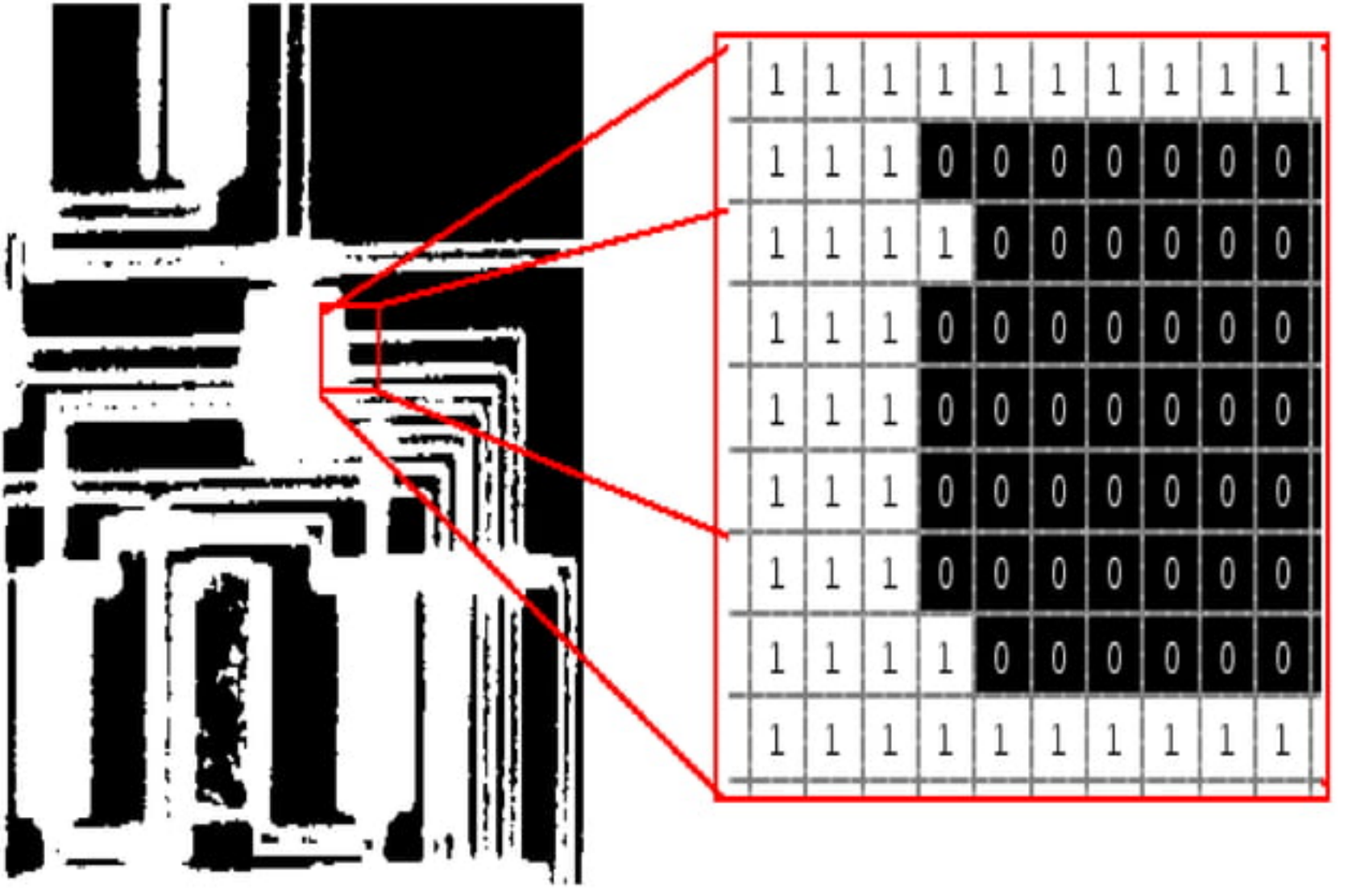
Pixel Values in a Grayscale Image Define Gray Levels



Binary Images

- In a binary image, each pixel assumes one of only two discrete values, "0" or "1".

