Image Processing Techniques Using MATLAB 2012

By

Dr. R. R. Manza

Reading and Writing Image Data

This chapter describes how to get information about the contents of a graphics file, read image data from a file, and write image data to a file, using standard graphics and medical file formats.

Getting Information About a Graphics File

To obtain information about a graphics file and its contents, use the imfinfo function. You can use imfinfo with any of the formats supported by MATLAB. Use the imformats function to determine which formats are supported.

Cont..

The information returned by imfinfo depends on the file format, but it always includes at least the following:

- Name of the file
- File format
- Version number of the file format
- File modification date
- File size in bytes

Cont..

- Image width in pixels
- Image height in pixels
- Number of bits per pixel
- Image type: truecolor (RGB), grayscale (intensity), or indexed

Reading Image Data

To import an image from any supported graphics image file format, in any of the supported bit depths, use the imread function. This example reads a truecolor image into the MATLAB workspace as the variable RGB.

RGB = imread('football.jpg');

Writing Image Data to a File

To export image data from the MATLAB workspace to a graphics file in one of the supported graphics file formats, use the imwrite function. When using imwrite, you specify the MATLAB variable name and the name of the file. If you include an extension in the filename, imwrite attempts to infer the desired file format from it. For example, the file extension .jpg infers the Joint Photographic Experts Group (JPEG) format. You can also specify the format explicitly as an argument to imwrite.

Cont..

This example loads the indexed image X from a MAT-file, clown.mat, along with the associated colormap map, and then exports the image as a bitmap (BMP) file.

```
load clown
whos
```

Cont..

Your output appears as shown:

Name	Size	Bytes	Class	Attributes
X	200x320	512000	double	
caption	2x1	4	char	
map	81x3	1944	double	

Cont..

Export the image as a bitmap file:

imwrite (X, map, ‘clown.bmp’)

Converting Between Graphics File Formats

To change the graphics format of an image, use imread to import the image into the MATLAB workspace and then use the imwrite function to export the image, specifying the appropriate file format.

To illustrate, this example uses the imread function to read an image in TIFF format into the workspace and write the image data as JPEG format.

Cont..

```
moon_tiff = imread ( 'moon.tif' );

imwrite ( moon_tiff, 'moon.jpg' );
```

Displaying and Exploring Images

The Image Processing Toolbox software includes two display functions, imshow and imtool. Both functions work within the Handle Graphics architecture: they create an image object and display it in an axes object contained in a figure object.

Displaying Images Using the imshow Function

To display image data, use the imshow function. The following example reads an image into the MATLAB workspace and then displays the image in a MATLAB figure window.

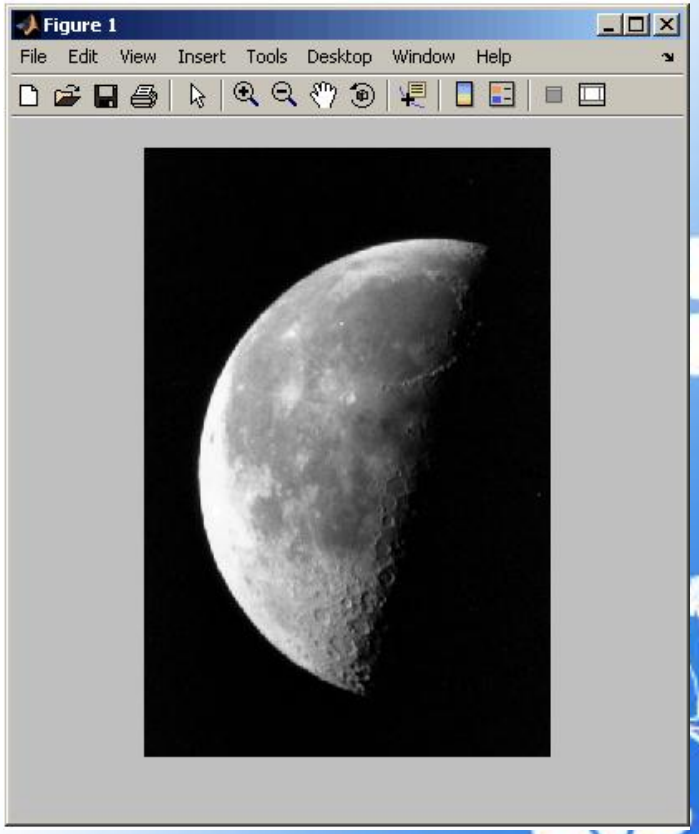
Cont..

Example

moon = imread (‘moon.tif’);

imshow (moon);

Output



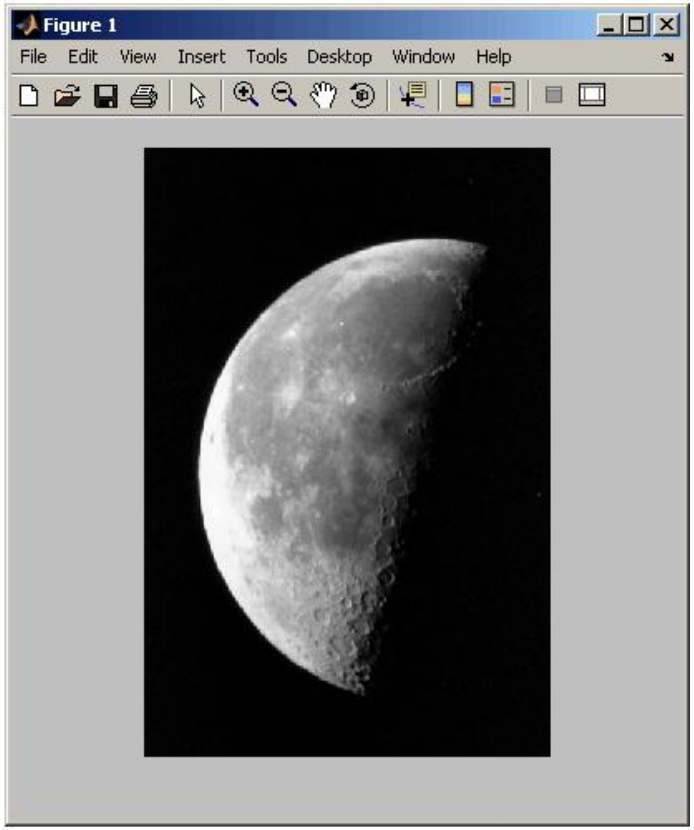
Controlling the Appearance of the Figure

By default, when imshow displays an image in a figure, it surrounds the image with a gray border. You can change this default and suppress the border using the 'border' parameter, as shown in the following example.

imshow ('moon.tif', 'Border','tight ')

Cont..

With Border



Without Border

Displaying Each Image in a Separate Figure

The simplest way to display multiple images is to display them in separate figure windows. MATLAB does not place any restrictions on the number of images you can display simultaneously.

```
imshow(I(:,:,:,1))
figure, imshow(I(:,:,:,2))
figure, imshow(I(:,:,:,3))
```

Dividing a Figure Window into Multiple Display Regions

subplot divides a figure into multiple display regions. The syntax of subplot is

subplot (m, n, p)

Cont..

Example

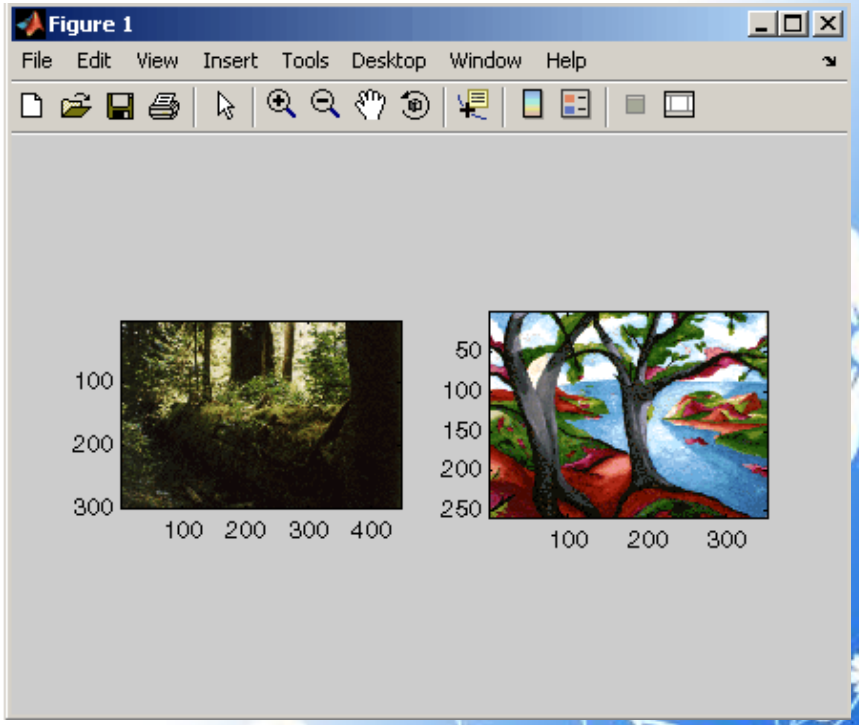
I = imread (‘forest.tif’);

J = imread (‘trees.tif’);

subplot (1, 2, 1), imshow (I)

subplot (1, 2, 2), imshow (J)

Output



Opening the Image Tool

To start the Image Tool, use the imtool function. You can also start another Image Tool from within an existing Image Tool by using the New option from the File menu.

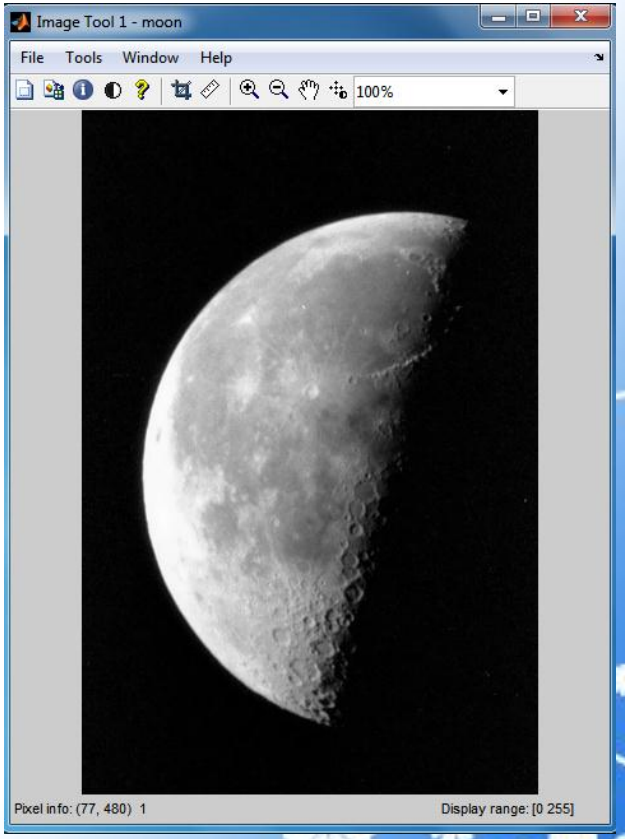
Cont..

Example

moon = imread (‘moon.tif’);

imtool (moon)

Output

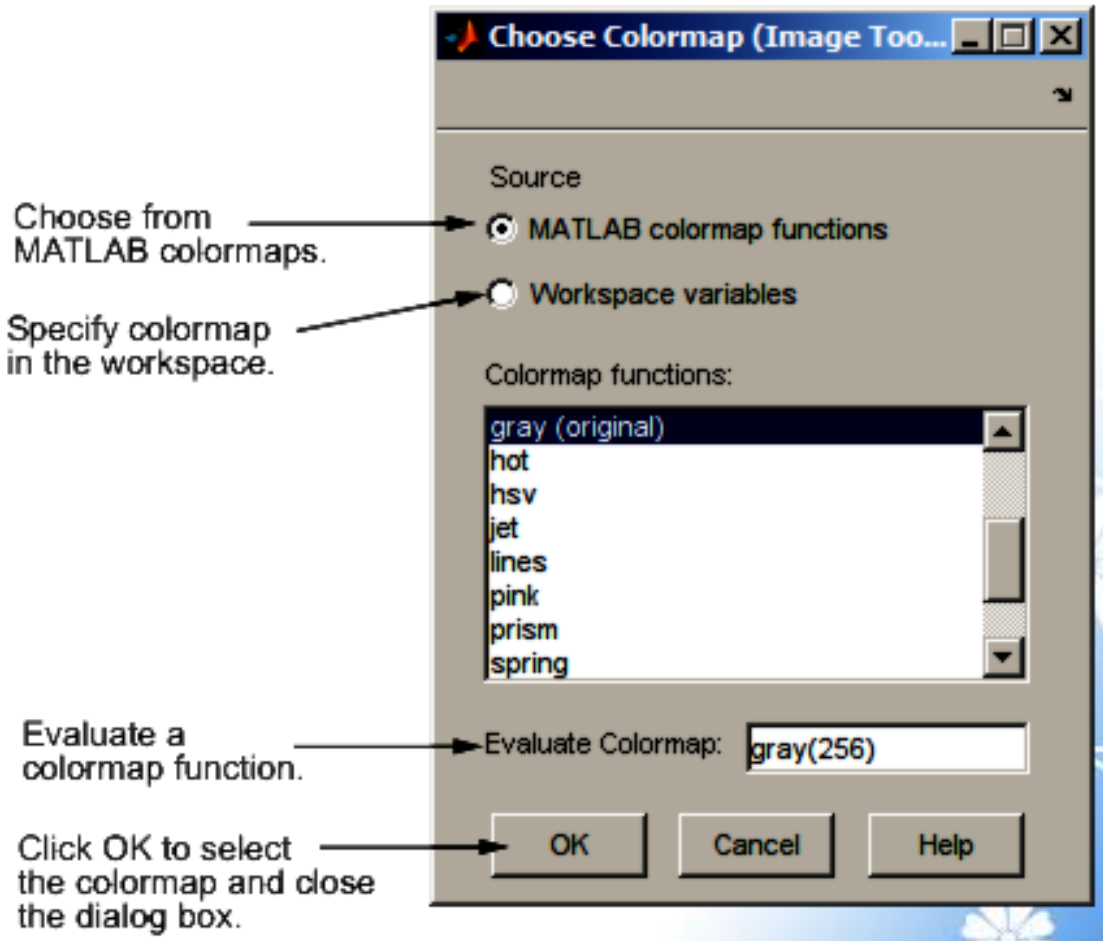


Specifying the Colormap

A colormap is a matrix that can have any number of rows, but must have three columns. Each row in the colormap is interpreted as a color, with the first element specifying the intensity of red, the second green, and the third blue.

To specify the color map used to display an indexed image or a grayscale image in the Image Tool, select the Choose Colormap option on the Tools menu. This activates the Choose Colormap tool. Using this tool you can select one of the MATLAB colormaps or select a colormap variable from the MATLAB workspace.

Cont..


Choose Colormap (Image Too...
Source
MATLAB colormap functions
Workspace variables
Colormap functions:
gray (original)
hot
hsv
jet
lines
pink
prism
spring
Evaluate Colormap:
gray(256)
OK
Cancel
Help
Choose from MATLAB colormaps.
Specify colormap in the workspace.
Evaluate a colormap function.
Click OK to select the colormap and close the dialog box.

