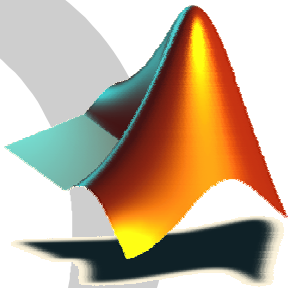
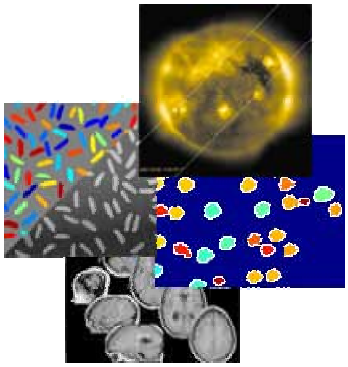
TECHSOURCE
www.techsource.com.my

# Image Processing Using MATLAB®





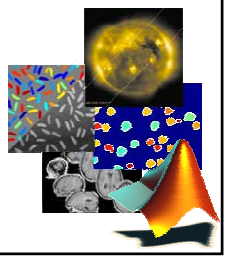
TechSource Systems Sdn. Bhd.



TECHSOURCE
www.techsource.com.my

## Course Outline:

1. **Working with Images in MATLAB**
   a) Image types and classes
   b) Read/write images
   c) Display images
2. **Basic Image Processing**
   a) Image contrast and brightness enhancement
   b) Image arithmetic
3. **Block Processing of Images**
4. **Image Restoration**
   a) Noise reduction (filtering)
   b) Image alignment
5. **Image Segmentation & Edge Detection**
6. **Case Studies**

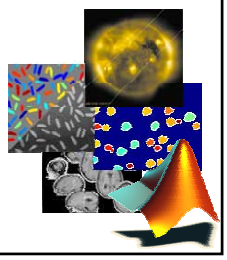


## Section Outline:

1. Image types
   - Index images
   - Intensity images
   - Binary images
   - RGB images

2. Importing and exporting images in MATLAB
   - `imfinfo`
   - `imread` and `imwrite`
   - `imshow`

3. Converting between image formats







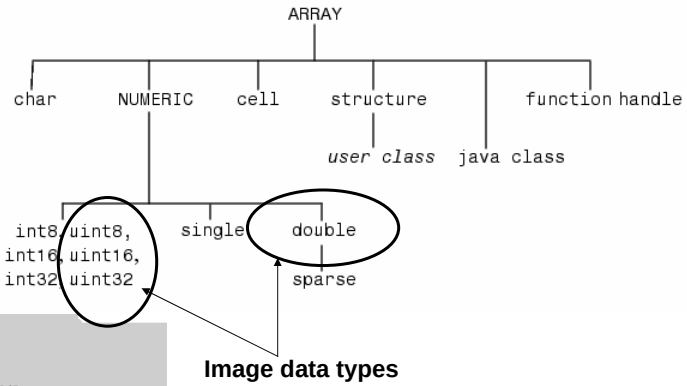
## Image Types

- Four basic types of images are supported in MATLAB
  - Index images: m-by-3 colormap matrix
  - Intensity images: [0 1] or `uint8`
  - Binary images: {0, 1}
  - RGB images: m-by-n-by-3 matrix

```
>> load sampleImages
```

## Image Types: MATLAB Data Types Used

- A wide array of different data types exist in MATLAB, but only a subset of the data types are used to represent images in MATLAB.

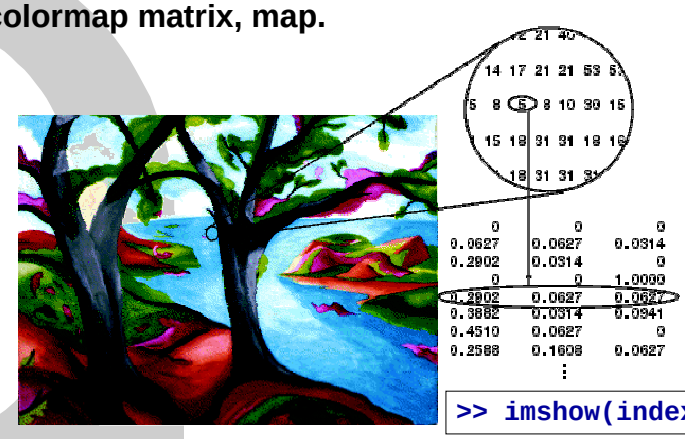







## Image Types: Index Images

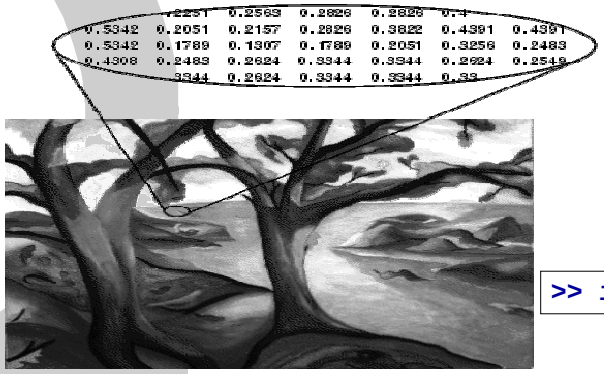
- An indexed image consists of a data matrix, X, and a colormap matrix, map.




```
>> imshow(indexImg, map)
```

## Image Types: Intensity Images

- An intensity image only consists of one matrix, I, whose values represent intensities within some range, for example [0 1] or `uint8`.



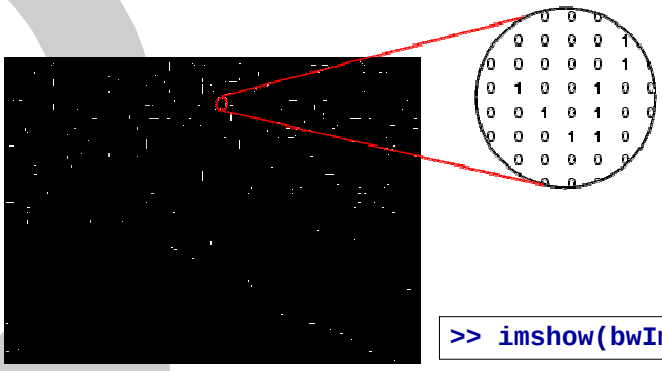
```
>> imshow(intenImg)
```





## Image Types: Binary Images

- In a binary image, each pixel assumes one of only two discrete values: 0 (off) and 1 (on).



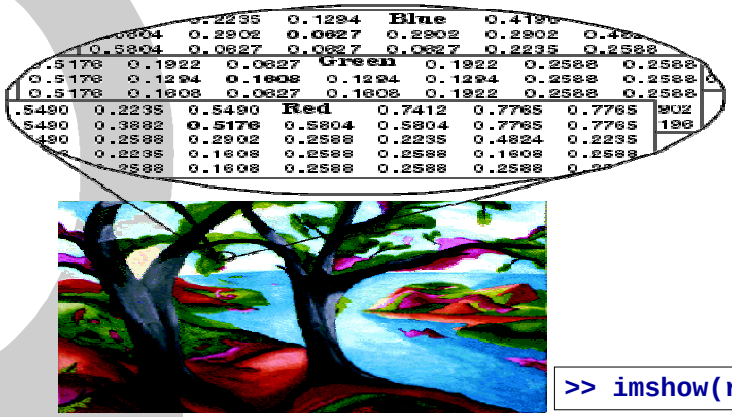
```
>> imshow(bwImg)
```

**Working with Images in MATLAB**

TECHSOURCE www.techsource.com.my

## Image Types: RGB Images

- **RGB image is stored in MATLAB as an m-by-n-by-3 data where each m-by-n page defines red (R), green (G) and blue (B) color components for each pixel.**




```
>> imshow(rgbImg)
```