Pixel Values in a Binary Image



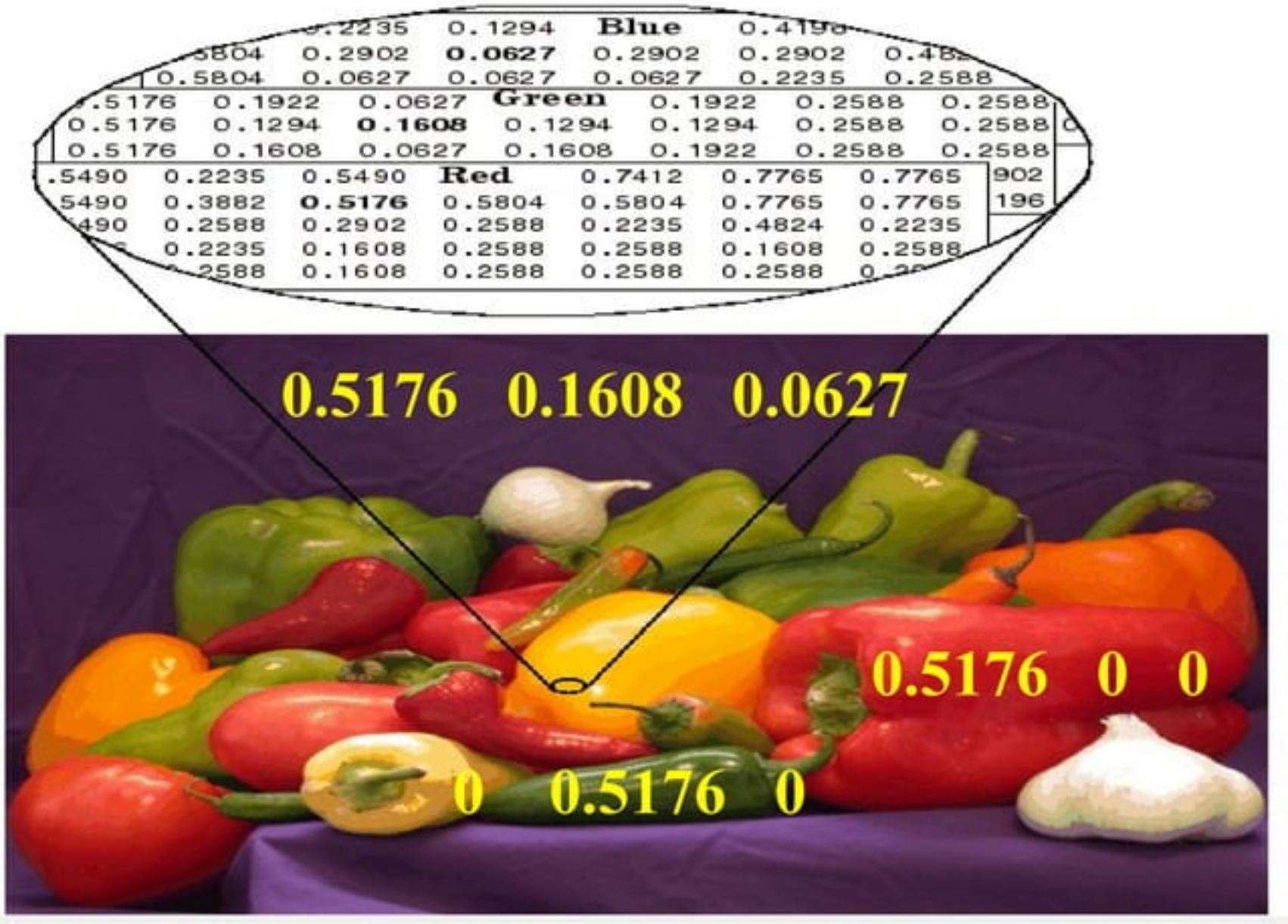
True Color Images

- A true color image is an image in which each pixel is specified by three values — one each for the red, blue, and green components of the pixel's color.

- A true color array can be of class uint8, uint16, single, or double. In a true color array of class single or double, each color component is a value between 0 and 1.

- Graphics file formats store true color images as 24-bit images, where the red, green, and blue components are 8 bits each.

The Color Planes of a Truecolor Image



Color-Space conversions in ImageTransform

- To change a color image into a grayscale image
- I = rgb2gray(RGB)

```
grayimage = rgb2gray('Dog.jpg');
figure; imshow(grayimage);
```

Practice1.m

```
clear all
close all
clc
[f p]=uigetfile('*.jpg');
I=imread([p f]);
figure;imshow(I);
title('Original Image','FontSize',14);
text(500,37,'Picture from Libraries','FontSize',14,'HorizontalAlignment','left');
grayimage=rgb2gray(I);
figure;imshow(grayimage);title('Grayscale Image');
```

Lab 2: Example
Image Types and Type Conversion

- Convert grayscale or binary image to indexed image

  [X, map] = gray2ind(I, n)

- Converts the grayscale image I to an indexed image X. n specifies the size of the colormap, gray(n). **n** must be an integer **between 1 and 65536**. If n is omitted, it defaults to 64.