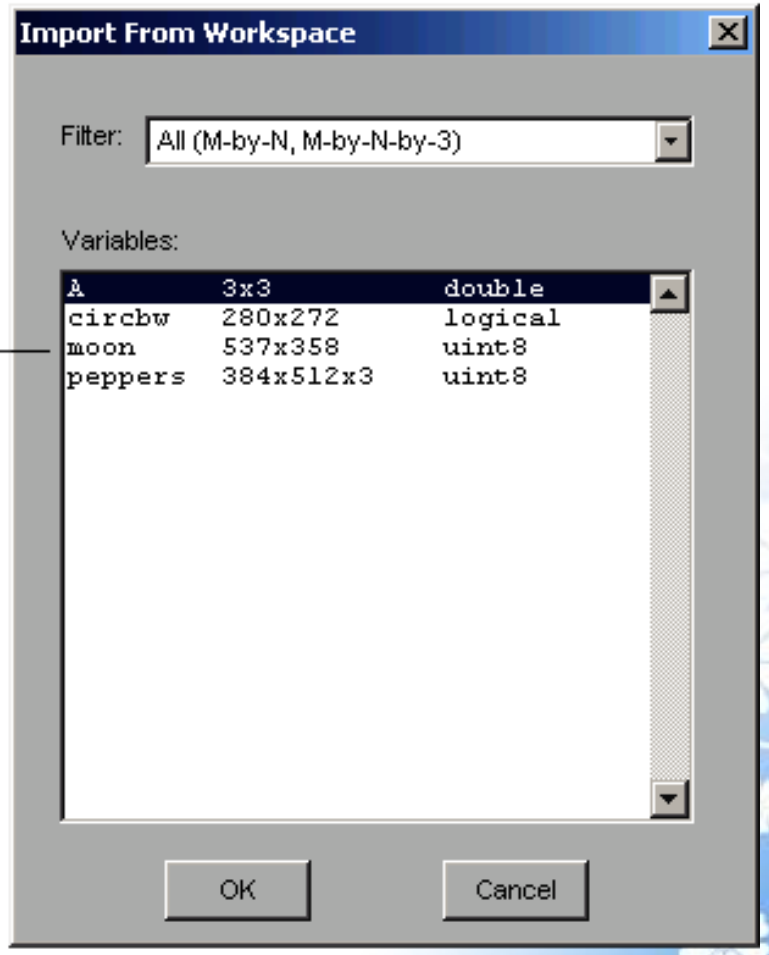
Importing Image Data from the Workspace

To import image data from the MATLAB workspace into the Image Tool, use the Import from Workspace option on the Image Tool File menu. In the dialog box, shown below, you select the workspace variable that you want to import into the workspace.

The following figure shows the Import from Workspace dialog box. You can use the Filter menu to limit the images included in the list to certain image types, i.e., binary, indexed, intensity (grayscale), or truecolor.

Cont..

Select a workspace variable.



Exporting Image Data to the Workspace

To export the image displayed in the Image Tool to the MATLAB workspace, you can use the Export to Workspace option on the Image Tool File menu. (Note that when exporting data, changes to the display range will not be preserved.) In the dialog box, shown below, you specify the name you want to assign to the variable in the workspace. By default, the Image Tool prefills the variable name field with BW, for binary images, RGB, for truecolor images, and I for grayscale or indexed images.

Cont..


Specify name of the workspace variable.
Export Image T...
Image variable name:
RGB
OK
Cancel

Using the getimage Function to Export Image Data

You can also use the getimage function to bring image data from the Image Tool into the MATLAB workspace.

moon = getimage (imgca)

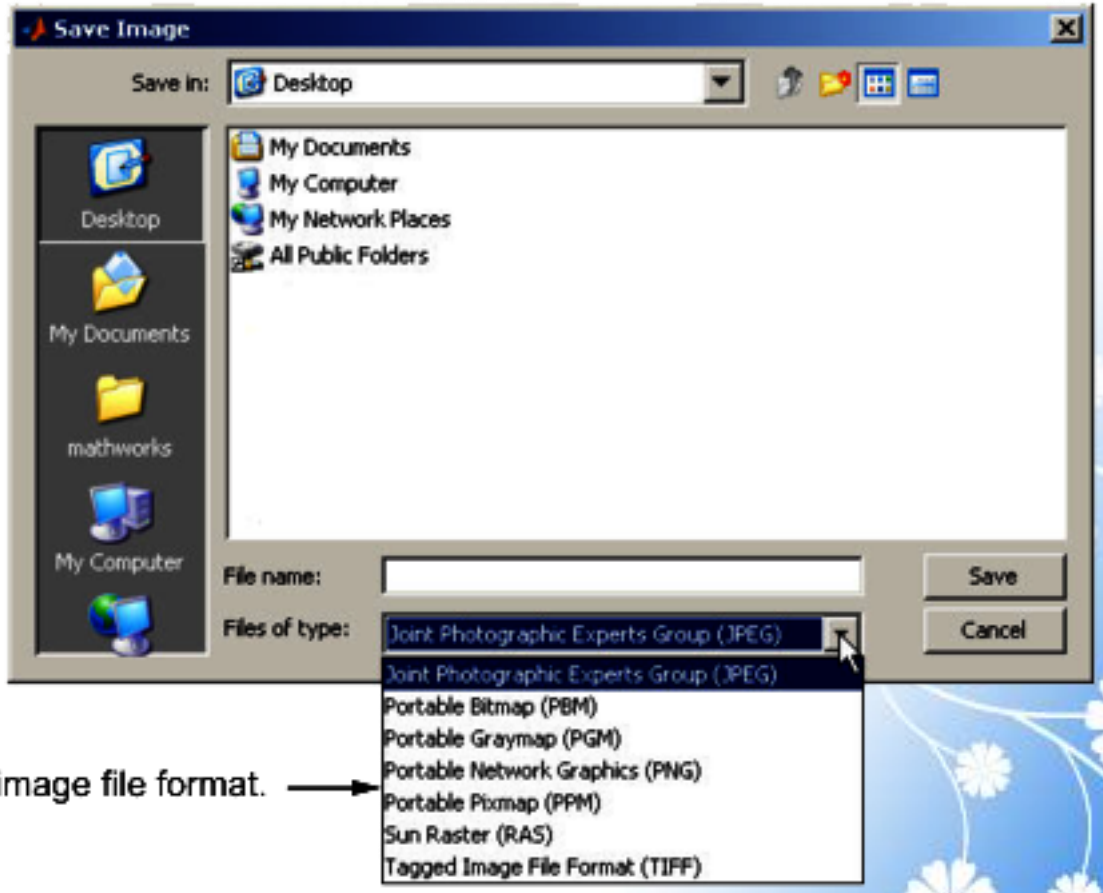
Saving the Image Data Displayed in the Image Tool

To save the image data displayed in the Image Tool, select the Save as option from the Image Tool File menu. The Image Tool opens the Save Image dialog box, shown in the following figure. Use this dialog box to navigate your file system to determine where to save the image file and specify the name of the file.

Cont..

Choose the graphics file format you want to use from among many common image file formats listed in the Files of Type menu. If you do not specify a file name extension, the Image Tool adds an extension to the file associated with the file format selected, such as .jpg for the JPEG format.

Cont..



Select image file format.

Printing the Image in the Image Tool

To print the image displayed in the Image Tool, select the Print to Figure option from the File menu. The Image Tool opens another figure window and displays the image. Use the Print option on the File menu of this figure window to print the image.

Exploring Very Large Images

If you are viewing a very large image, it might not load in imtool, or it could load, but zooming and panning are slow. In either case, creating a reduced resolution data set (R-Set) can improve performance. Use imtool to navigate an R-Set image the same way you navigate a standard image.

Creating an R-Set File

To create an R-Set file, use the function rsetwrite. For example, to create an R-Set from a TIFF file called 'LargeImage.tif', enter the following:

rsetfile = rsetwrite (‘LargeImage.tif’)

Opening an R-Set File

Open an R-Set file from the command line:

imtool (‘LargeImage.rset’)

Navigating an Image Using the Overview Tool

If an image is large or viewed at a large magnification, the Image Tool displays only a portion of the entire image, including scroll bars to allow navigation around the image. To determine which part of the image is currently visible in the Image Tool, use the Overview tool. The Overview tool displays the entire image, scaled to fit. Superimposed over this view of the image is a rectangle, called the detail rectangle. The detail rectangle shows which part of the image is currently visible in the Image Tool. You can change the portion of the image visible in the Image Tool by moving the detail rectangle over the image in the Overview tool.

Cont..

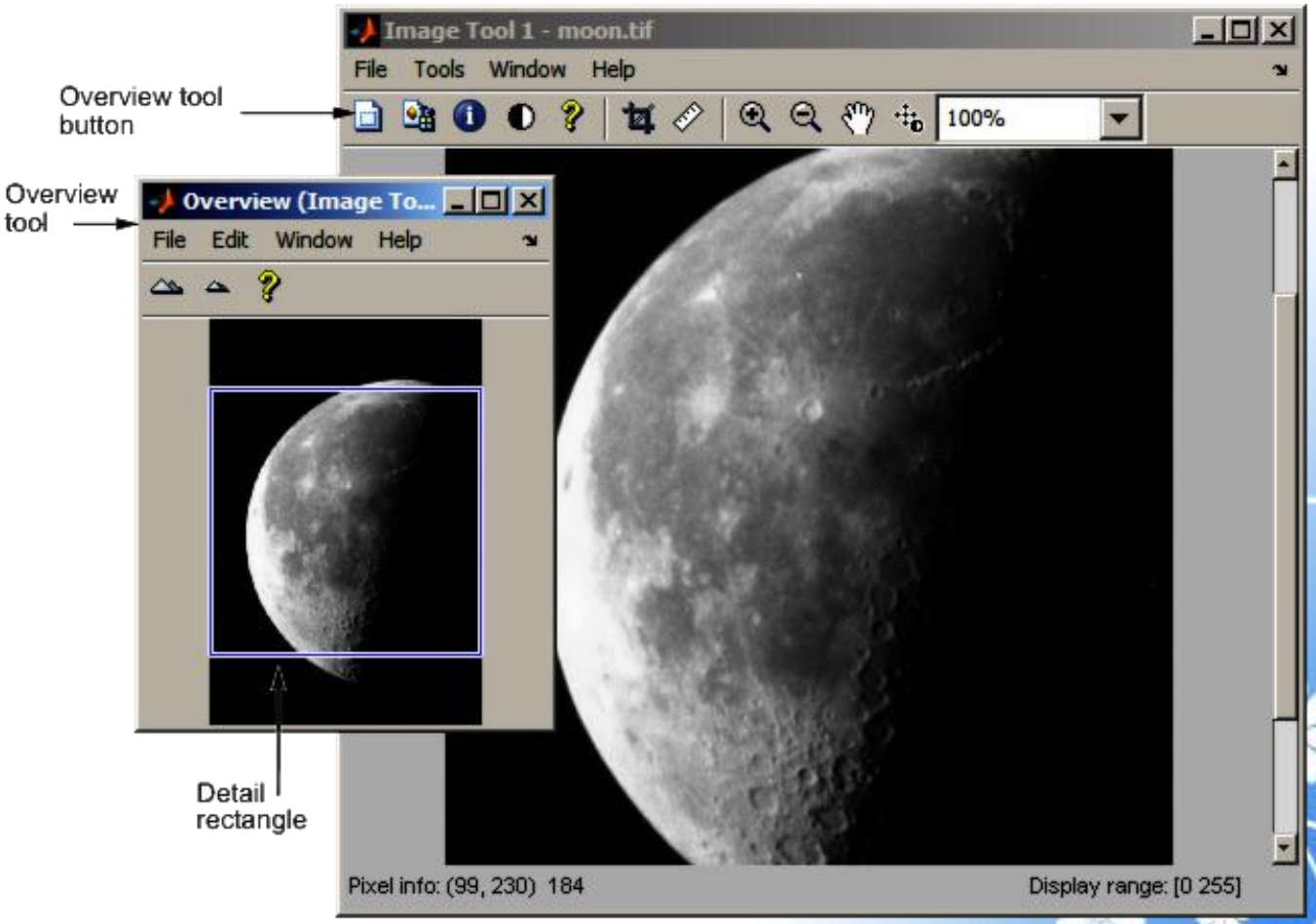


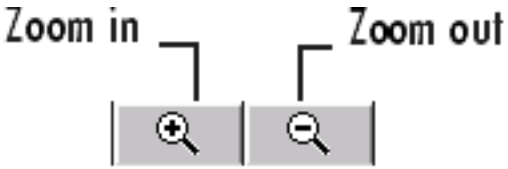
Image Tool 1 - moon.tif
File Tools Window Help
100%
Overview tool button
Overview tool
Overview (Image To...
File Edit Window Help
Detail rectangle
Pixel info: (99, 230) 184
Display range: [0 255]

Panning the Image Displayed in the Image Tool

To change the portion of the image displayed in the Image Tool, you can use the Pan tool to move the image displayed in the window. This is called panning the image. 

Zooming In and Out on an Image in the Image Tool

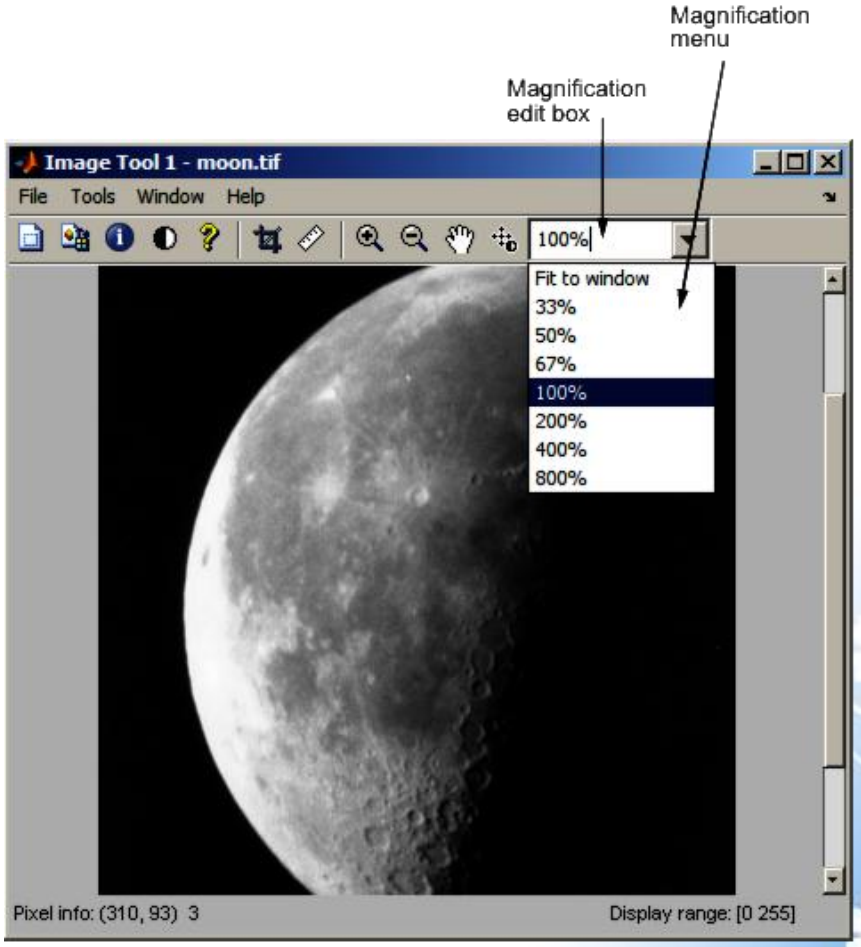
To enlarge an image to get a closer look or shrink an image to see the whole image in context, use the Zoom buttons on the toolbar. (You can also zoom in or out on an image by changing the magnification — see Specifying the Magnification of the Image or by using the Ctrl+Plus or Ctrl+Minus keys. Note that these are the Plus(+) and Minus(-) keys on the numeric keypad of your keyboard.)



Specifying the Magnification of the Image

To enlarge an image to get a closer look or to shrink an image to see the whole image in context, you can use the magnification edit box, shown in the following figure. (You can also use the Zoom buttons to enlarge or shrink an image.

Cont..