**Working with Images in MATLAB**

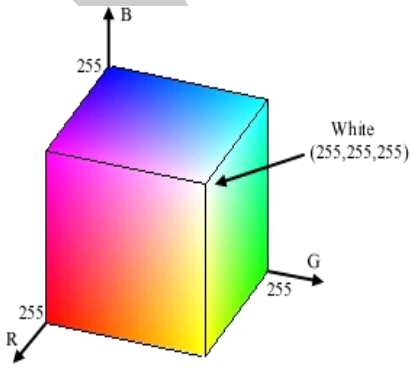
TECHSOURCE www.techsource.com.my

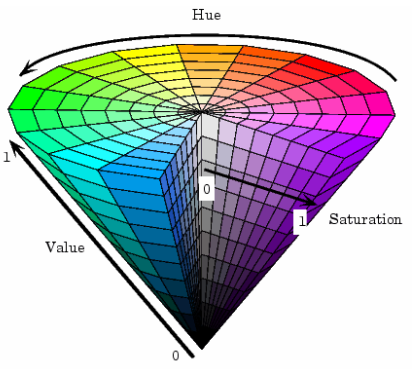
## Overview: How Images are Represented?

| Image Type | Indexed | Intensity | Binary | RGB |
|---|---|---|---|---|
| Double Data | Image is an M-by-N array of integers in the range [1,P].<br><br>Colormap is a P-by-3 array of floating-point values in the range [0,1]. | Image is an M-by-N array of floating-point values. The conventional dynamic range is [0,1]. | Image is an M-by-N logical array containing only 0's and 1's. | Image is an M-by-N-by-3 array of floating-point values in the range [0,1]. |
| Uint8 Data | Image is an M-by-N array of integers in the range [0,P-1].<br><br>Colormap is a P-by-3 array of floating-point values in the range [0,1]. | Image is an M-by-N array of unsigned 8-bit integers. The conventional dynamic range is [0,255]. | Image is an M-by-N logical array containing only 0's and 1's. Unlike uint8 intensity images, 1 represents white | Image is an M-by-N-by-3 array of floating-point values in the range [0,255]. |

## Color Space of RGB and HSV

- There are two main types of color spaces that are used with images: RGB and Hue-Saturation- Value (HSV).







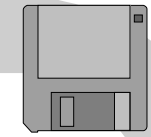

## Importing and Exporting Images in MATLAB

- Image Processing Toolbox
  - `imfinfo`- Returns info about graphics file.
  - `imread` - Read image from graphics file.
  - `imwrite`- Write image to graphics file.
- MATLAB
  - `uiimport` - Starts the Import Wizard.

```
>> imfinfo('canoe.tif')
>> [X, map] = imread('canoe.tif');
>> imshow(X, map);
>> imwrite(X, map, 'canoe2.bmp');
```

**Working with Images in MATLAB** TECHSOURCE www.techsource.com.my

## Graphical Representation of Importing an Image



file.fmt

`imfinfo('file.fmt')`

Colormap?

`[x,map]=imread('file.fmt');`

`x=imread('file.fmt');`

| Image Type | Indexed |
|---|---|
| Double Data | [MXN] + |
| Uint8 Data | [MXN] + |

| Image Type | Intensity | Binary | RGB |
|---|---|---|---|
| Double Data | 1 0.5 0.2<br>0 0.1 0.9<br>0 0.7 0.8 | 1 0 1<br>0 1 0<br>0 1 1 | 0.4 0.3<br>0.1 0.7 0.3<br>1 0.5<br>0.9 0.2 |
| Uint8 Data | 1 45 220<br>100 78 110<br>200 7 98 | 1 0 1<br>0 1 0<br>0 1 1 | 90 140 30<br>100 70 230<br>10 50<br>109 220 |



**Working with Images in MATLAB** TECHSOURCE www.techsource.com.my

## Displaying Images

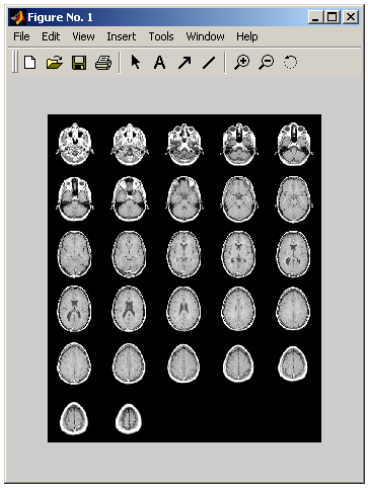
- `imshow` - Display image.
- `image` - Create and display image object (MATLAB).
- `imagesc` - Scale data and display as image (MATLAB).

- `colorbar` - Display colorbar (MATLAB).
- `colormap` - Sets the color map of the image (MATLAB).
- `montage` - Display multiple image frames.
- `truesize` - Adjust display size of image.
- `warp` - Display image as texture-mapped surface.

## Montage

- An example of displaying multiple image frames as a rectangular montage.

```
>> load mri
>> montage(D, map)
```



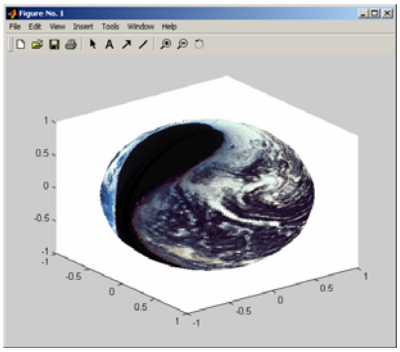





## Warping

- The warp function allows you to display an image as a texture-mapped surface.

```
>> [x, y, z] = sphere;
>> load earth
>> warp(x, y, z, X, map)
```



## Converting Image Formats

- `ind2gray` – indexed image to intensity image.
- `ind2rgb` – indexed image to RGB image (MATLAB).
- `gray2ind` – intensity image to indexed image.
- `rgb2gray` – RGB image or colormap to grayscale.
- `rgb2ind` – RGB image to indexed image.

- `mat2gray` – matrix to intensity image.
- `im2bw` – image to binary image by thresholding.
- `im2double` – image array to double precision.
- `im2uint8` – image array to 8-bit unsigned integers.
- `im2uint16` – image array to 16-bit unsigned integers.





## Exercise 1: Loading and Viewing an Image

1. Load in the `trees.tif` file into MATLAB.
   - What type of image is it? Can you tell before loading in the file?
2. Display the loaded image.
3. Convert the image to an intensity image (grayscale).
4. Now convert it to a binary (black and white) image.

Extra credit:

1. Use subplots to display all three images.
2. Did you find the "Easter egg" in the "trees"?

## Solution: Loading and Viewing an Image

```
>> im_info = imfinfo('trees.tif');
>> im_info(1).ColorType

>> [I,map] = imread('trees.tif');
>> subplot(2,2,1), subimage(I,map)

>> I_gray = ind2gray(I,map);
>> subplot(2,2,2), subimage(I_gray)

>> I_bw = im2bw(I,map,0.4);
>> subplot(2,2,3), subimage(I_bw)

% Easter egg
>> figure
>> [I2,map] = imread('trees.tif',2);
>> imshow(I2,map)
```

Why use subimage?

Threshold



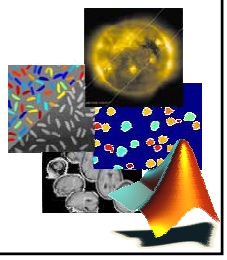


## Section Summary:

1. Image types
   - Index images
   - Intensity images
   - Binary images
   - RGB images

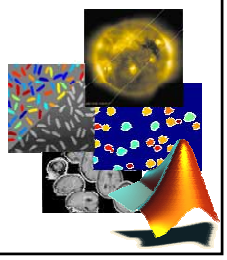
2. Importing and exporting images in MATLAB
   - `imfinfo`
   - `imread` and `imwrite`
   - `imshow`

3. Converting between image formats



## Section Outline:

1. Image enhancement
   - Image histogram
   - Image contrast adjustment
   - Image brightness adjustment
2. Image thresholding
3. Image arithmetic







## Image Enhancement

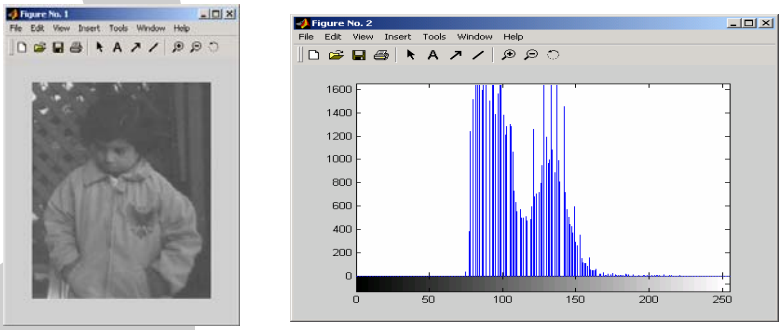
- One of the most basic ways to enhance an image is to change its *brightness* and its *contrast*, and this can be done is by working with the image's histogram.
  - By stretching the color distribution
  - By equalizing the distribution of colors to use the full range
  - By adjusting the scaling of the colors
- If you are separating an object from its background, *thresholding* is a technique that could be used as well.