- Example 1

```
I = imread('cameraman.tif');
[X, map] = gray2ind(I, 16);
imshow(X, map);
```

- Convert grayscale image to indexed image using multilevel thresholding

  grayslice

- Example 2

```
I = imread('snowflakes.png');
X = grayslice(I,16);
imshow(I)
figure, imshow(X,jet(16))
```

- Global image threshold

  graythresh

- Example 3

```
I = imread('coins.png');
level = graythresh(I);
BW = im2bw(I,level);
imshow(BW)
```

- Convert image to binary image, based on threshold

BW = im2bw(I, level)

- Specify level **in the range [0,1]**

- Example 4

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

- Convert indexed image to grayscale image

  ind2gray

- Example 5

```
load trees
I = ind2gray(X,map);
imshow(X,map)
figure,imshow(I)
```

- Convert indexed image to RGB image

  ind2rgb

- Example 6

  Syntax

  RGB = ind2rgb(X,map)

- Convert RGB image or colormap to grayscale

  rgb2gray

- Example 7

```
I = imread('board.tif');
J = rgb2gray(I);
figure, imshow(I), figure, imshow(J);
Convert the colormap to a grayscale colormap.
[X,map] = imread('trees.tif');
gmap = rgb2gray(map);
figure, imshow(X,map), figure, imshow(X,gmap);
```

Practice2.m

```
clear all, close all, clc;
rgbImage=imread('peppers.png');
figure,imshow(rgbImage);colorbar
title('Original Image');
figure,imshow(rgbImage);
bwImage=im2bw(rgbImage);
imshow(bwImage);
title('Black and White Image');
grayImage=rgb2gray(rgbImage);
figure,imshow(grayImage);
title('Grayscale Image');
indexImage=gray2ind(grayImage);
figure,imshow(indexImage);
title('Index Image');
doubleImage=im2double(rgbImage);
figure,imshow(doubleImage);
title('Double Image');
igrayImage=ind2gray(indexImage,cool);
figure,imshow(igrayImage);
title('Index Gray Image');
rgbindexImage=rgb2ind(rgbImage,cool);
figure,imshow(rgbindexImage);
title('RGB Indexex Image');
```

Data types in MATLAB

- Double (64-bit double-precision floating point)
- Single (32-bit single-precision floating point)
- Int32 (32-bit signed integer)
- Int16 (16-bit signed integer)
- Int8 (8-bit signed integer)
- Uint32 (32-bit unsigned integer)
- Uint16 (16-bit unsigned integer)
- Uint8 (8-bit unsigned integer)

Image Types and Type Conversions	
dither	Convert image using dithering
gray2ind	Convert intensity image to indexed image
grayslice	Create indexed image from intensity image by thresholding
im2bw	Convert image to binary image by thresholding
im2double	Convert image array to double precision
im2uint8	Convert image array to 8-bit unsigned integers
im2uint16	Convert image array to 16-bit unsigned integers
ind2gray	Convert indexed image to intensity image
ind2rgb	Convert indexed image to RGB image
isbw	Return true for binary image
isgray	Return true for intensity image
isind	Return true for indexed image
isrgb	Return true for RGB image
mat2gray	Convert matrix to intensity image
rgb2gray	Convert RGB image or colormap to grayscale
rgb2ind	Convert RGB image to indexed image

Color Space Conversions	
hsv2rgb	Convert HSV values to RGB color space (MATLAB)
ntsc2rgb	Convert NTSC values to RGB color space
rgb2hsv	Convert RGB values to HSV color space (MATLAB)
rgb2ntsc	Convert RGB values to NTSC color space
rgb2ycbcr	Convert RGB values to YCbCr color space
ycbcr2rgb	Convert YCbCr values to RGB color space

Dr. Moe Moe Myint
Information Technology Department
Technological University (Kyaukse)

Spatial Transformations(Lab 3)

- Modify the spatial relationship between pixels in an image, mapping pixel locations in an input image to new locations in an output image
-Perform certain specialized spatial transformations, such as resizing and rotating an image

Resize Image

- **B = imresize(A, scale)** returns image B that is scale times the size of A.
- The input image A can be a grayscale, RGB, or binary image.
- If scale is between 0 and 1.0, B is smaller than A. If scale is greater than 1.0, B is larger than A.

Step 1: Resizing an Image

- To resize an image, use the imresize function

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

- To create an output image with 100 rows and 150 columns

```
I = imread('circuit.tif');
J = imresize(I,[100 150]);
imshow(I)
figure, imshow(J)
```

Rotate Image

- B = imrotate(A, angle) rotates image A by angle degrees in a counterclockwise direction around its center point.
- To rotate the image clockwise, specify a negative value for angle.