Magnification menu
Magnification edit box
Image Tool 1 - moon.tif
File Tools Window Help
100%
Fit to window
33%
50%
67%
100%
200%
400%
800%
Pixel info: (310, 93) 3
Display range: [0 255]

Determining the Value of Individual Pixels

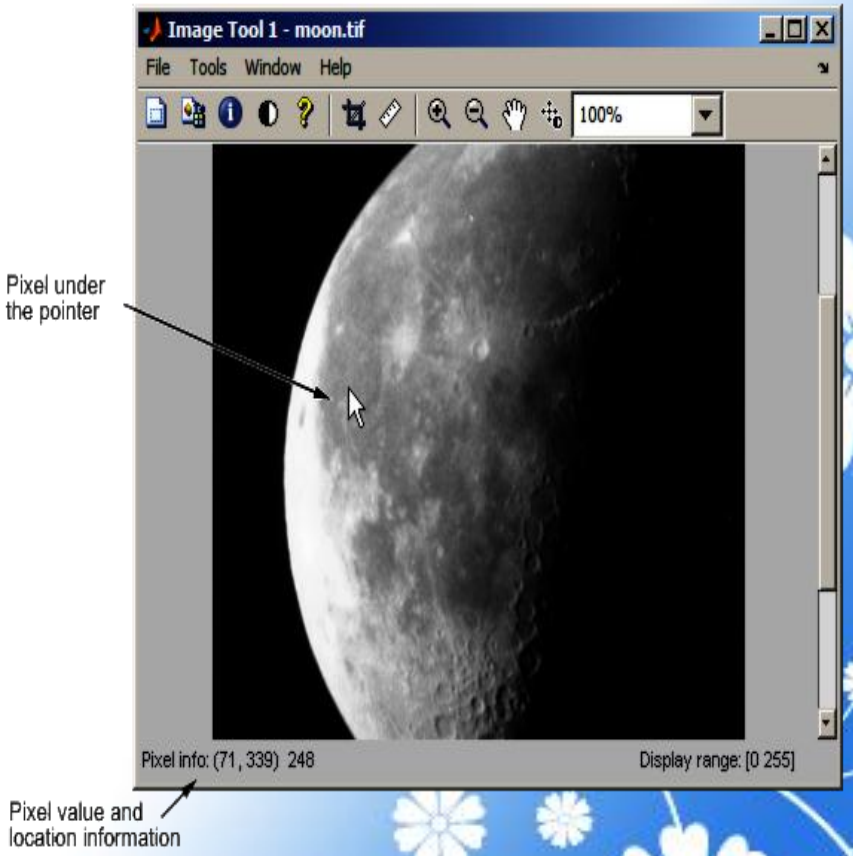
The Image Tool displays information about the location and value of individual pixels in an image in the bottom left corner of the tool. (You can also obtain this information by opening a figure with imshow and then calling impixelinfo from the command line.) The pixel value and location information represent the pixel under the current location of the pointer. The Image Tool updates this information as you move the pointer over the image.

Cont..

Example

imtool (‘moon.tif’)

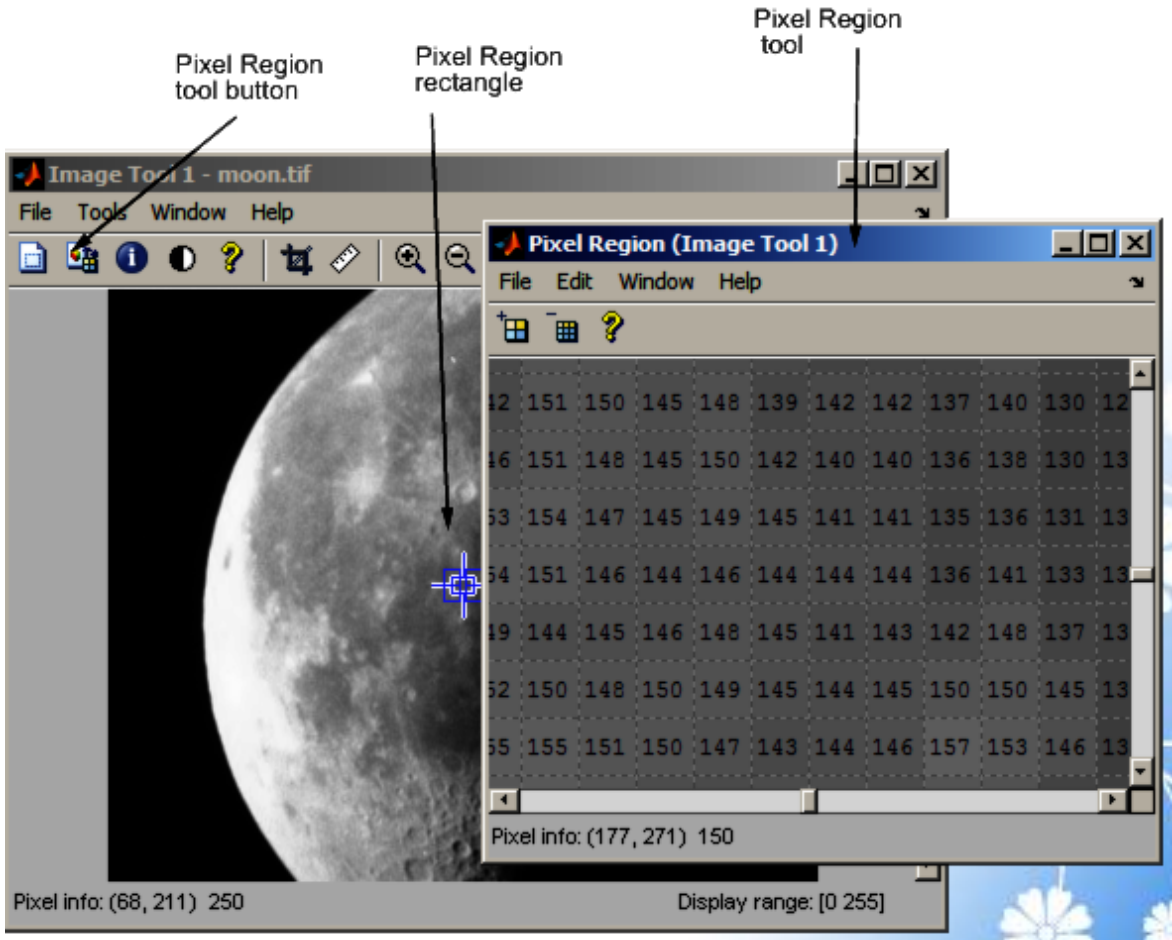
Output



Determining the Values of a Group of Pixels

To view the values of pixels in a specific region of an image displayed in the Image Tool, use the Pixel Region tool. The Pixel Region tool superimposes a rectangle, called the pixel region rectangle, over the image displayed in the Image Tool. This rectangle defines the group of pixels that are displayed, in extreme close-up view, in the Pixel Region tool window. The following figure shows the Image Tool with the Pixel Region tool. Note how the Pixel Region tool includes the value of each pixel in the display.

Cont..


Pixel Region tool button
Pixel Region rectangle
Pixel Region tool
Image Tool 1 - moon.tif
File Tools Window Help
Pixel Region (Image Tool 1)
File Edit Window Help
Pixel info: (177, 271) 150
Pixel info: (68, 211) 250
Display range: [0 255]

Determining the Display Range of an Image

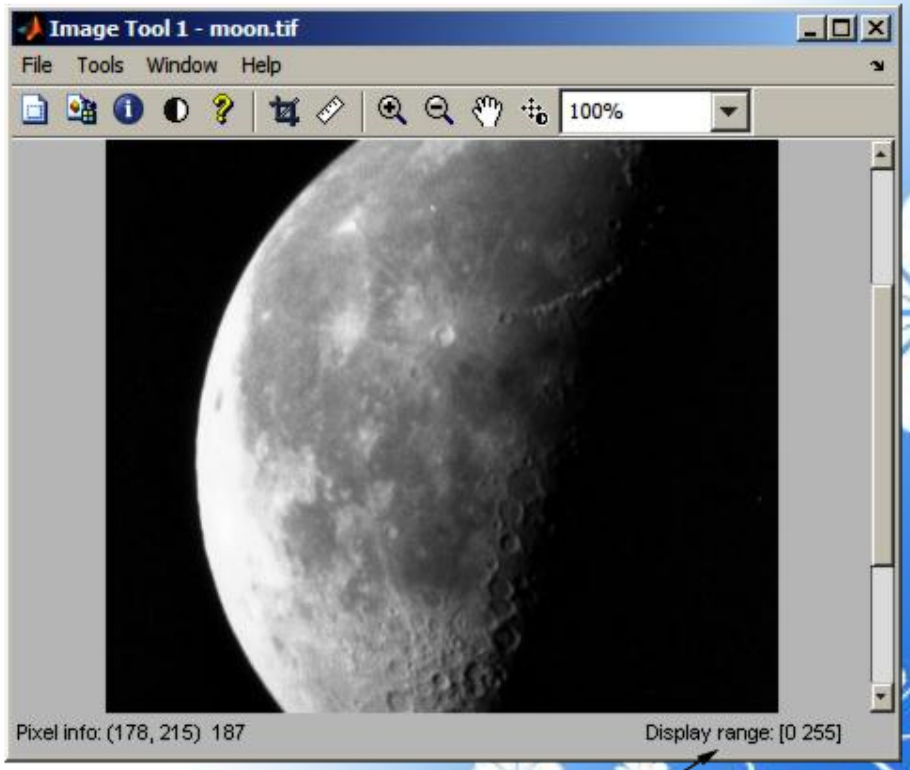
The Image Tool displays this information in the Display Range tool at the bottom right corner of the window. The Image Tool does not show the display range for indexed, truecolor, or binary images. (You can also obtain this information by opening a figure window with imshow and then calling imdisplayrange from the command line.)

Cont..

Example

imtool (‘moon.tif’)

Output



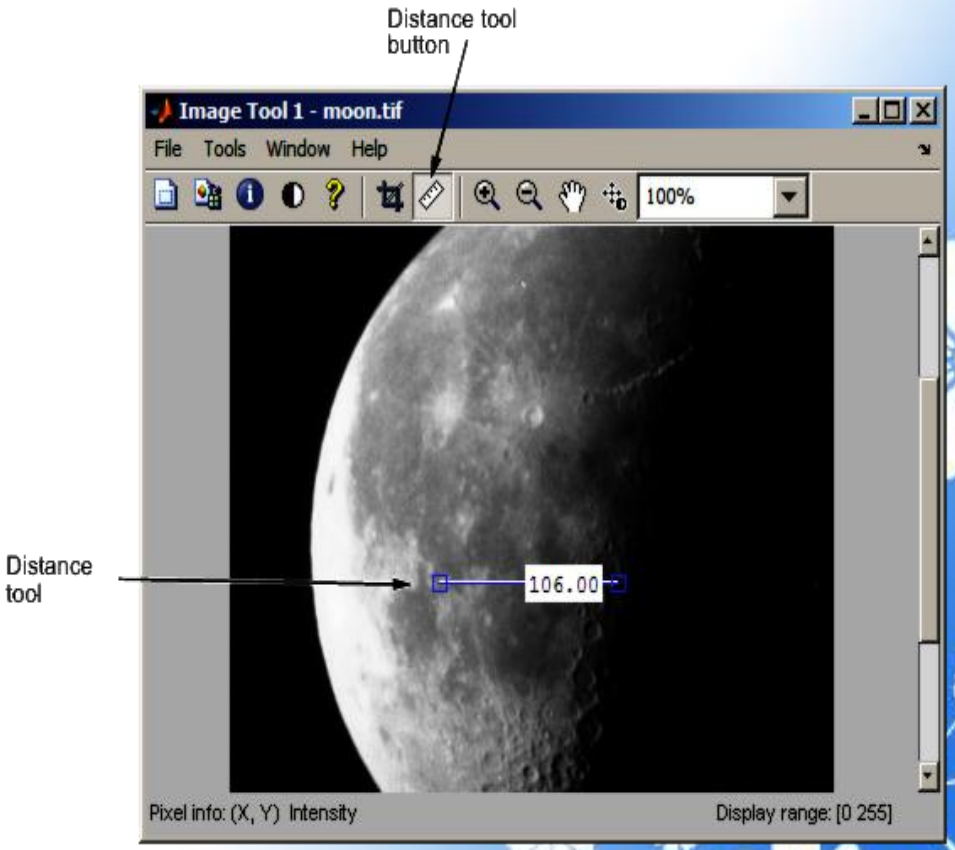
Display Range tool

Measuring the Distance Between Two Pixels

Example

imtool (‘moon.tif’)

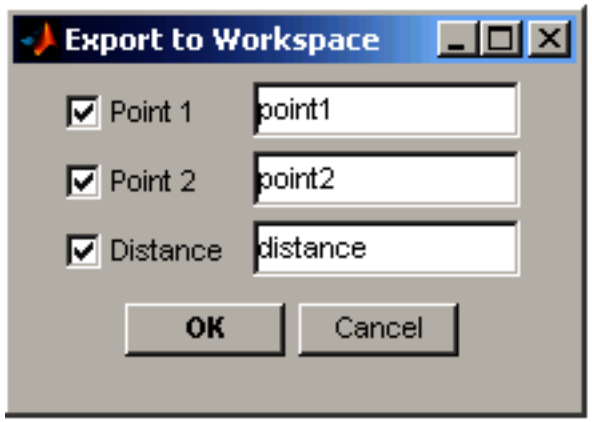
Output



Exporting Endpoint and Distance Data

To save the endpoint locations and distance information, right-click the Distance tool and choose the Export to Workspace option from the context menu. The Distance tool opens the Export to Workspace dialog box. You can use this dialog box to specify the names of the variables used to store this information.

Cont..


Export to Workspace
Point 1
point1
Point 2
point2
Distance
distance
OK
Cancel

Cont..

After you click OK, the Distance tool creates the variables in the workspace, as in the following example.

```
whos
```

Name	Size	Bytes	Class	Attributes
distance	1x1	8	double	
point1	1x2	16	double	
point2	1x2	16	double	

Customizing the Appearance of the Distance Tool

Using the Distance tool context menu, you can customize many aspects of the Distance tool appearance and behavior. Position the pointer over the line and right-click to access these context menu options.

- Toggling the distance tool label on and off using the Show Distance Label option.

Cont..

- Changing the color used to display the Distance tool line using the Set color option.
- Constraining movement of the tool to either horizontal or vertical using the Constrain drag option.
- Deleting the distance tool object using the Delete option.
- Right-click the Distance tool to access this context menu.

Getting Information About an Image Using the Image Information Tool

To get information about the image displayed in the Image Tool, use the Image Information tool. The Image Information tool can provide two types of information about an image:

- Basic information — Includes width, height, class, and image type. For grayscale and indexed images, this information also includes the minimum and maximum intensity values.
- Image metadata — Displays all the metadata from the graphics file that contains the image. This is the same information returned by the imfinfo function or the dicominfo function.