**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Histogram

- **A histogram of an image shows the current level of contrast (the distribution of gray levels).**

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



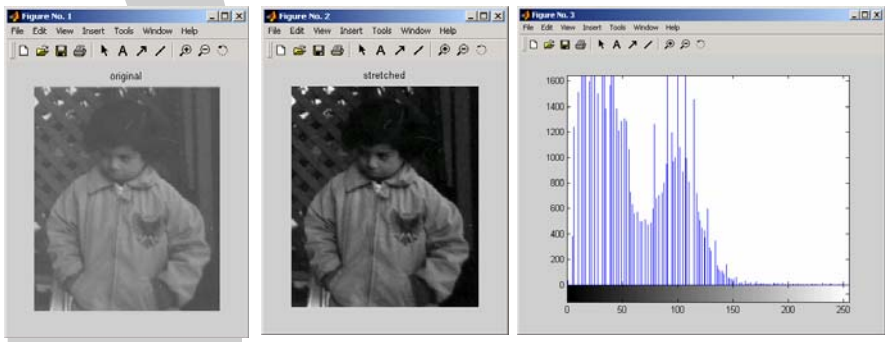



**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Histogram Stretching

- **One way to increase the contrast of an image is to stretch the pixel values (min == 0 and max == 255).**

$$J = 255 \cdot \frac{I - I_{\min}}{I_{\max} - I_{\min}}$$

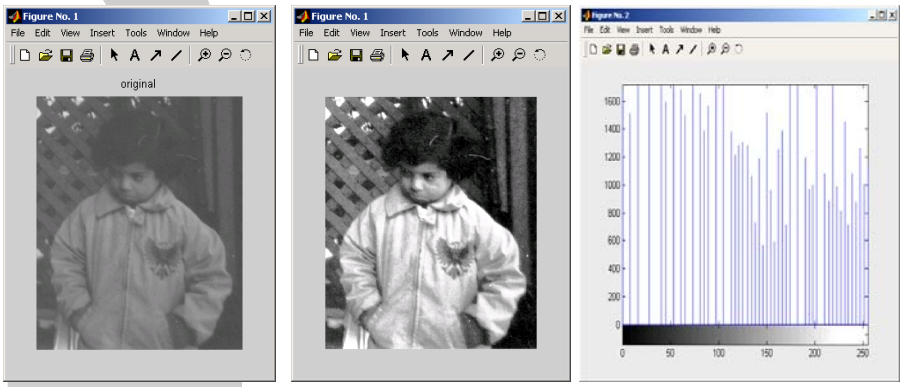


**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Histogram Equalization

- The `histeq` function can be used to equally distribute the histogram and enhance the contrast.

```
>> J = histeq(I);
```






**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Histogram Adjustment

- Intensity adjustment is a technique for mapping an image's intensity values to a new range (`imadjust`).

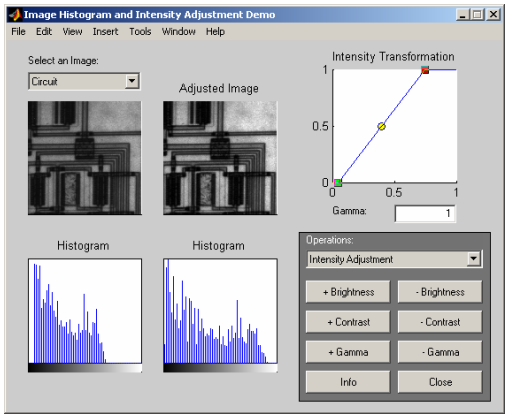
```
>> I = imread('cameraman.tif');
>> J = imadjust(I,[0 0.2],[0.5 1]);
>> imshow(I)
>> figure, imshow(J)
```

**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Understanding Intensity Adjustment

- The following example demonstrates how an image's intensity can be changed to enhance different characteristics of an image.

```
>> imadjdemo
```

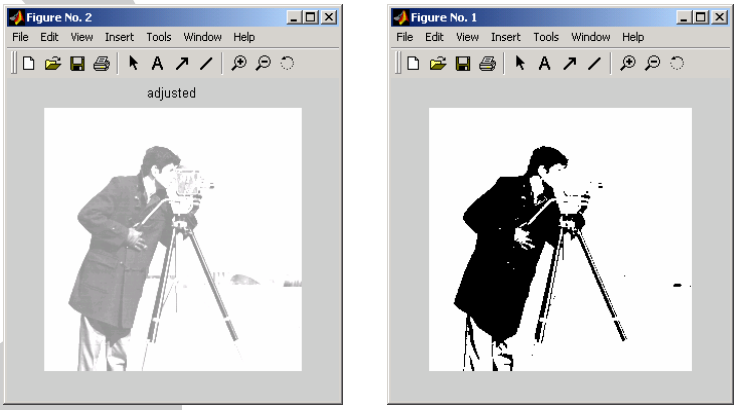





**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Image Thresholding

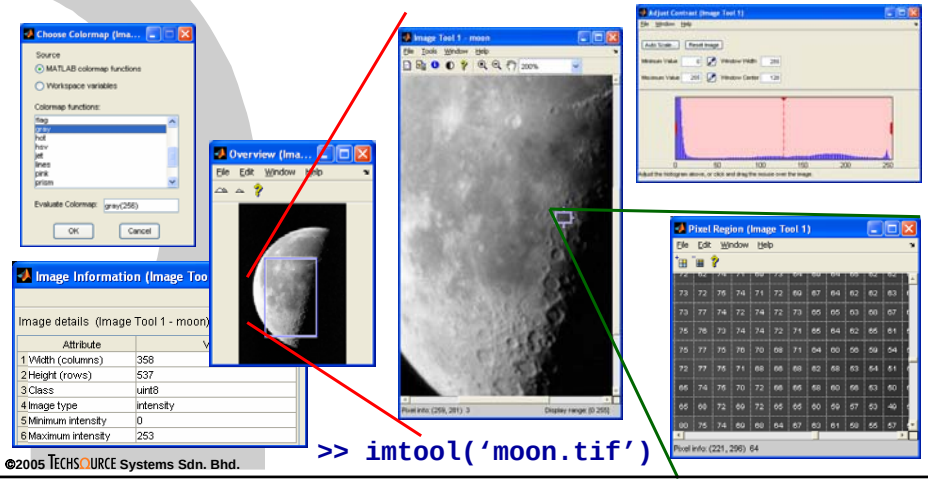
- The adjusted cameraman image can be thresholded to create a black and white image of the cameraman and the camera.