Step 2: Rotating an Image

- To rotate an image, use the imrotate function

```
I = imread('circuit.tif');
J = imrotate(I,35);
%counterclockwise direction around its center point
imshow(I)
figure, imshow(J)
```

- To rotate an image, use the imrotate function

```
I = imread('circuit.tif');
J = imrotate(I,-35);
%clockwise direction around its center point
imshow(I)
figure, imshow(J)
```

Step 3: Cropping an Image

- To extract a rectangular portion of an image, use the imcrop function

```
I = imread('circuit.tif')
J = imcrop(I);
```

```
I = imread('circuit.tif');
J = imcrop(I,[60 40 100 90]);
```

Image pyramid

- B = impyramid(A, direction) computes a Gaussian pyramid reduction or expansion of A by one level direction can be 'reduce' or 'expand'.

Step 4: Image pyramid reduction and expansion

Reduction

```
I0 = imread('cameraman.tif');
I1 = impyramid(I0, 'reduce');
I2 = impyramid(I1, 'reduce');
I3 = impyramid(I2, 'reduce');
imshow(I0)
figure, imshow(I1)
figure, imshow(I2)
figure, imshow(I3)
```

Expansion

```
I0 =imread('cameraman.tif');
I1 = impyramid(I0, 'expand');
I2 = impyramid(I1, 'expand');
I3 = impyramid(I2, 'expand');
imshow(I0)
figure, imshow(I1)
figure, imshow(I2)
figure, imshow(I3)
```

Exercise

- **Read** the image and display it with title 'Input Image'
- **Resize** the input image with scale greater than 1.0 and display it with title 'Resize Image with **>1.0**'
- Also .....with scale between 0 and 1.0 and display it with title 'Resize Image with **<1.0**'
- **Rotate** the 'Resize image with >1.0' %**Clockwise direction**
- **Crop** the input image with copying the position
- **Compute** a **four-level multiresolution pyramid** of the input image

Practice3.m

```
%Shrink by factor of two using the defaults of bicubic interpolation and
antialiasing
I = imread('rice.png');
J = imresize(I, 0.5);
figure, imshow(I), figure, imshow(J)
%Shrink by factor of two using nearest-neighbor interpolation (This is the
fastest method, but it has the lowest quality.)
J2 = imresize(I, 0.5, 'nearest');  %Resize an indexed image
[X, map] = imread('trees.tif');
[Y, newmap] = imresize(X, map, 0.5);
imshow(Y, newmap)
%Resize an RGB image to have 64 rows (The number of columns is
computed automatically.)
RGB = imread('peppers.png');
RGB2 = imresize(RGB, [64 NaN]);
```

Practice4.m

```
%Read a solar spectra image, stored in FITS format, and rotate the image to
bring it into horizontal alignment. A rotation of -1 degree is all that is
required.
I = fitsread('solarspectra.fts');
I = mat2gray(I);
J = imrotate(I,-1,'bilinear','crop');
figure, imshow(I)
figure, imshow(J)
```

Dr. Moe Moe Myint
Information Technology Department
Technological University (Kyaukse)

Image Analysis (Lab 4)

Image Analysis

bwboundaries	Trace region boundaries in binary image
bwtraceboundary	Trace object in binary image
cornermetric	Create corner metric matrix from image
edge	Find edges in grayscale image
hough	Hough transform
houghlines	Extract line segments based on Hough transform
houghpeaks	Identify peaks in Hough transform
Qtdecomp	Quadtree decomposition
qtgetblk	Block values in quadtree decomposition
qtsetblk	Set block values in quadtree decomposition

Edge Detection

- The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

```
BW = edge(I,'sobel')
```

- The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

```
BW = edge(I,'prewitt')
```

- The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

```
BW = edge(I,'roberts')
```

- The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.

```
BW = edge(I,'log')
```

- The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

```
BW = edge(I,'canny')
```

Examples

- Find the edges of an image using the Prewitt and Canny methods.

```
I = imread('circuit.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
imshow(BW1);
figure, imshow(BW2
```

Practice1.m

```
clc,clear all, close all
I=imread('peppers.png');
imshow(I);title('Input Image');
I=rgb2gray(I);
figure,imshow(I);title('Gray Image');
BW1=edge(I);
figure,imshow(BW1);title('Edge Image');
BW2=edge(I,'sobel');
figure,imshow(BW2);title('Sobel');
BW3=edge(I,'prewitt');
figure,imshow(BW3);title('Prewitt');
BW4=edge(I,'roberts');
figure,imshow(BW4);title('Roberts');
BW5=edge(I,'log');
figure,imshow(BW5);title('Log');
BW7=edge(I,'canny');
figure,imshow(BW7);title('Canny');
```

Questions ???