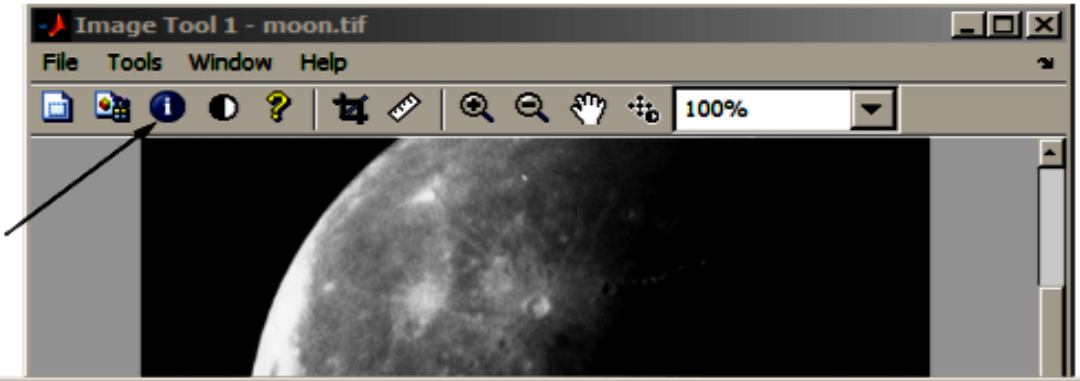


Image Information tool button

Image Information tool

Basic image information

Image metadata

Image Information (Image Tool 1)

Image details (Image Tool 1 - moon.tif)

Attribute	Value
Width (columns)	358
Height (rows)	537
Class	uint8
Image type	intensity
Minimum intensity	0
Maximum intensity	253

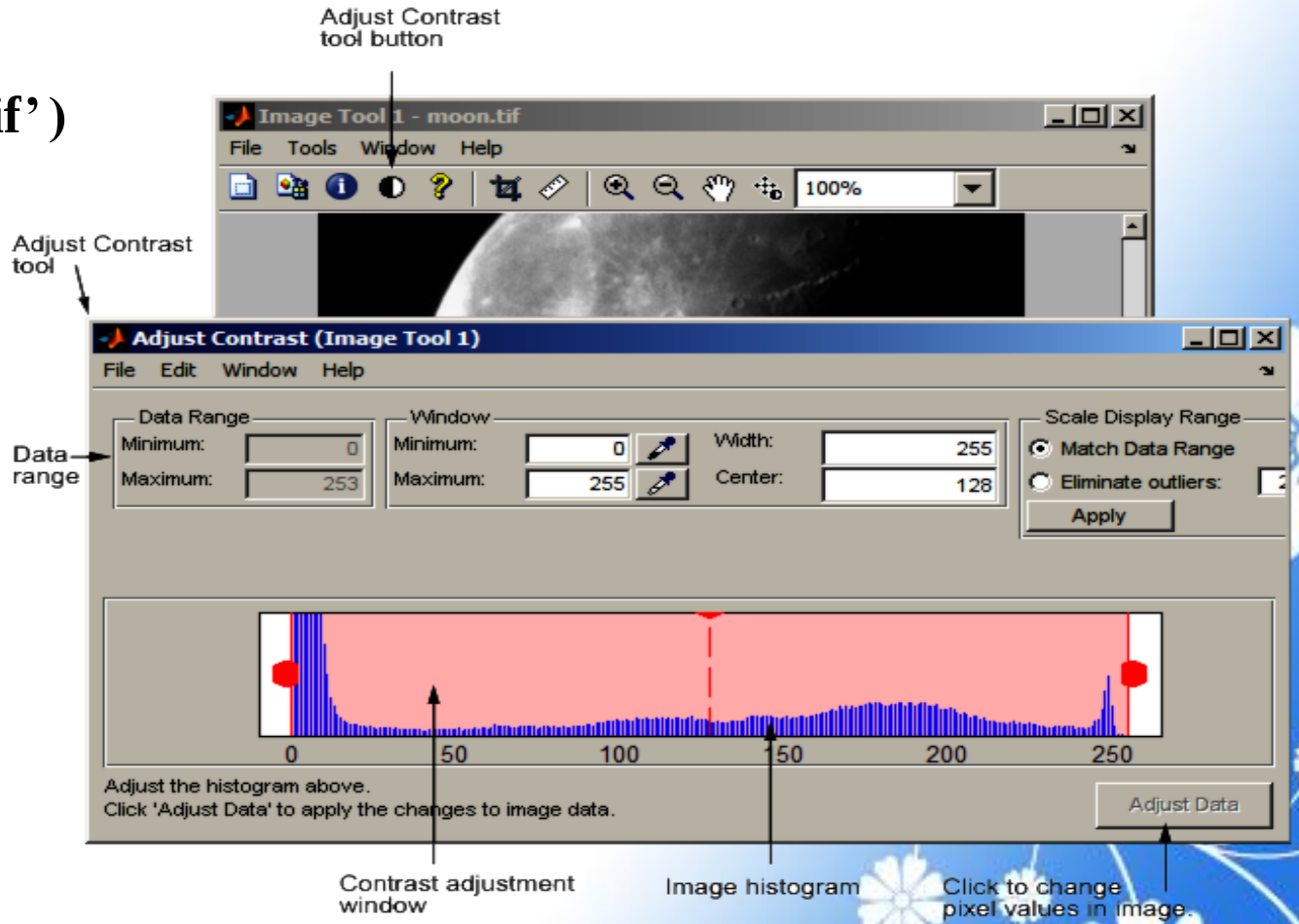
Metadata (moon.tif)

Attribute	Value
Filename	F:\bat\Aimage\perfect\matlab\toolbox\images\imdemos\moon.tif
FileModDate	04-Dec-2000 13:57:59
FileSize	183950
Format	tif
FormatVersion	[]
Width	358
Height	537
BitDepth	8
ColorType	grayscale
FormatSignature	[73 73 42 0]
ByteOrder	little-endian
NewSubFileType	0
BitsPerSample	8

Starting the Adjust Contrast Tool

Example:

imtool (‘moon.tif’)



Viewing Image Sequences as a Montage

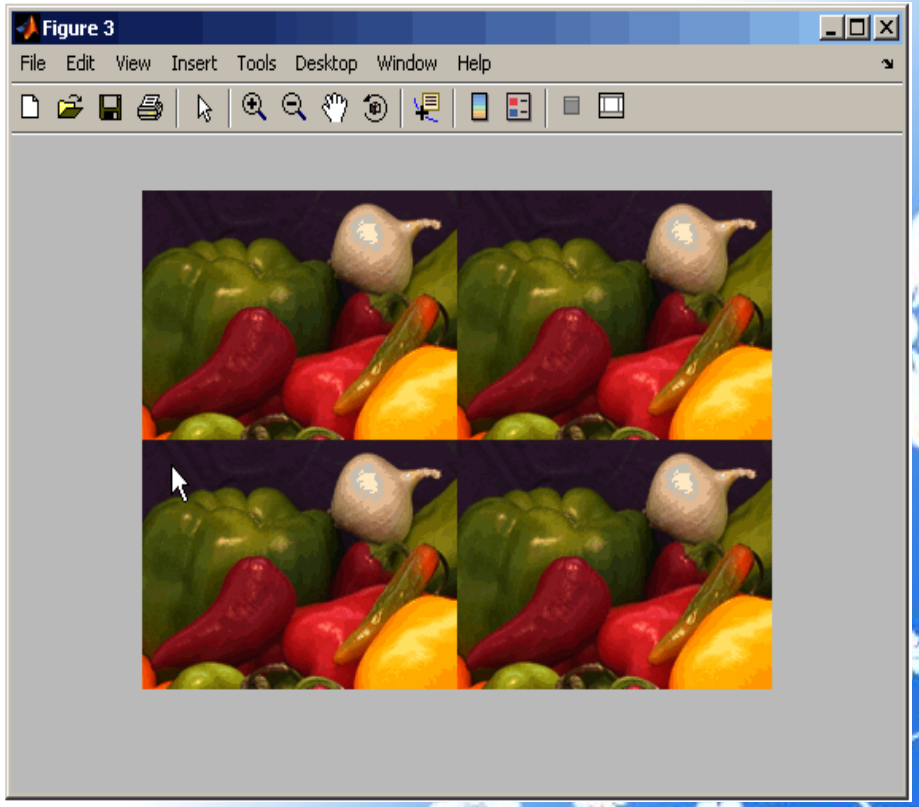
To view multiple frames in a multiframe array at one time, use the montage function. montage displays all the image frames, arranging them into a rectangular grid. The montage of images is a single image object. The image frames can be grayscale, indexed, or truecolor images. If you specify indexed images, they all must use the same colormap.

Cont..

Example

```
onion = imread('onion.png');
onionArray = repmat(onion, [ 1 1 1 4 ]);
montage(onionArray);
```

Output



Displaying Binary Images

In MATLAB, a binary image is of class logical. Binary images contain only 0's and 1's. Pixels with the value 0 are displayed as black; pixels with the value 1 are displayed as white.

Cont..

Example

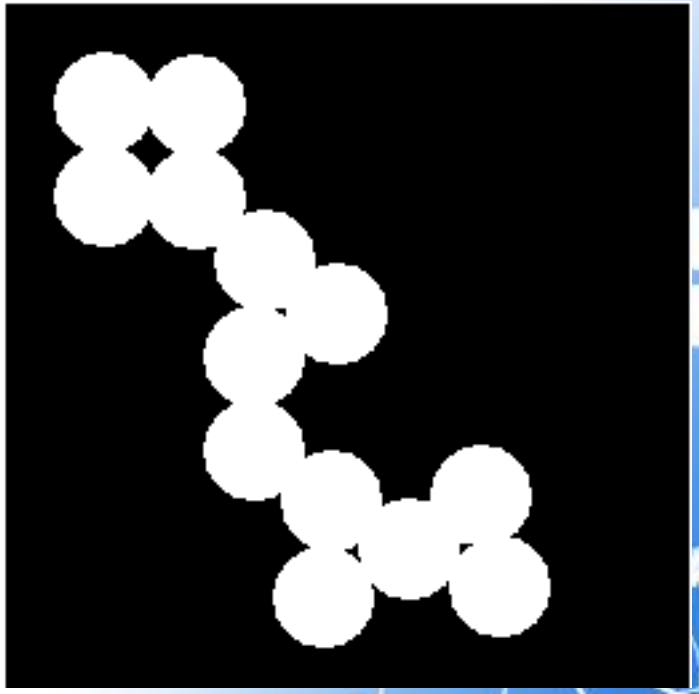
BW = imread ('circles.png');

imshow (BW)

or

imtool (BW)

Output



Displaying Truecolor Images

Example

RGB = imread(‘peppers.png’);

imshow(RGB)

or

imtool(RGB)

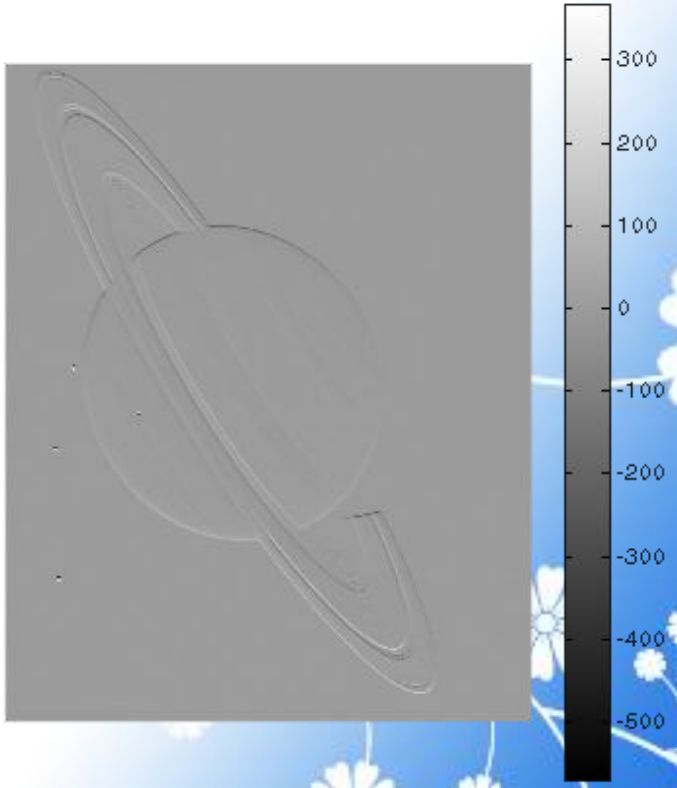
Output

Adding a Colorbar to a Displayed Image

Example

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
h = [1 2 1; 0 0 0; -1 -2 -1];
I2 = filter2(h,I);
imshow(I2,'DisplayRange',[]),
colorbar
```

Output



Printing Images

If you want to output a MATLAB image to use in another application (such as a word-processing program or graphics editor), use imwrite to create a file in the appropriate format.

Building GUIs with Modular Tools

Associating Modular Tools with a Particular Image

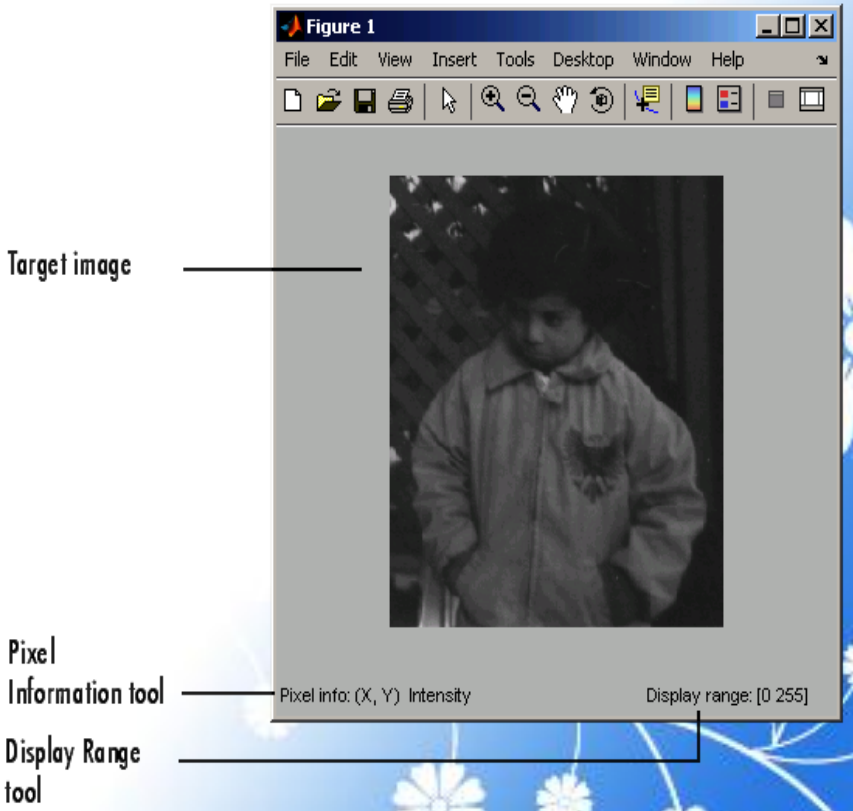
Example

himage = imshow('pout.tif');

hpixelinfopanel = impixelinfo(himage);

hdrangepanel = imdisplayrange(himage);

Output

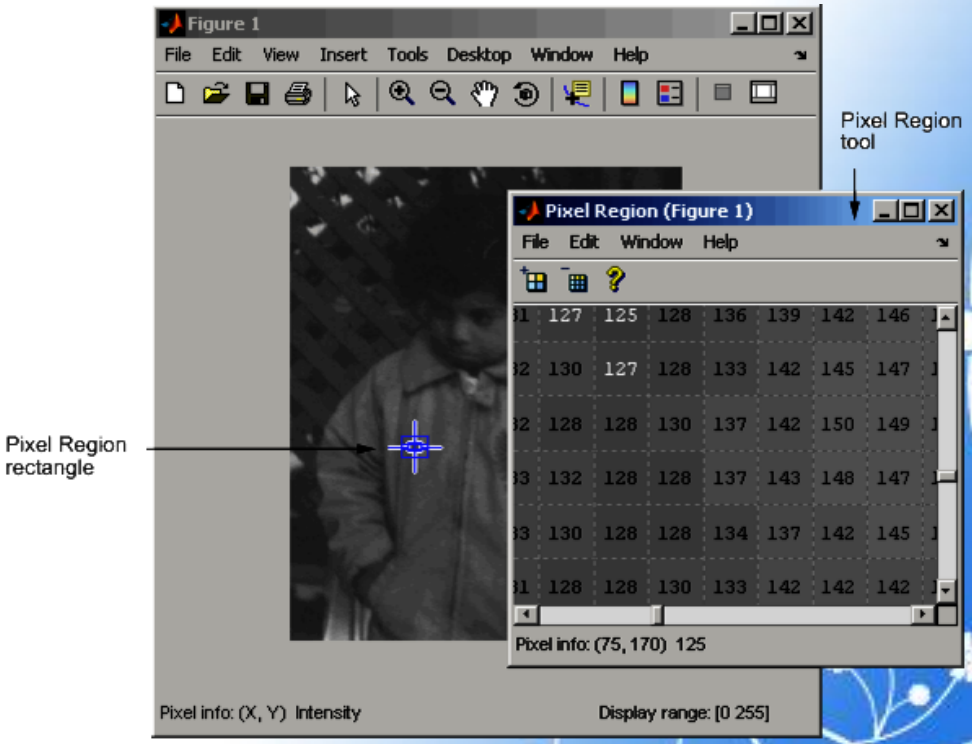


Example: Embedding the Pixel Region Tool in an Existing Figure

Example