**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Using `imtool` GUI for Image Analysis

- The **`imtool`** is an image display GUI that provides access to Pixel Region tool, the Image Information tool, and the Adjust Contrast tool.




```
>> imtool('moon.tif')
```



**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Image Arithmetic

- **With just simple addition, subtraction, multiplication and division a number of different image processing techniques can be implemented.**
  - With addition and multiplication an image contrast can be increased that facilitates edge detection process.
  - With subtraction and division changes can be detected from one image to another.

| | |
|---|---|
| **`imabsdiff`** | - Compute absolute difference of two images |
| **`imadd`** | - Add two images or add constant to image |
| **`imcomplement`** | - Complement image |
| **`imdivide`** | - Divide two images or divide image by a constant |
| **`imlincomb`** | - Compute linear combination of images |
| **`immultiply`** | - Multiply two images or multiply image by constant |
| **`imsubtract`** | - Subtract two images or subtract constant from image |

Basic Image Processing

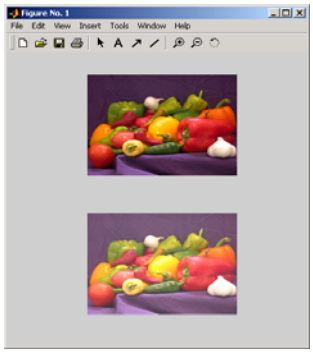
TECHSOURCE www.techsource.com.my

## Image Addition

- Image addition makes it possible to superimpose an image on top of another or control the brightness of an image.
- Each resulting pixel is the sum of the respective pixels of the two images, of the same size and of the same class.

```
>> I1 = imread('peppers.png');
>> I2 = imadd(I1, 50);
>> subplot(2,1,1), imshow(I1)
>> subplot(2,1,2), imshow(I2)
```

```
>> % MATLAB 7 New Features
>> I1 = imread('peppers.png');
>> I2 = I1 + 50 % direct addition
>> subplot(2,1,1), imshow(I1)
>> subplot(2,1,2), imshow(I2)
```





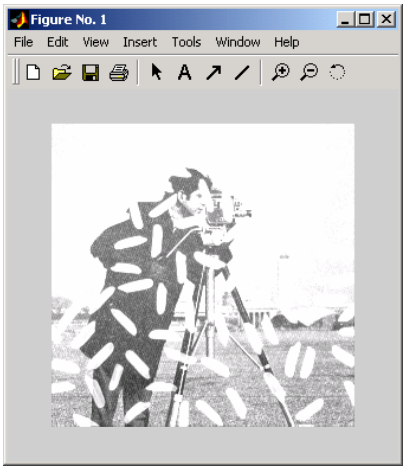
Basic Image Processing

TECHSOURCE www.techsource.com.my

## Image Addition (Continued)

```
>> I = imread('rice.png');
>> J = imread('cameraman.tif');
>> K = imadd(I, J);
>> imshow(K)
```

```
>> % MATLAB 7 New Features
>> I = imread('rice.png');
>> J = imread('cameraman.tif');
>> K = I + J; % direct addition
>> imshow(K);
```

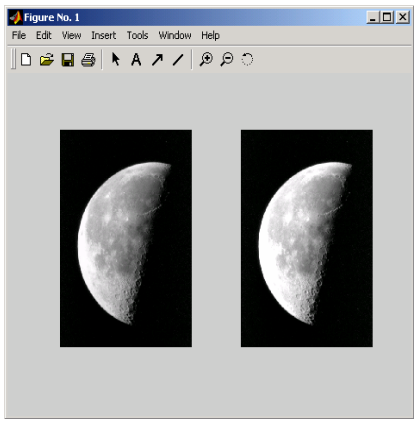


**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Image Multiplication

```
>> I = imread('moon.tif');
>> J = immultiply(I, 1.2);
>> subplot(1,2,1),imshow(I)
>> subplot(1,2,2),imshow(J)
```

```
>> % MATLAB 7 New Features
>> I = imread('moon.tif');
>> J = 1.2 * I; % direct multiply
>> subplot(1,2,1),imshow(I)
>> subplot(1,2,2),imshow(J)
```

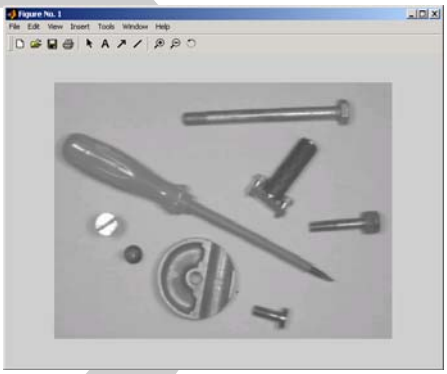


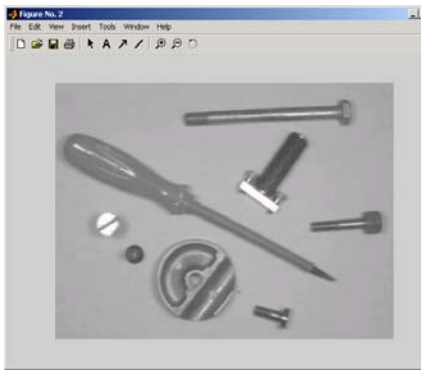



**Basic Image Processing** TECHSOURCE www.techsource.com.my

## Image Subtraction

- Can you see anything different about the two images?

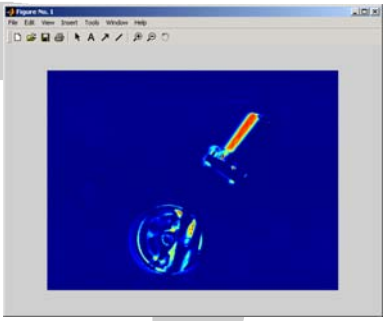


## Basic Image Processing

TECHSOURCE www.techsource.com.my

- **Using subtraction, you can identify the following differences.**



**Did the image match up with what you were expecting?**

```
>> [im1, map1] = imread('change1.gif');
>> [im2, map2] = imread('change2.gif');
>> im_diff = imsubtract(im1, im2);
>> % im_diff = im1 - im2; % direct subtraction
>> imshow(im_diff)
>> colormap(jet)
>> set(gca, 'clim', [0 60]); % adjust colorbar
```

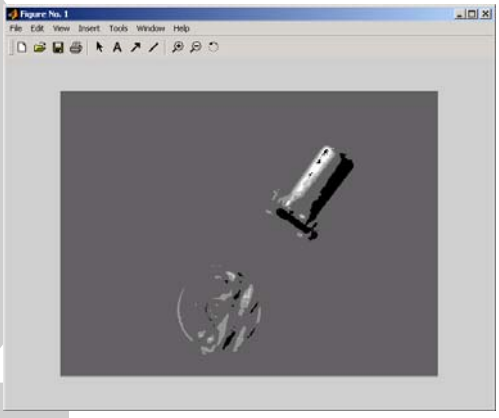


## Basic Image Processing

TECHSOURCE www.techsource.com.my

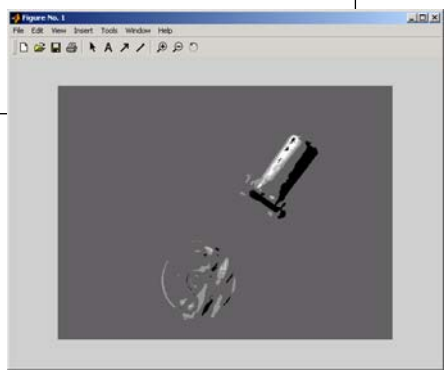
### Exercise 2: Image Division

- **Repeat what you just did with image subtraction, except, this time, use division instead.**



## Solution: Loading and Viewing an Image

```
>> [im1, map1] = imread('change1.gif');
>> [im2, map2] = imread('change2.gif');
>> im_diff = imdivide(im1, im2);
>> % im_diff = im1./im2; % direct element-wise division
>> imshow(im_diff)
>> colormap(map1)
>> min(im_diff(:)); % ans = 0
>> max(im_diff(:)); % ans = 5
>> set(gca, 'clim', [0 5])
```

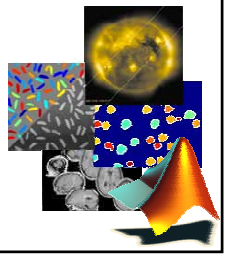






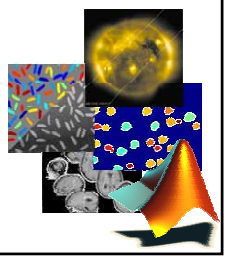
## Section Summary:

1. Image enhancement
   - Image histogram
   - Image contrast adjustment
   - Image brightness adjustment
2. Image thresholding
3. Image arithmetic



## Section Outline:

1. What is block processing?
2. Distinct block operations
3. Sliding neighbourhood operations
4. Example of block processing
   - Convolution
   - Correlation
5. Column processing







## What is block processing?

An operation in which an image is processed in blocks rather than all at once.

- The blocks have the same size across the image.
- An operation is applied to one block at a time.
- Once processed, the blocks are re-assembled to form an output image.

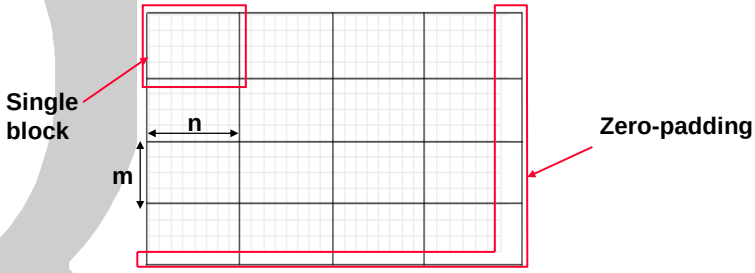
**Block Processing of Images**

TECHSOURCE www.techsource.com.my

## Distinct Block Operations

**Distinct blocks are rectangular partitions that divide an image matrix into m-by-n sections.**

- **Blocks are overlaid by starting at the upper-left corner.**
- **Zeros are padded onto blocks that exceed the size of the image.**




```
B = blkproc(A,[m n],fun)
```



**Block Processing of Images**

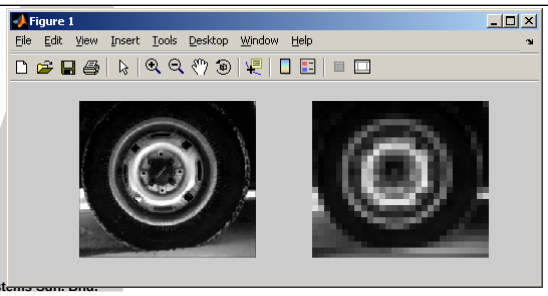
TECHSOURCE www.techsource.com.my

## Block Processing – Averaging Filter

**Find the average value of each 8-by-8 block and replace all the pixels in the block with the average value.**

```
>> I1 = imread('tire.tif');
>> f = @(x) uint8(round(mean2(x)*ones(size(x))))
>> I2 = blkproc(I1,[8 8], f);
>> subplot(1,2,1), imshow(I1)
>> subplot(1,2,2), imshow(I2)
```

Function handle

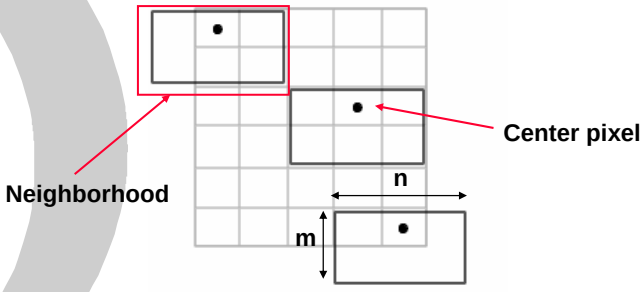


Block Processing of Images

TECHSOURCE www.techsource.com.my

## Sliding Neighborhood Operations

**Each center pixel value is determined by applying some algorithm to its neighboring pixels of a defined size – neighborhood**




```
B = nlfilter(A,[m n],fun)
```

**Note:**
**For an m-by-n neighborhood, the center pixel is `floor(([m n]+1)/2)`**



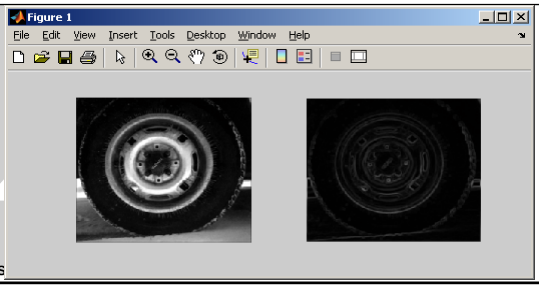
Block Processing of Images

TECHSOURCE www.techsource.com.my

## Sliding Neighborhood Operations – Nonlinear Filter

**Replace each pixel with the standard deviation of the values of the input pixel's 3-by-3 neighborhood.**

```
>> I1 = imread('tire.tif');
>> f = @(x) uint8(round(std2(x))); % function handle
>> I2 = nlfilter(I1,[3 3], f);
>> subplot(1,2,1), imshow(I1)
>> subplot(1,2,2), imshow(I2)
```



**Block Processing of Images**

TECHSOURCE www.techsource.com.my

## Example of Block Processing - Convolution

In *convolution*, the value of an output pixel is computed as a weighted sum of neighboring pixels. The matrix of weights is called the *convolution kernel* (or *filter*).

Steps for convolving an image

- Rotate the convolution kernel 180 degrees about the center.
- Slide the rotated convolution kernel over the image.
- Multiply each weight in the rotated convolution kernel by the pixel of the image
- Sum up all individual products.



**Block Processing of Images**

TECHSOURCE www.techsource.com.my

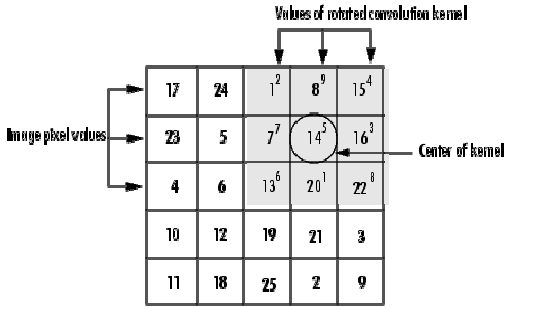
Given an image, A, and the convolution kernel, h, the procedure for calculating the convolution for the value of 14 would be as follows.

- Notice that in the calculation the convolution kernel, h, is rotated a 180 degrees.

```
>> A = [17 24 1  8  15
        23 5  7  14 16
        4  6  13 20 22
        10 12 19 21 3
        11 18 25 2  9]
>> h = [8 1 6
        3 5 7
        4 9 2]

>> conv2(A,h, 'same')
```

$1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575$



**Block Processing of Images**

TECHSOURCE www.techsource.com.my

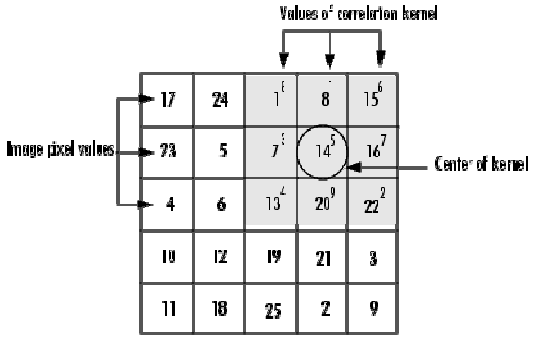
## Example of Block Processing - Correlation

In *correlation*, the value of an output pixel is also computed as a weighted sum of neighboring pixels. The difference is that the matrix of weights (*correlation kernel*) is *not* rotated.

```
>> A = [17 24 1  8  15
        23 5  7  14 16
        4  6  13 20 22
        10 12 19 21 3
        11 18 25 2  9]
>> h = [8 1 6
        3 5 7
        4 9 2]

>> filter2(h,A)
```

$$1\cdot 8+8\cdot 1+15\cdot 6+7\cdot 3+14\cdot 5+16\cdot 7+13\cdot 4+20\cdot 9+22\cdot 2=585$$






**Block Processing of Images**

TECHSOURCE www.techsource.com.my

## Column Processing

Arranging each sliding neighborhood or distinct block as separate columns, and process each column as a block.

- Faster, since most of MATLAB is column-based.
- But uses more memory.