```
himage = imshow('pout.tif')

hpixelinfopanel =

impixelinfo(himage);

hdrangepanel =

imdisplayrange(himage);

hpixreg = impixelregion(himage);
```

Output

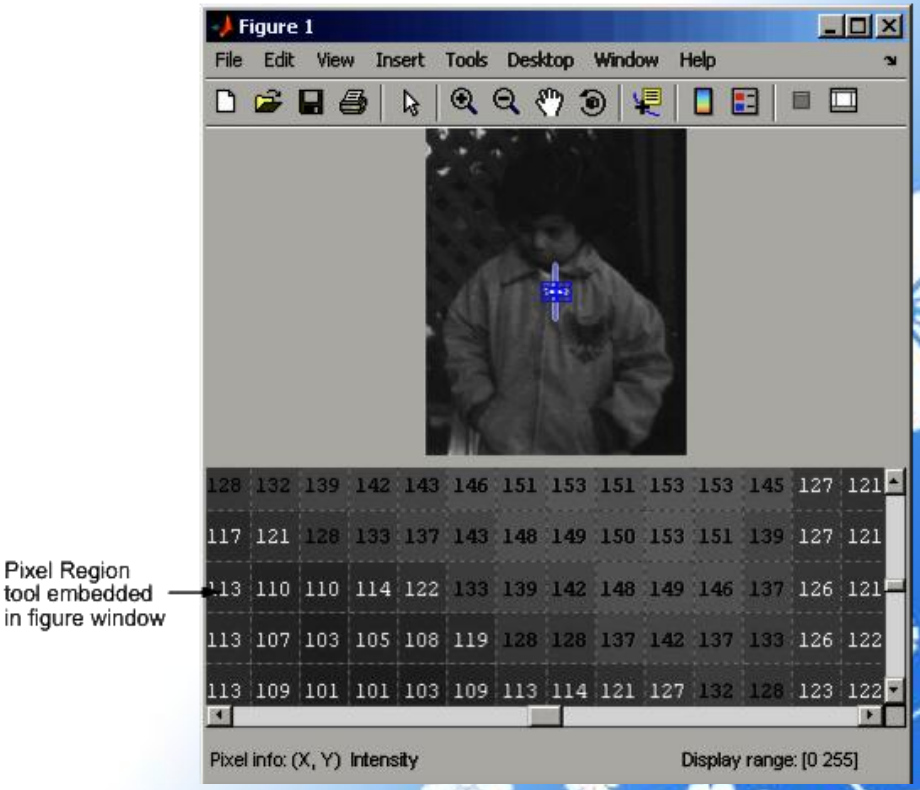


Cont..

Example

```
hfig = figure;
hax = axes('units','normalized','position',[0 .5 1
.5]);
himage = imshow('pout.tif')
hpixelinfopanel = impixelinfo(himage);
hdrangepanel = imdisplayrange(himage);
hpixreg = impixelregionpanel(hfig,himage);
set(hpixreg, 'Units','normalized','Position',[0
.08 1 .4]);
```

Output



Positioning the Modular Tools in a GUI

When you create the modular tools, they have default positioning behavior. For example, the impixelinfo function creates the tool as a uipanel object that is the full width of the figure window, positioned in the lower left corner of the target image figure window.

Because the modular tools are constructed from standard Handle Graphics objects, such as uipanel objects, you can use properties of the objects to change their default positioning or other characteristics.

Adding Navigation Aids to a GUI

The toolbox includes several modular tools that you can use to add navigation aids to a GUI application:

- Scroll Panel
- Overview tool
- Magnification box

Cont..

The Scroll Panel is the primary navigation tool; it is a prerequisite for the other navigation tools. When you display an image in a Scroll Panel, the tool displays only a portion of the image, if it is too big to fit into the figure window. When only a portion of the image is visible, the Scroll Panel adds horizontal and vertical scroll bars, to enable viewing of the parts of the image that are not currently visible.

Understanding Scroll Panels

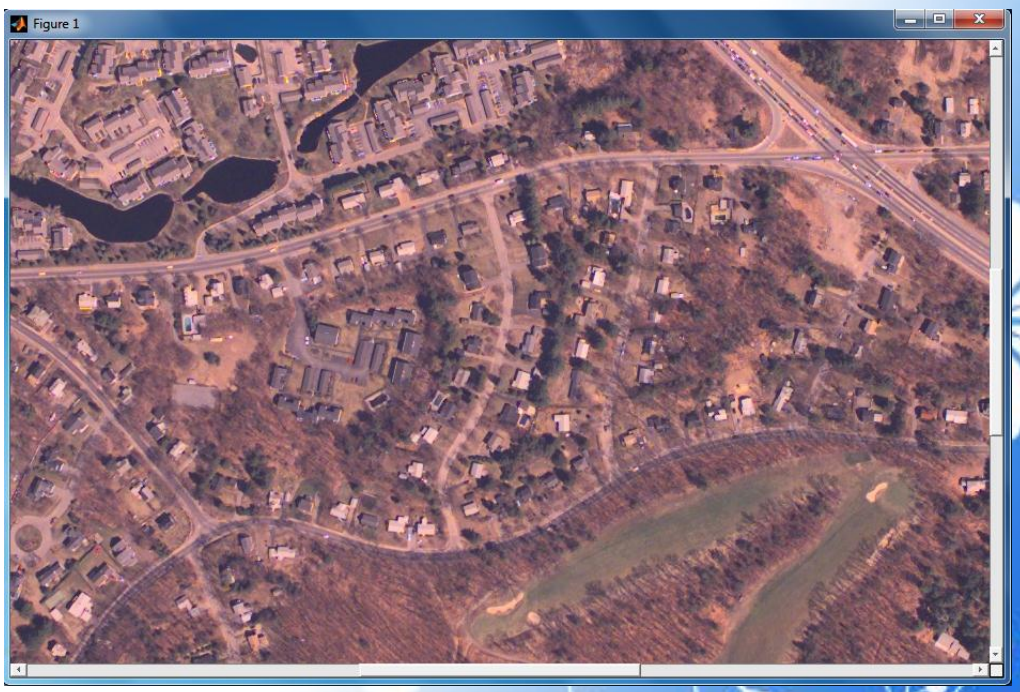
When you display an image in a scroll panel, it changes the object hierarchy of your displayed image. This diagram illustrates the typical object hierarchy for an image displayed in an axes object in a figure object.

Cont..

Example

```
himage =
imread('concordaerial.png');
hfig = figure('Toolbar','none',...
            'Menubar', 'none');
himage =
imshow('concordaerial.png');
hpanel = imscrollpanel(hfig,
himage)
```

Output



Example: Building a Navigation GUI for Large Images

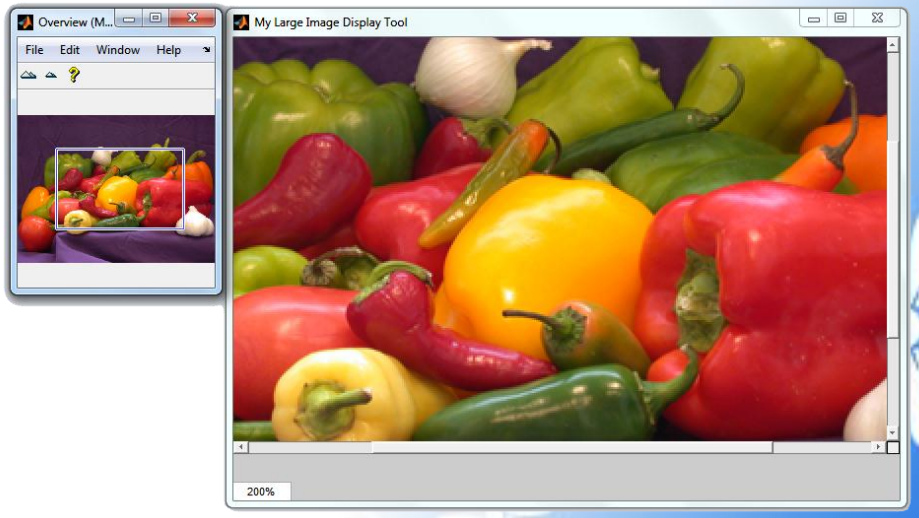
```
function my_large_image_display(im)

% Create a figure without toolbar and menubar.
hfig = figure('Toolbar','none',...
            'Menubar', 'none',...
            'Name','My Large Image Display
Tool',...
            'NumberTitle','off',...
            'IntegerHandle','off');

% Display the image in a figure with imshow.
himage = imshow(im);

% Add the scroll panel.
hpanel = imscrollpanel(hfig,himage);

% Position the scroll panel to accommodate the
other tools.
set(hpanel,'Units','normalized','Position',[0 .1 1
.9]);

% Add the Magnification box.
hMagBox = immagbox(hfig,himage);

% Position the Magnification box
pos = get(hMagBox,'Position');
set(hMagBox,'Position',[0 0 pos(3) pos(4)]);

% Add the Overview tool.
hovervw = imoverview(himage);
```

Cont..

Example

```
big_image = imread('peppers.png');
my_large_image_display(big_image)
```

Output



Spatial Transformations

Resizing an Image

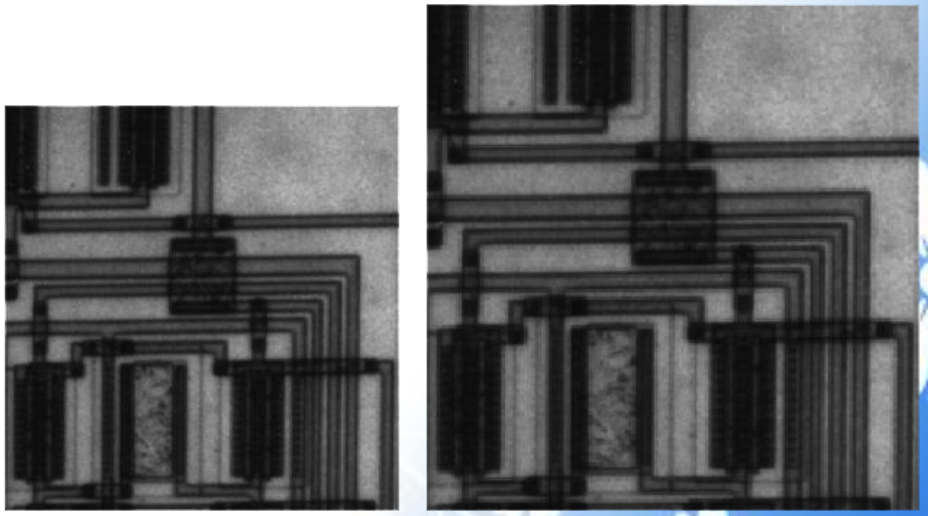
To resize an image, use the imresize function. When you resize an image, you specify the image to be resized and the magnification factor. To enlarge an image, specify a magnification factor greater than 1. To reduce an image, specify a magnification factor between 0 and 1.

Cont..

Example

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

Output

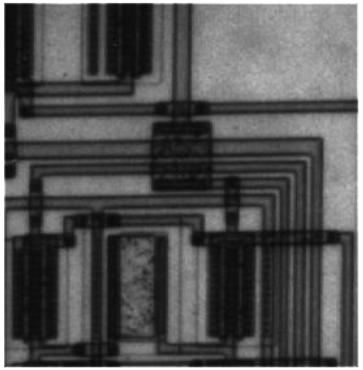


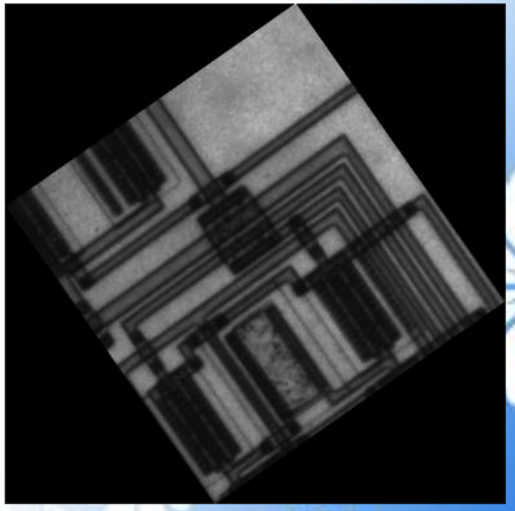
Rotating an Image

Example

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

J = imrotate(I,35,'bilinear');

imshow(I)

figure, imshow(J)
```

Output





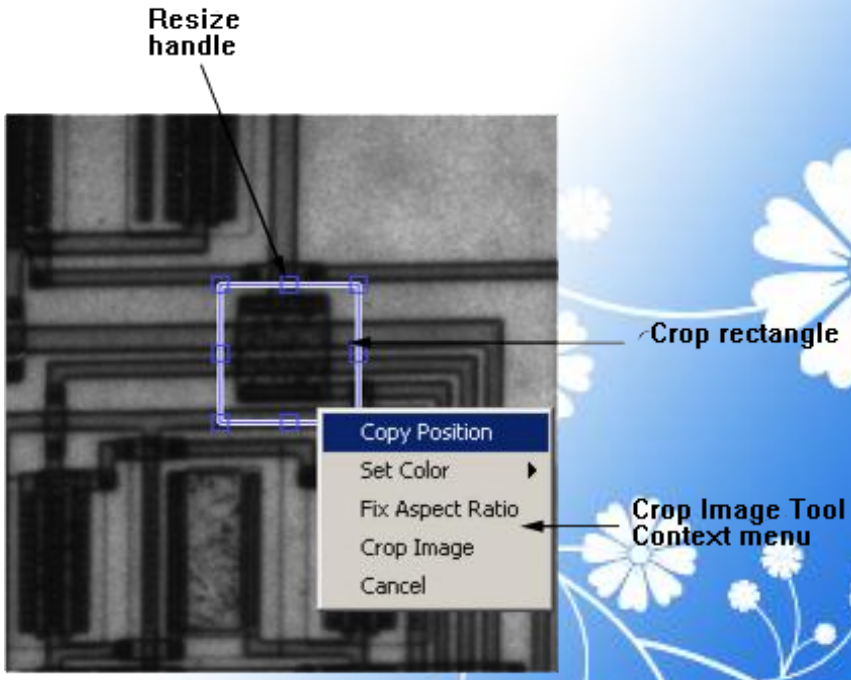
Cropping an Image

Example

I = imread('circuit.tif')

J = imcrop(I);

Output



Using a Transformation Matrix

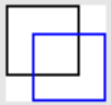
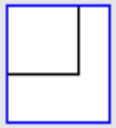
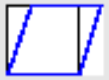
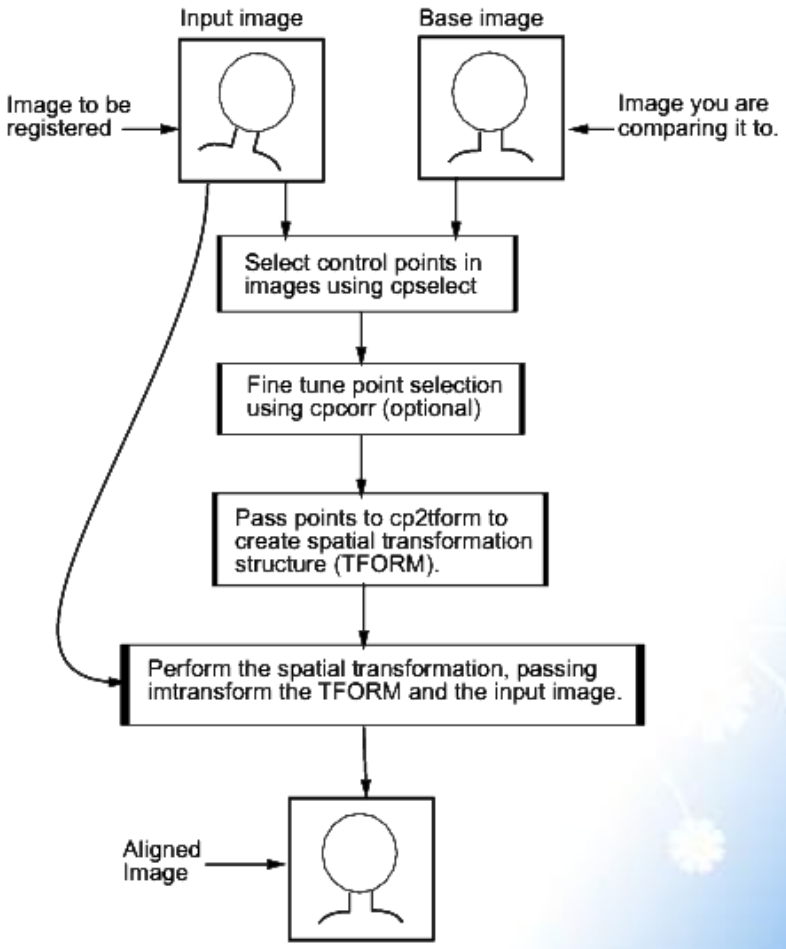
Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the *x* axis t_y specifies the displacement along the *y* axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the *x* axis s_y specifies the scale factor along the *y* axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the *x* axis sh_y specifies the shear factor along the *y* axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.