Syntax

```
B = colfilt(A,[m n],block_type,fun)
```

`A` – image matrix

`[m n]` – size of block

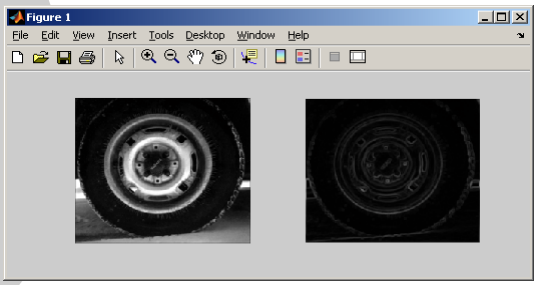
`block_type` – Either `'distinct'` or `'sliding'`

`fun` – function to apply

## Block Processing of Images

TECHSOURCE www.techsource.com.my

```
>> I1 = imread('tire.tif');
>> f = @(x) uint8(round(std(double(x))));
>> I2 = colfilt(I1,[3 3],'sliding',f);
>> subplot(1,2,1), imshow(I1)
>> subplot(1,2,2), imshow(I2)
```



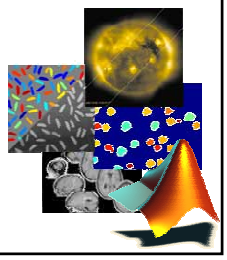



## Block Processing of Images

TECHSOURCE www.techsource.com.my

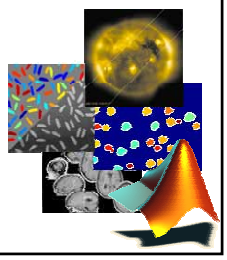
### Section Summary:

1. What is block processing?
2. Distinct block operations
3. Sliding neighbourhood operations
4. Example of block processing
   - Convolution
   - Correlation
5. Column processing



## Section Outline:

1. Reducing noise
   - Filters
   - Region-based processing
2. Image alignment
   - Rotation
   - Cropping
   - Resizing







## Reducing Noise

Where does noise come from?
- Scanner resolution
- Film grain (granularity)
- Hardware (interference patterns)
- Other

Common types of noise
- Whitenoise (Gaussian)
- Local variance (Gaussian with intensity-dependent variance)
- Salt and pepper
- Speckle

## Image Restoration

TECHSOURCE
www.techsource.com.my

A filter can be used to reduce the effect of noise in an image. The Image Processing Toolbox provides three main methods of filtering:

- Linear filtering
- Median filtering
- Adaptive filtering

Different methods are better for different kinds of noise.

In the following section we will investigate the types of methods, and what type of noise they are most effective in reducing.



## Image Restoration

TECHSOURCE
www.techsource.com.my

### How Do I Model Noise?

The function `imnoise` allows different types of noise to be modeled.

Syntax:

```
J = imnoise(I,type,parameters)
```

`I` – Image

`type` – `gaussian`, `localvar`, `poisson`, `salt & pepper`, `speckle`

`parameters` – additional parameters needed given the type of noise

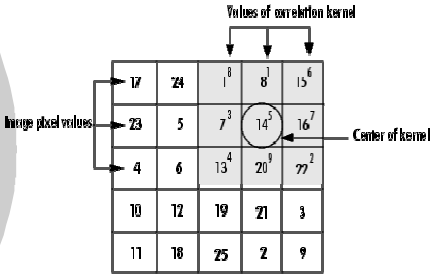
Image Restoration

TECHSOURCE
www.techsource.com.my

## Linear Filtering

A linear filter computes each output pixel value according to a linear combination of the input pixel's neighborhood.

- The basics of linear filtering are done through *correlation* and *convolution*.




- In the Image Processing Toolbox both these operations are performed using the `imfilter` command.



Image Restoration

TECHSOURCE
www.techsource.com.my

## Filtering using `imfilter`

**Syntax**

```
B = imfilter(A,H)
B = imfilter(A,H,option1,option2,...)
```

**`A` – Input image**

**`H` – The filter, also known as the correlation/convolution kernel**

**`options` – Boundary options, output size option, correlation and convolution option**

**Note: By default `imfilter` performs correlation.**

Image Restoration

TECHSOURCE www.techsource.com.my

## Averaging Filter

**A very basic example of a linear filter is an averaging filter.**

```
>> I = imread('cameraman.tif');
>> % addition of graininess (i.e. noise)
>> I_noise = imnoise(I, 'speckle', 0.01);
>> % the average of 3^2, or 9 values
>> h = ones(3,3) / 3^2;
>> I2 = imfilter(I_noise,h);
>> subplot(1,2,1), imshow(I_noise), title('Original image')
>> subplot(1,2,2), imshow(I2), title('Filtered image')
```

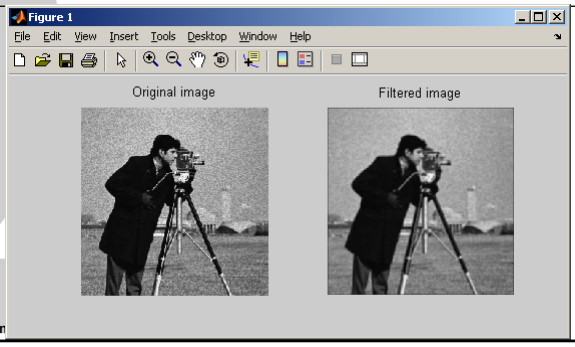





Image Restoration

TECHSOURCE www.techsource.com.my

## Special Linear Filters

**The function `fspecial` creates a variety of two-dimensional filters.**

**Syntax**

```
h = fspecial(type, parameter)
```

**h – two-dimensional correlation kernel**

**`type` – one of the specified special filter types: `gaussian`, `sobel`, `prewitt`, `laplacian`, `log`, `motion`, averaging (`average`), circular averaging (`disk`), and a contrast sharpening (`unsharp`) filter**

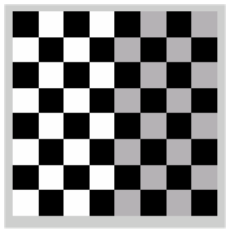
**`parameters` – particular to the type of filter chosen**

Image Restoration

TECHSOURCE
www.techsource.com.my

## Exercise 3: Investigating Linear Filters

- Create a checkerboard image using the `checkerboard` function.



- Filter the checkerboard image with each `fspecial` filter type and display the image. For each filter type you can just use its default parameter(s).
- Extra credit: Display all nine images in one figure.



Image Restoration

TECHSOURCE
www.techsource.com.my

## Solution: Investigating Linear Filters

```
>> I = checkerboard;
>> type = {'gaussian','sobel','prewitt',...
        'laplacian','log','motion',...
        'average','disk','unsharp'}; % cell arrays
>> for index = 1:length(type)
     h = fspecial(type{index});
     I2 = imfilter(I,h);
     subplot(3,3,index)
     imshow(I2)
     title(type{index})
   end
```

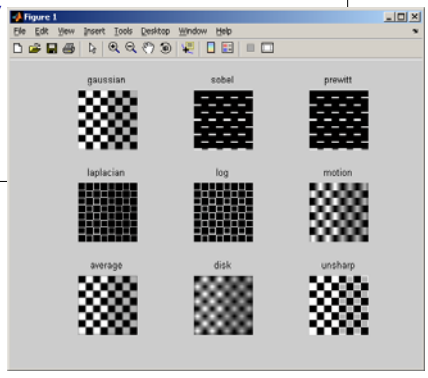


Image Restoration

TECHSOURCE
www.techsource.com.my

## Median Filtering

When noise causes the pixels to vary greatly from the original value (salt and pepper), a median filter is more effective in reducing the noise.

Syntax

```
B = medfilt2(A,[m n])
```

A – Input image

B – Output image

`[m n]` – Neighborhood block size to be used to calculate the median.



Image Restoration

TECHSOURCE
www.techsource.com.my

## Example: Median Filter

```
>> I = imread('eight.tif');
>> I_noise = imnoise(I,'salt & pepper');
>> h = fspecial('average',[3 3]);
>> I_avg = imfilter(I_noise,h);
>> I_med = medfilt2(I_noise, [3 3]);
>> subplot(2,2,1.5)
>> imshow(I_noise), title('Quarters w/ Salt & Pepper')
>> subplot(2,2,3)
>> imshow(I_avg), title('Averaging Filter')
>> subplot(2,2,4)
>> imshow(I_med), title('Median Filter')
```

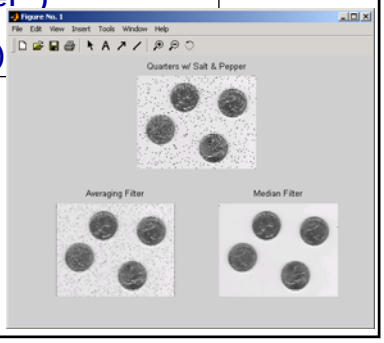


Image Restoration

TECHSOURCE
www.techsource.com.my

## Adaptive Filter

The `wiener2` adaptive filter tailors itself to the local image variance *adaptively*. If the variance is large, it minimally smoothes the image. If the variance is small, it performs more smoothing.

This type of filter is effective in reducing the effects of Gaussian whitenoise.

Syntax

```
[J,noise] = wiener2(I,[m n],noise)
```

`I` – Input image

`[m n]` – Size neighborhood block used to calculate the median.

`J` – Output image

`noise` – Noise variance

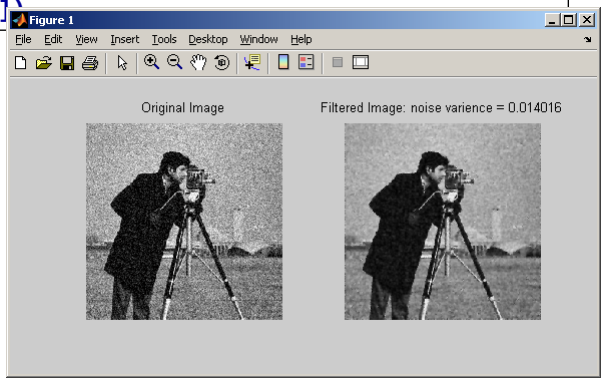


Image Restoration

TECHSOURCE
www.techsource.com.my

## Wiener Filter Example

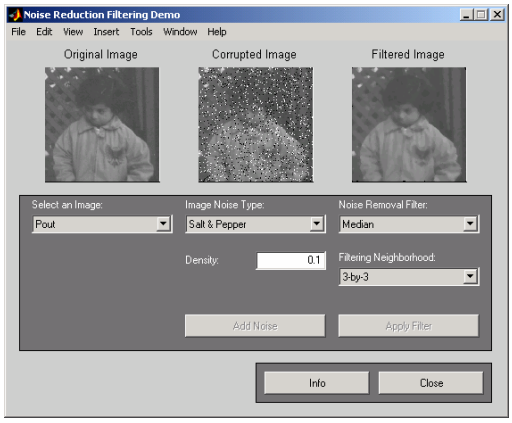
```
>> I = imread('cameraman.tif');
>> I_noise = imnoise(I, 'gaussian', 0.01);
>> [I2, noise] = wiener2(I_noise,[3 3]);
>> subplot(1,2,1), imshow(I_noise), title('Original Image')
>> subplot(1,2,2), imshow(I2)
>> title(['Filtered Image: noise variance =',...
        num2str(noise)])
```



## Filter Demonstration

**Explore noise reduction in images using linear and non-linear filtering techniques by running the following demo:**

```
>> nrfiltdemo
```








## Region-Based Processing

**Region-based processing allows you to select a region of interest (ROI), and process only upon the selected area.**

- **A ROI is defined using a binary mask – The mask contains 1's for all pixels that are part of the region of interest and 0's everywhere else.**
- **Types of region-based processing that can be done:**
  - **Specify a region of interest (`roipoly`, `roicolor`)**
  - **Filter a region (`roifilt2`)**
  - **Fill in a region (`roifill`)**

Image Restoration

TECHSOURCE
www.techsource.com.my

## Specifying a Region of Interest

A region of interest can be specified using one of the Image Processing functions or with any user defined binary mask.

The options are:

- Using `roipoly`, allows you to specify a polygonal region of interest. When called with no inputs a cursor can be used to create a polygon.
- Using `roicolor`, where you can define the region of interest based on a color or intensity range.
- Using Boolean indexing to create a binary mask.



Image Restoration

TECHSOURCE
www.techsource.com.my

## Filtering a Region

Once a region has been specified, a filter can be created and implemented on the region.

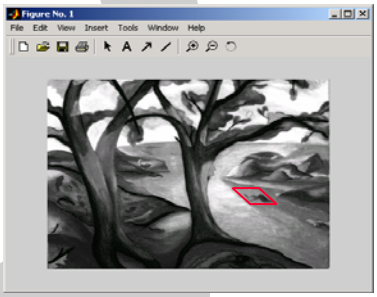
```
>> I = imread('cameraman.tif');
>> imshow(I)
>> BW = roipoly;
>> figure, imshow(BW)
>> h = fspecial('unsharp');
>> I2 = roifilt2(h,I,BW);
>> imshow(I2)
```

Image Restoration

TECHSOURCE
www.techsource.com.my

## Filling in a Region of Interest (ROI)

```
>> [X,map] = imread('trees.tif');
>> I = ind2gray(X,map);
>> imshow(I)
>> I2 = roifill; % intensity img only
>> figure, imshow(I2)
```



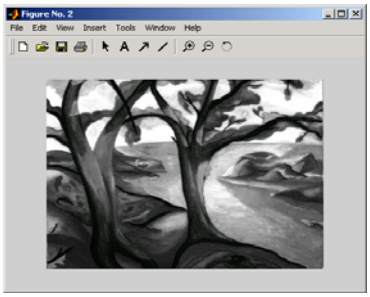






Image Restoration

TECHSOURCE
www.techsource.com.my

## Image Alignment

When the alignment is off in an image, you can apply some basic spatial transformation techniques.

- Rotate the image (`imrotate`)
- Crop the image (`imcrop`)
- Resize the image (`imresize`)

With the `imtransform` function, you can transform your image to any geometric shape as well.

Image Restoration

TECHSOURCE
www.techsource.com.my

## Image Rotation

Syntax

```
B = imrotate(A,angle,method)
B = imrotate(A,angle,method,'crop')
```

`B` – Output image

`A` – Input image

`angle` – Degrees of rotation in the counter-clockwise direction

`method` – Type of interpolation:
`[{nearest}, bilinear, bicubic]`

`'crop'` – Returns only central portion of `B` which is the same size as `A`.



Image Restoration

TECHSOURCE
www.techsource.com.my

## Image Cropping

Syntax

```
I2 = imcrop(I,rect)
```

`I2` – Output image

`I` – Input image

`rect` – Spatial coordinates of
`[xmin ymin width height]`

If `rect` is omitted, you specify the crop region on the image directly using the mouse.

Image Restoration

TECHSOURCE
www.techsource.com.my

## Image Resizing

**Syntax**

```
B = imresize(A,m,method)
```

**B – Output image**

**A – Input image**

**m – Magnification factor**

**method – Type of interpolation:**
**[{nearest}, bilinear, bicubic]**



Image Restoration

TECHSOURCE
www.techsource.com.my

## Exercise 4: Aligning an Object

- Rotate the image so that the letter L is standing right side up.
- Crop the image so that only the letter is showing.
- Resize the image so that it is the same as the original image.

```
>> [X, map] = imread('bw_L.gif');
>> I = ind2gray(X,map);
```

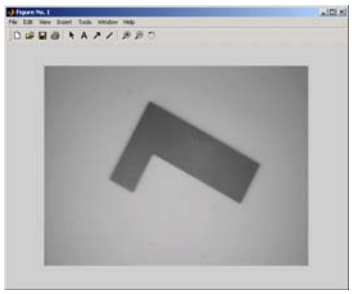


Image Restoration

TECHSOURCE
www.techsource.com.my

## Solution: Aligning an Object

```
>> [X, map] = imread('bw_L.gif');
>> I = ind2gray(X,map);
>> size_I = size(I);
>> subplot(2,2,1), imshow(I)

>> J = imrotate(I,120,'nearest','crop');
>> subplot(2,2,2), imshow(J)

>> K = imcrop(J);
>> subplot(2,2,3), imshow(K)

>> L = imresize(K,size_I);
>> subplot(2,2,4), imshow(L)

>> whos I J K L
```