Image Registration

Control Point Registration



Step 1: Read the Images

Example

```
orthophoto =
imread('westconcordorthophoto.pn
g');
figure, imshow(orthophoto)
unregistered =
imread('westconcordaerial.png');
figure, imshow(unregistered)
```

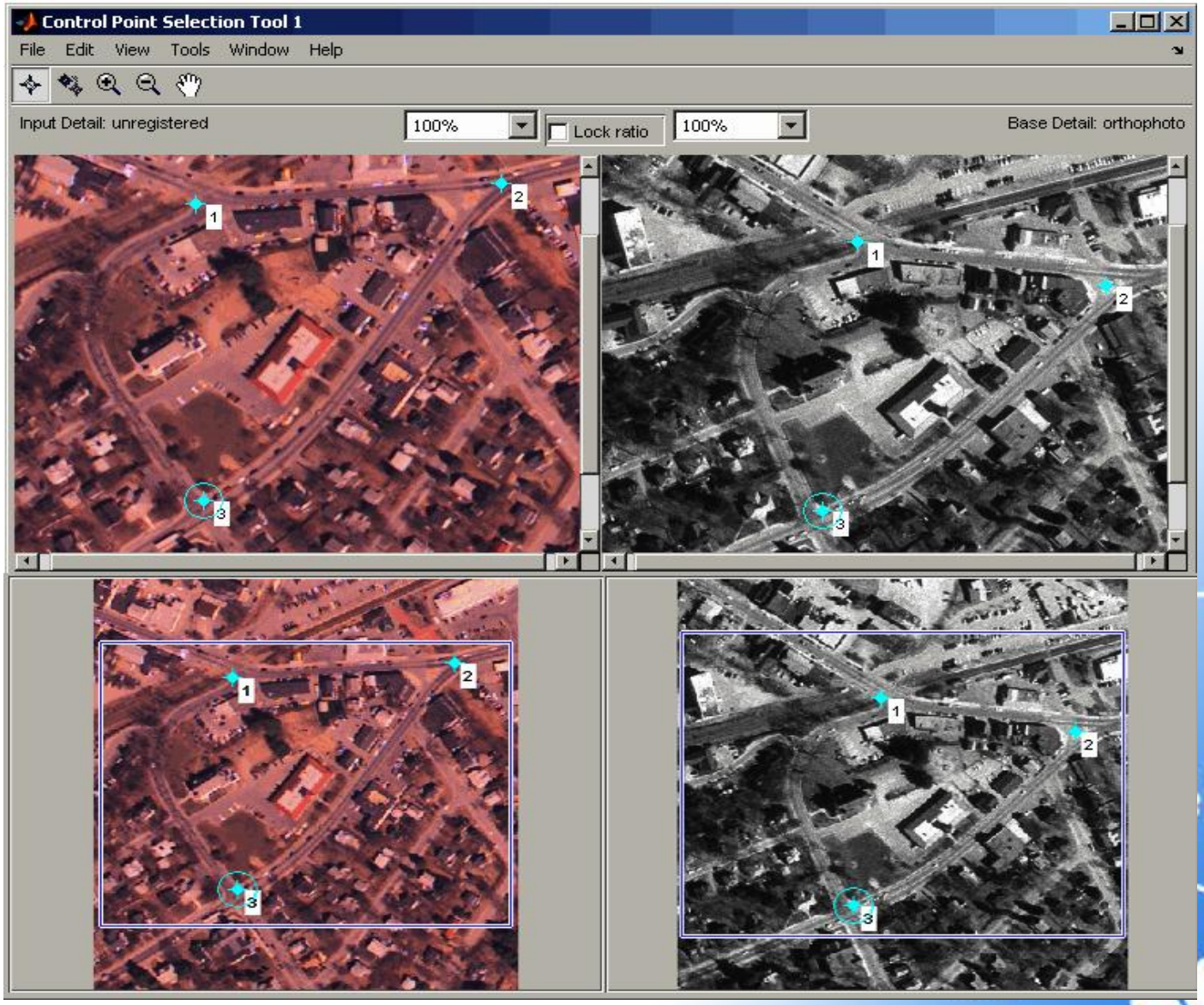
Output

Step 2: Choose Control Points in the Images

Example

cpselect(unregistered, orthophoto)

Output


Control Point Selection Tool 1
File Edit View Tools Window Help
Input Detail: unregistered
100%
Lock ratio
100%
Base Detail: orthophoto

Step 3: Save the Control Point Pairs to the MATLAB Workspace

For example, the following set of control points in the input image represent spatial coordinates; the left column lists x-coordinates, the right column lists y-coordinates.

```
input_points =

  118.0000   96.0000

  304.0000   87.0000

  358.0000  281.0000

  127.0000  292.0000
```

Step 4: Fine-Tune the Control Point Pair Placement (Optional)

This is an optional step that uses cross-correlation to adjust the position of the control points you selected with cpselect. To use cross-correlation, features in the two images must be at the same scale and have the same orientation. They cannot be rotated relative to each other. Because the Concord image is rotated in relation to the base image, cpcorr cannot tune the control points.

Step 5: Specify the Type of Transformation and Infer Its Parameters

In this step, you pass the control points to the cp2tform function that determines the parameters of the transformation needed to bring the image into alignment. cp2tform is a data-fitting function that determines the transformation based on the geometric relationship of the control points. cp2tform returns the parameters in a geometric transformation structure, called a TFORM structure.

```
mytform = cp2tform(input_points, base_points, 'projective');
```

Step 6: Transform the Unregistered Image

As the final step in image registration, transform the input image to bring it into alignment with the base image. You use imtransform to perform the transformation, passing it the input image and the TFORM structure, which defines the transformation. imtransform returns the transformed image.

Cont..

Example

registered =

imtransform(unregistered,

mytform);

Output

Specifying Control Points Using the Control Point Selection Tool

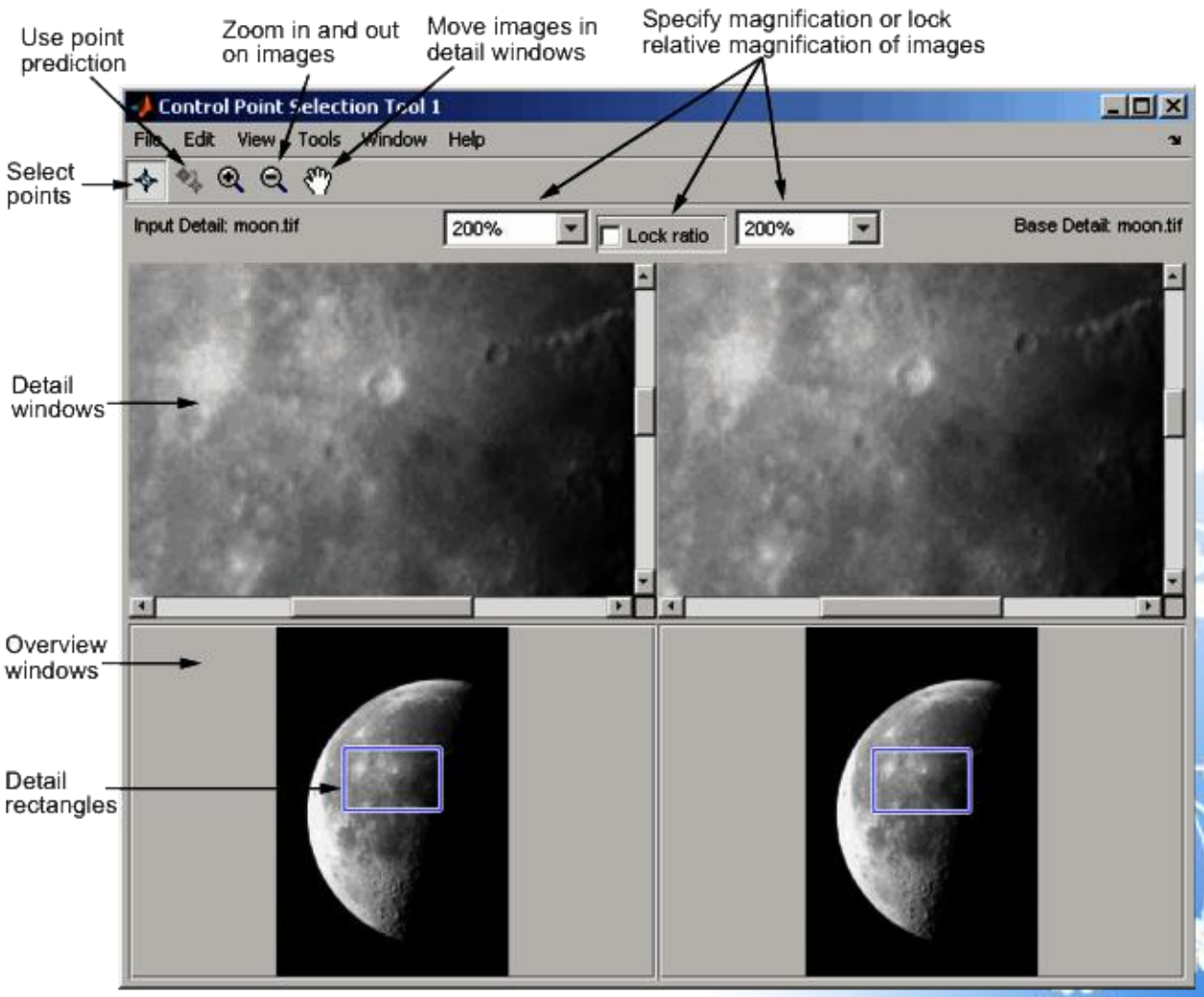
To specify control points in a pair of images you want to register, use the Control Point Selection Tool, cpselect. The tool displays the image you want to register, called the input image, next to the image you want to compare it to, called the base image or reference image.

Cont..

Specifying control points is a four-step process:

1. Start the tool, specifying the input image and the base image.
2. Use navigation aids to explore the image, looking for visual elements that you can identify in both images. cpselect provides many ways to navigate around the image, panning and zooming to view areas of the image in more detail.
3. Specify matching control point pairs in the input image and the base image.
4. Save the control points in the MATLAB workspace.

Cont..


Use point prediction
Zoom in and out on images
Move images in detail windows
Specify magnification or lock relative magnification of images
Control Point Selection Tool 1
File Edit View Tools Window Help
Select points
Input Detail: moon.tif
200%
Lock ratio
200%
Base Detail: moon.tif
Detail windows
Overview windows
Detail rectangles

Registering Multimodal MRI Images

This example shows how you can use imregister to automatically align two magnetic resonance images (MRI) to a common coordinate system using intensity-based image registration. Unlike some other techniques, it does not find features or use control points. Intensity-based registration is often well-suited for medical and remotely sensed imagery.

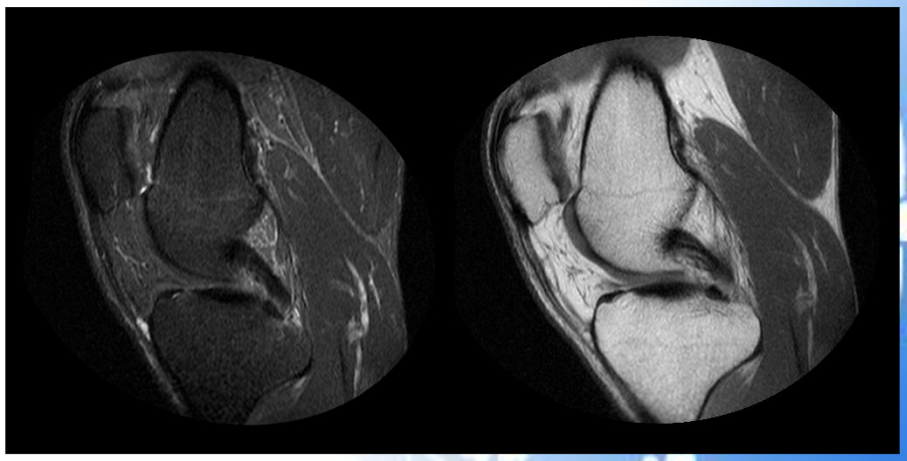
Step 1: Load Images

Example

```
fixed = dicomread('knee1.dcm');
moving = dicomread('knee2.dcm');
figure, imshowpair(moving, fixed,
'montage')
title('Unregistered')
```

Output

Unregistered



Cont..