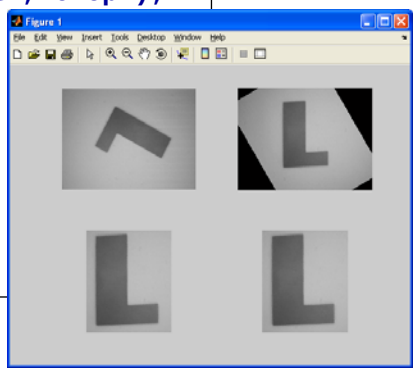


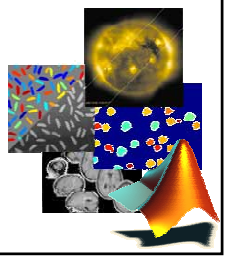



Image Restoration

TECHSOURCE
www.techsource.com.my

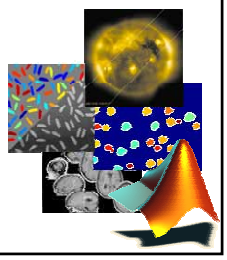
## Section Summary:

1. Reducing noise
   - Filters
   - Region-based processing
2. Image alignment
   - Rotation
   - Cropping
   - Resizing



## Section Outline:

1. Morphology and segmentation
2. Structuring element
3. Dilation and erosion
4. Edge detection







## Example Problem

**We are assigned to locate all the rectangular chips on a black and white integrated circuit image, but how?**

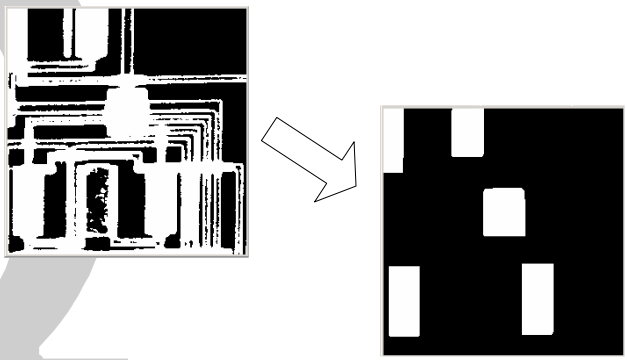


Image Segmentation & Edge Detection

TECHSOURCE www.techsource.com.my

## Morphology & Segmentation

One way we can solve the problem of identifying objects is using morphological techniques to segment the objects.

- Morphology – technique used for processing image based on shapes.
- Segmentation – the process used for identifying objects in an image.



Image Segmentation & Edge Detection

TECHSOURCE www.techsource.com.my

## Structuring Element

To process an image according to a given shape, we need to first define the shape, or *structuring element*.

The Image Processing Toolbox provides a function for creating a structuring element that can be used, `strel`.

Syntax

```
SE = strel(shape,parameters)
```

SE – Structuring element

shape – Flat ['arbitrary' 'pair' 'diamond' 'periodicline' 'disk' 'rectangle' 'line' 'square' 'octagon']
Nonflat ['arbitrary' 'ball']

`parameters` – Associated with the selected shape

**Image Segmentation & Edge Detection**

TECHSOURCE www.techsource.com.my

**Below is an example of a diamond shaped structuring element with the distance from the structuring element origin to the points of the diamond being 3.**

```
>> SE = strel('diamond',3)

SE =
Flat STREL object containing 25 neighbors.
Decomposition: 3 STREL objects containing a total of 13
neighbors

Neighborhood:
     0     0     0     1     0     0     0
     0     0     1     1     1     0     0
     0     1     1     1     1     1     0
     1     1     1     1     1     1     1
     0     1     1     1     1     1     0
     0     0     1     1     1     0     0
     0     0     0     1     0     0     0
```



**Image Segmentation & Edge Detection**

TECHSOURCE www.techsource.com.my

## Dilation & Erosion

**To dilate an image, use the `imdilate` function, and to erode an image, use the `imerode` function.**

**Syntax**

```
IM2 = imdilate(IM,SE)
IM2 = imerode(IM,SE)
```

**`IM2` – The dilated/eroded image**

**`IM` – The image to dilate/erode**

**`SE` – The structuring element**

## Image Segmentation & Edge Detection

TECHSOURCE www.techsource.com.my

**Below is an example of dilating a square binary image.**

```
BW = zeros(9,10);
BW(4:6,4:7) = 1
SE = strel('square',3)
BW2 = imdilate(BW,SE)
```

```
BW =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

```
BW2 =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```



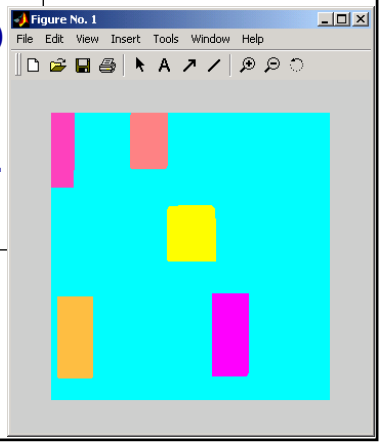
## Image Segmentation & Edge Detection

TECHSOURCE www.techsource.com.my

# Example Solution

**To locate the chips on the integrated circuit we can use the morphological techniques of erosion and dilation.**

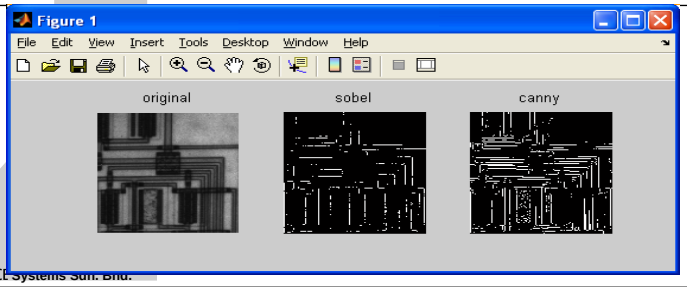
```
>> BW1 = imread('circbw.tif');
>> SE = strel('rectangle',[40 30])
>> BW2 = imerode(BW1,SE);
>> BW3 = imdilate(BW2,SE);
>> L = bwlabel(BW3);
>> RGB = label2rgb(L, 'spring',...
 'c', 'shuffle');
>> imshow(RGB)
```



## Edge Detection

**The edge function of the Image Processing Toolbox makes it easy to perform edge detection.**

```
>> I = imread('circuit.tif');
>> BW1 = edge(I,'sobel');
>> BW2 = edge(I,'canny');
>> subplot(131),imshow(I),title('Original')
>> subplot(132),imshow(BW1),title('sobel')
>> subplot(133),imshow(BW2),title('canny')
>> edgedemo
```

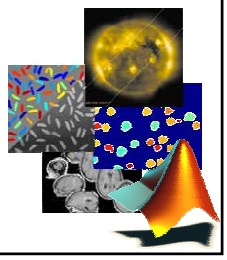







## Section Summary:

1. Morphology and segmentation
2. Structuring element
3. Dilation and erosion
4. Edge detection



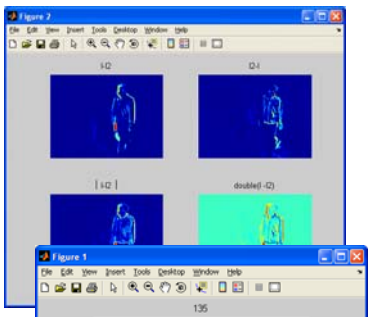
Case Studies

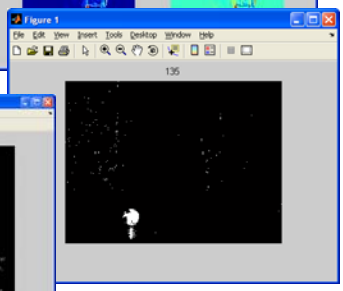
TECHSOURCE
www.techsource.com.my

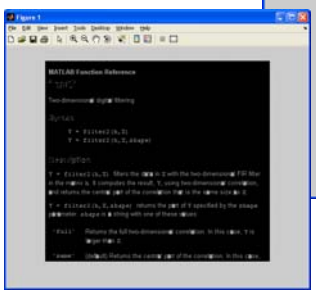
## Section Outline:

1. Motion detection
   `[motion_detect.m]`
2. Object tracking
   `[BounceJasc.m]`
3. Text recognition
   `[recognize_text.m]`

Motion Detection







Text Recognition

Object Tracking



TECHSOURCE
www.techsource.com.my

©2005 TECHSOURCE Systems Sdn. Bhd.