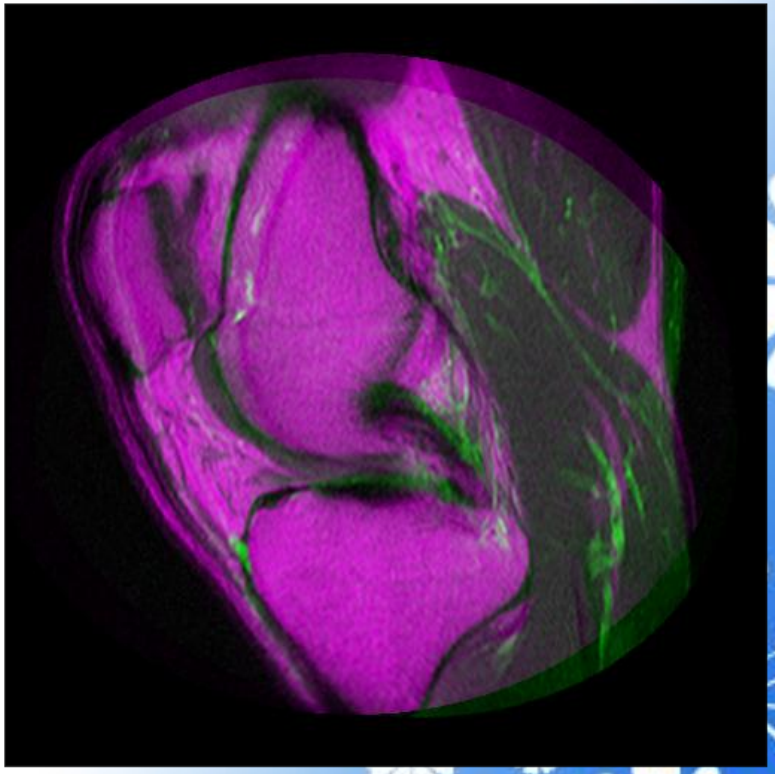
Example

figure, imshowpair(moving, fixed)

title('Unregistered')

Output

Unregistered



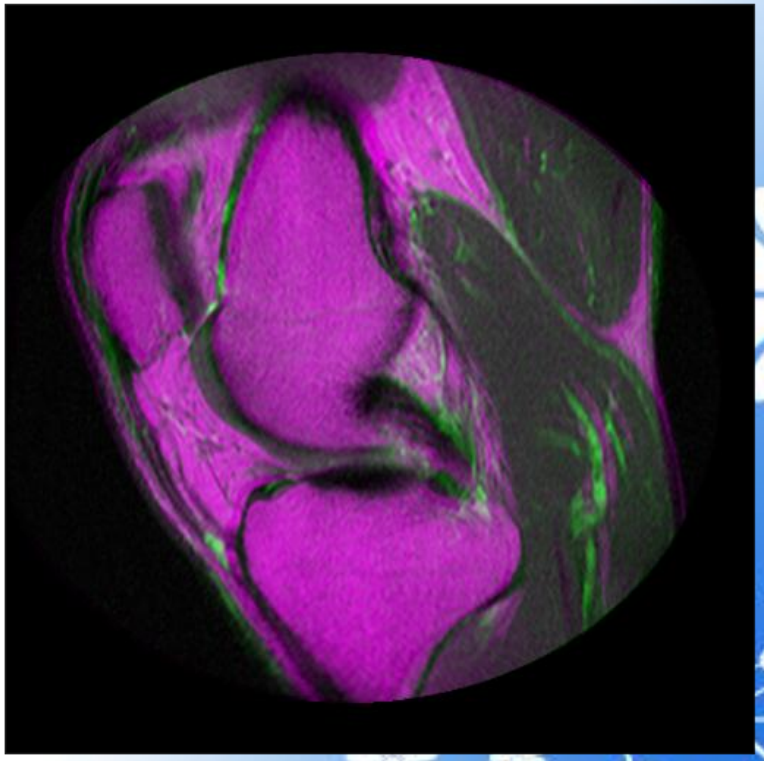
Step 2: Set up the Initial Registration

Example

```
[optimizer,metric] = imregconfig('multimodal');
tic
movingRegisteredDefault = imregister(moving, fixed, 'affine', optimizer, metric);
timeDefault = toc

figure, imshowpair(movingRegisteredDefault, fixed)
title('A: Default registration')
```

Output

A: Default registration



Step 3: Improve the Registration

Example

optimizer

metric

Output

```
optimizer =
  registration.optimizer.OnePlusOneEvolutionary

  Properties:
        GrowthFactor: 1.050000e+00
             Epsilon: 1.500000e-06
       InitialRadius: 6.250000e-03
    MaximumIterations: 100

metric =
  registration.metric.MattesMutualInformation

  Properties:
    NumberOfSpatialSamples: 500
     NumberOfHistogramBins: 50
             UseAllPixels: 1
```

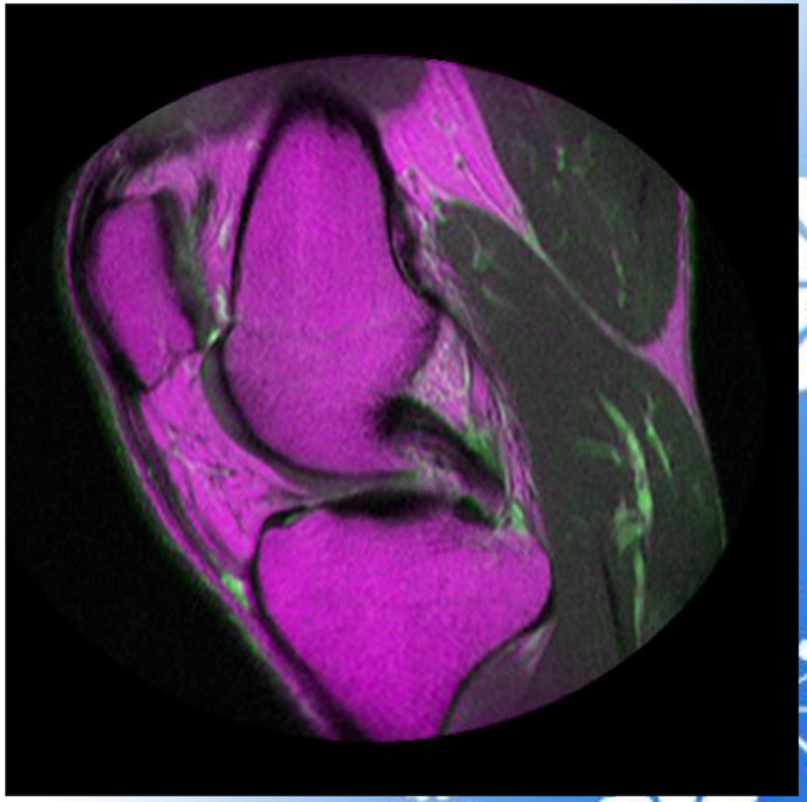
Cont..

Example

```
optimizer.MaximumIterations = 300;
movingRegistered =
imregister(moving, fixed, 'affine',
optimizer, metric);
figure,
imshowpair(movingRegistered,
fixed)
title('MaximumIterations = 300')
```

Output

MaximumIterations = 300



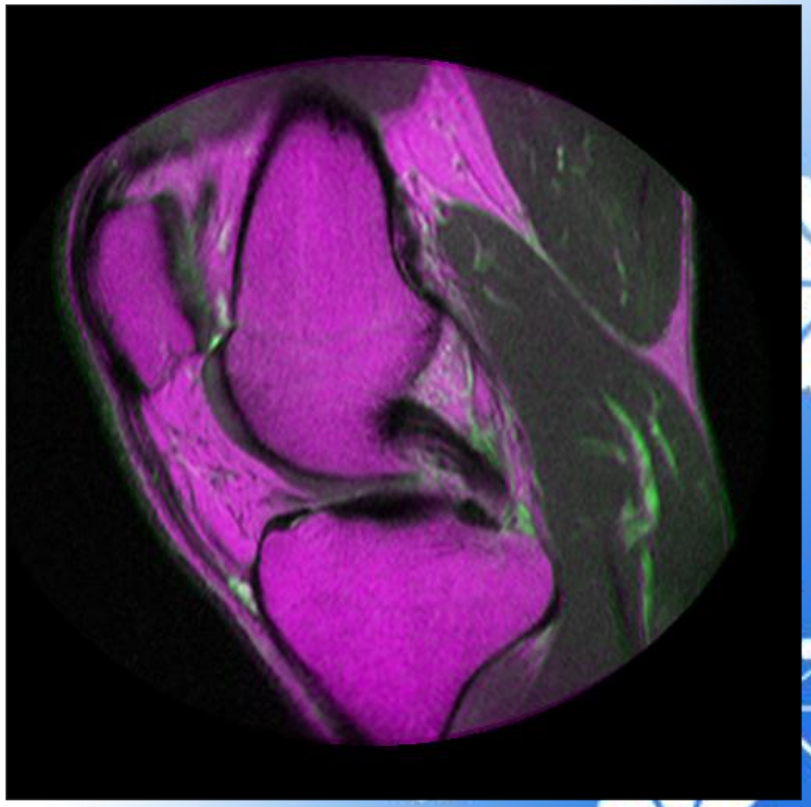
Cont..

Example

```
optimizer.MaximumIterations = 500;
tic
movingRegistered500 =
imregister(moving, fixed, 'affine',
optimizer, metric);
time500 = toc

figure,
imshowpair(movingRegistered500,
fixed)
title('B: MaximumIterations = 500')
```

Output

MaximumIterations = 500



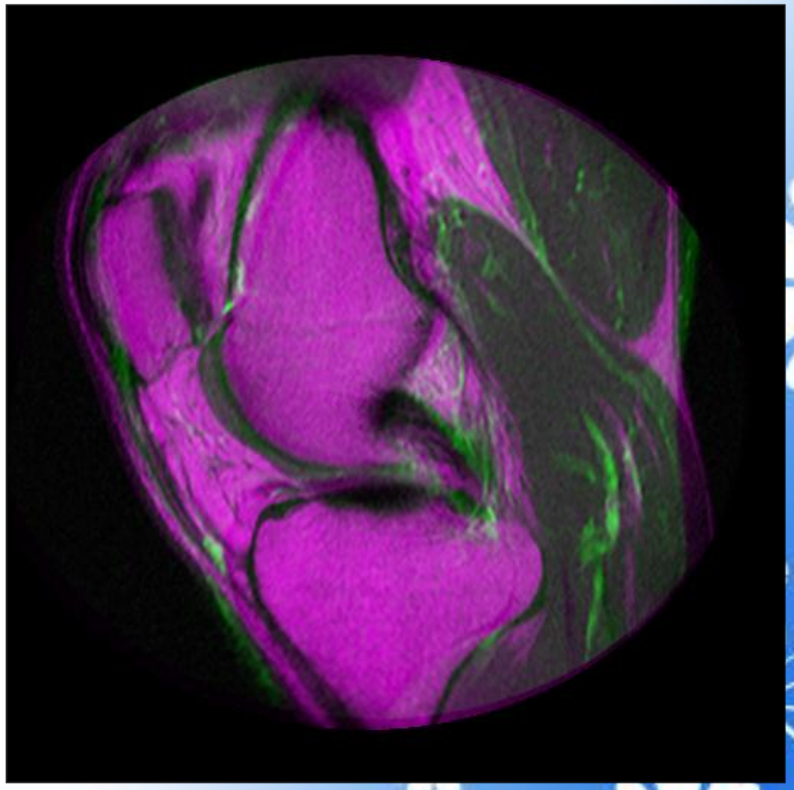
Step 4: Improve the Speed of Registration

Example

```
optimizer.MaximumIterations = 100;
optimizer.InitialRadius = 0.009;

movingRegistered = imregister(moving, fixed, 'affine', optimizer, metric);
figure, imshowpair(movingRegistered, fixed)
title('MaximumIterations = 100, InitialRadius = 0.009')
```

Output

MaximumIterations = 100, InitialRadius = 0.009



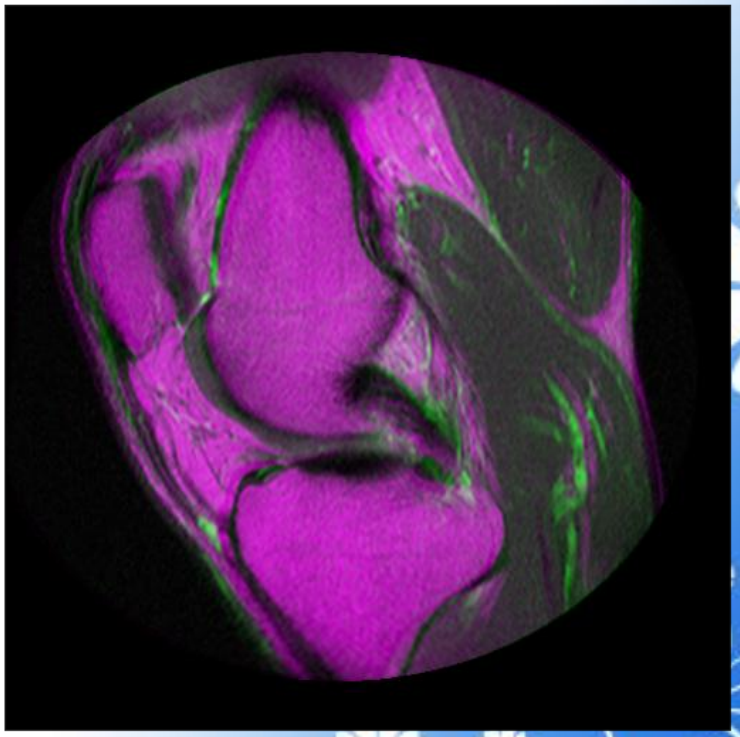
Cont..

Example

```
[optimizer,metric] = imregconfig('multimodal');
optimizer.Epsilon = 1.5e-4;

movingRegistered = imregister(moving, fixed, 'affine', optimizer, metric);
figure, imshowpair(movingRegistered, fixed)
title('MaximumIterations = 100, Epsilon = 1.5e-4')
```

Output

MaximumIterations = 100, Epsilon = 1.5e-4



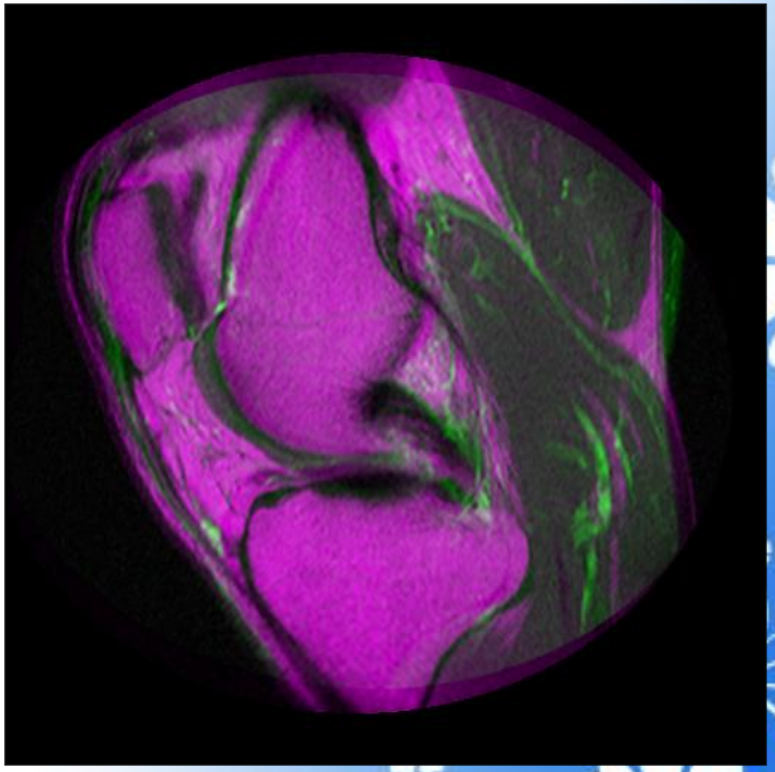
Cont..

Example

```
[optimizer,metric] = imregconfig('multimodal');
optimizer.GrowthFactor = 1.1;

movingRegistered = imregister(moving, fixed, 'affine', optimizer, metric);
figure, imshowpair(movingRegistered, fixed)
title('MaximumIterations = 100, GrowthFactor = 1.1')
```

Output

MaximumIterations = 100, GrowthFactor = 1.1



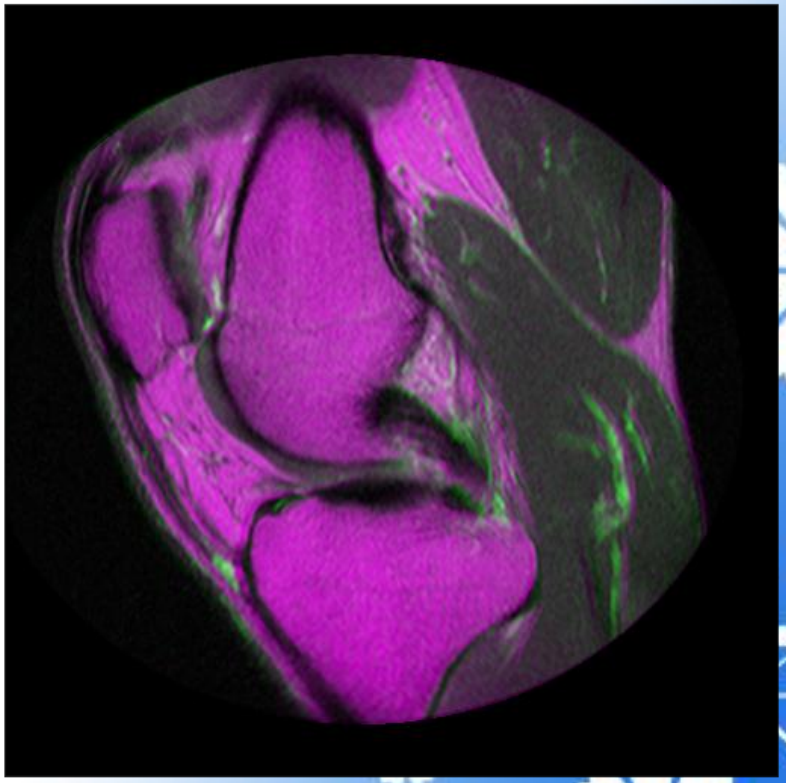
Cont..

Example

```
optimizer.GrowthFactor = 1.01;

tic
movingRegisteredGrowth = imregister(moving, fixed, 'affine', optimizer, metric);
timeGrowth = toc

figure, imshowpair(movingRegisteredGrowth, fixed)
title('C: MaximumIterations = 100, GrowthFactor = 1.01')
```

Output

C: MaximumIterations = 100, GrowthFactor = 1.01



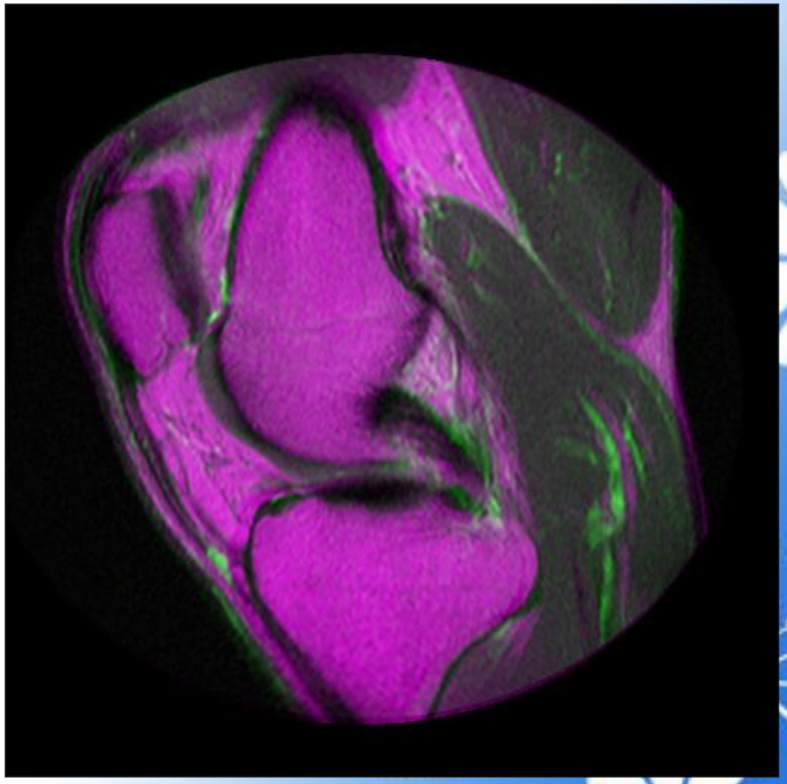
Step 5: Further Refinement

Example

```
[optimizer,metric] = imregconfig('multimodal');
optimizer.GrowthFactor = 1.01;
optimizer.InitialRadius = 0.009;
optimizer.Epsilon = 1.5e-4;
optimizer.MaximumIterations = 300;

tic
movingRegisteredTuned = imregister(moving, fixed, 'affine', optimizer, metric);
timeTuned = toc

figure, imshowpair(movingRegisteredTuned, fixed)
title('D: MaximumIterations = 300, GrowthFactor = 1.01, Epsilon = 1.5e-4, InitialRadius = 0.009')
```

Output

D: MaximumIterations = 300, GrowthFactor = 1.01, Epsilon = 1.5e-4, InitialRadius = 0.009

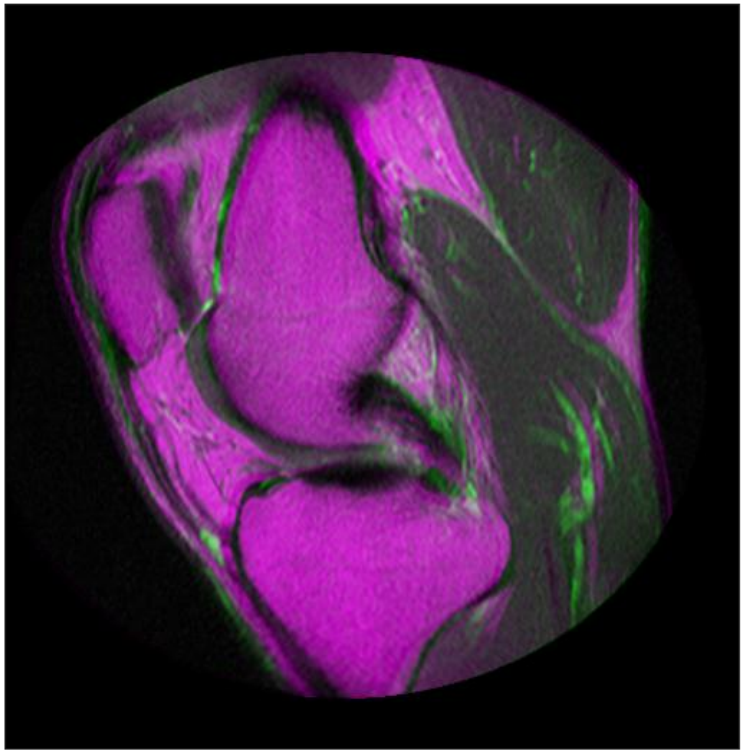


Step 6: Deciding When Enough is Enough

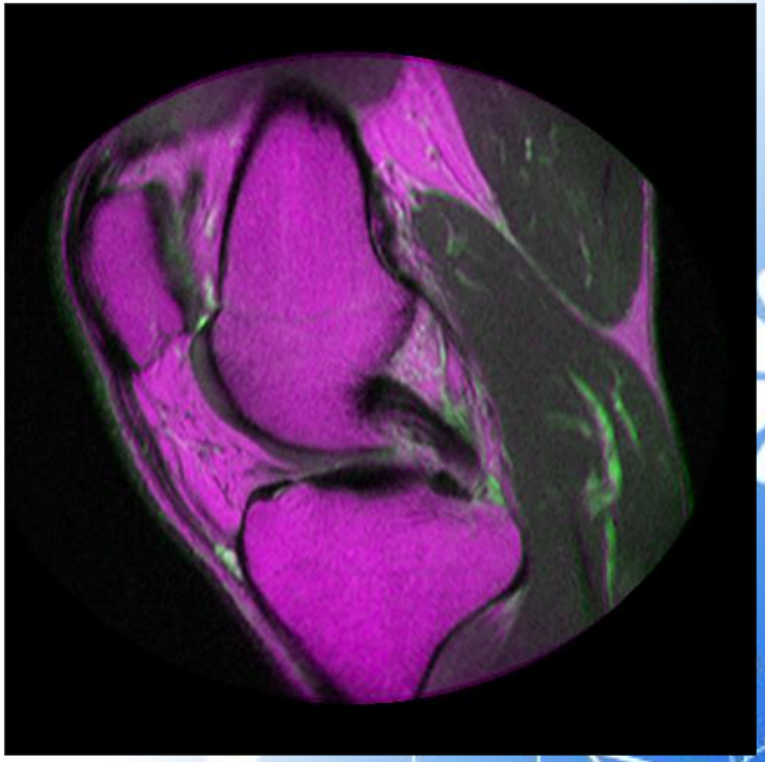
```
figure
imshowpair(movingRegisteredDefault, fixed)
title(sprintf('A - Default settings - %0.2f sec', timeDefault))

figure
imshowpair(movingRegistered500, fixed)
title(sprintf('B - Default settings, 500 iterations - %0.2f sec', time500))

figure
imshowpair(movingRegisteredGrowth, fixed)
title(sprintf('C - Growth factor, 100 iterations - %0.2f sec', timeGrowth))

figure
imshowpair(movingRegisteredTuned, fixed)
title(sprintf('D - Highly tuned, 300 iterations - %0.2f sec', timeTuned))
```

Cont..

A - Default settings - 13.35 sec

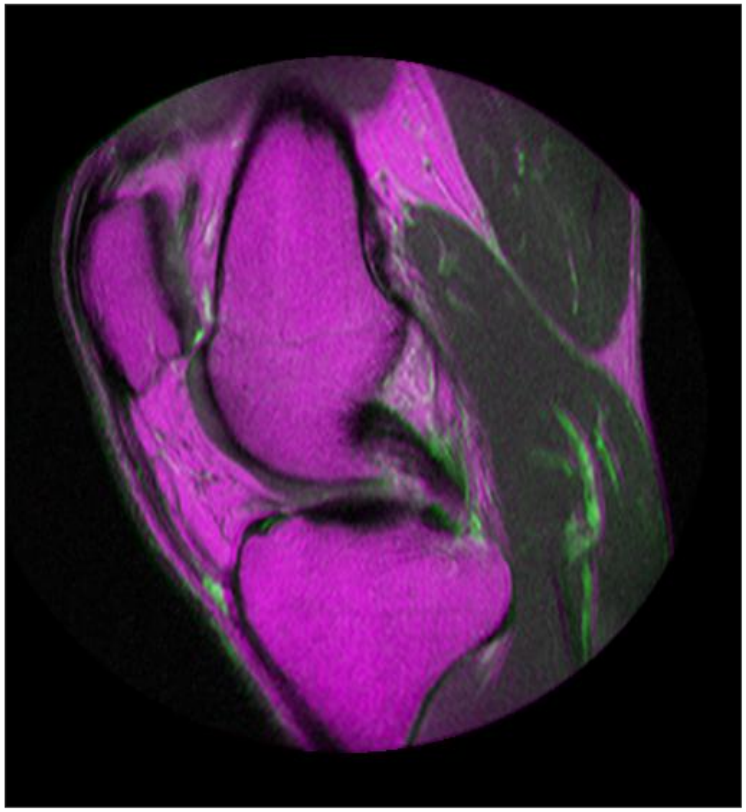


B - Default settings, 500 iterations - 65.53 sec

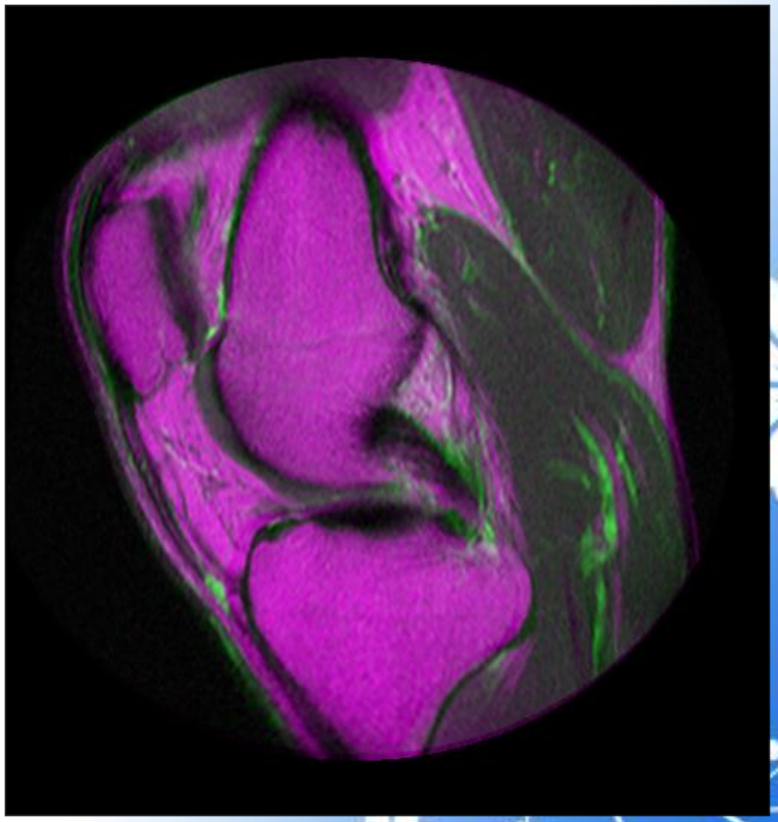


Cont..

C - Growth factor, 100 iterations - 12.14 sec



D - Highly tuned, 300 iterations - 36.02 sec



Step 7: Alternate Visualizations

Often as the quality of multimodal registrations improve it becomes more difficult to judge the quality of registration visually. This is because the intensity differences can obscure areas of misalignment. Sometimes switching to a different display mode for imshowpair exposes hidden details.

Designing and Implementing 2-D Linear Filters for Image Data

Performing Linear Filtering of Images Using imfilter

Example

```
I = imread('coins.png');

h = ones(5,5) / 25;

I2 = imfilter(I,h);

imshow(I), title('Original Image');

figure, imshow(I2), title('Filtered Image')
```

Output

Original Image

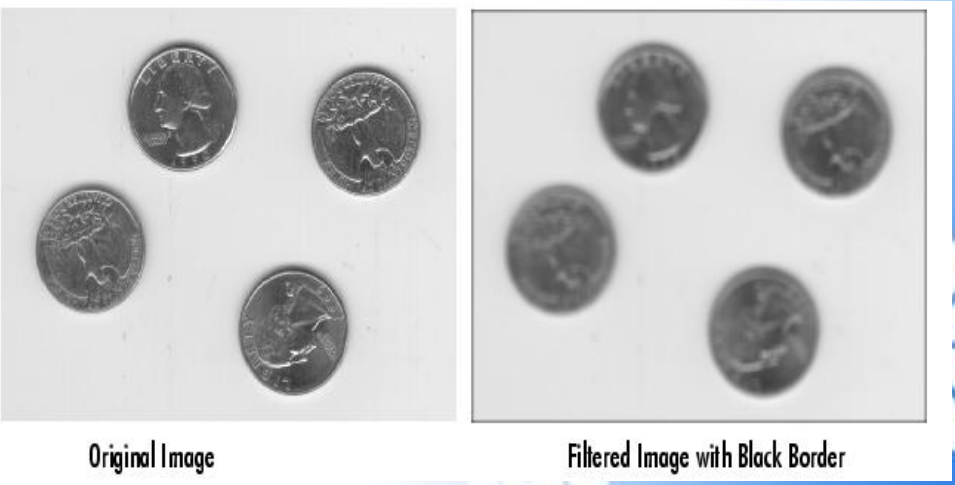
Filtered Image

Boundary Padding Options

Example

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

h = ones(5,5) / 25;

I2 = imfilter(I,h);

imshow(I), title('Original Image');

figure, imshow(I2), title('Filtered
Image with Black Border')
```

Output



Multidimensional Filtering

Example

```
rgb = imread('peppers.png');

imshow(rgb);
```

Output

Cont..

Example

```
h = ones(5,5)/25;

rgb2 = imfilter(rgb,h);

figure, imshow(rgb2)
```

Output

Filtering an Image with Predefined Filter Types

Example

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

Output

Transforms

Locating Image Features

Example

1. Read in the sample image.

bw = imread('text.png');

2. Create a template for matching by extracting the letter "a" from the image.

a = bw(32:45,88:98);

imshow(bw);

figure, imshow(a);

Output

Image (left) and the Template to Correlate (right)

Discrete Cosine Transform (DCT) and Image Compression

```
I = imread('cameraman.tif');
I = im2double(I);
T = dctmtx(8);
dct = @(block_struct) T * block_struct.data * T';
B = blockproc(I,[8 8],dct);
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        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   0   0   0   0   0   0];
B2 = blockproc(B,[8 8] , @(block_struct)  mask .* block_struct.data);
invdct = @(block_struct) T' * block_struct.data * T;
I2 = blockproc(B2,[8 8],invdct);
imshow(I), figure, imshow(I2)
```

Cont..

Reconstructing an Image from Parallel Projection Data

The commands below illustrate how to reconstruct an image from parallel projection data. The test image is the Shepp-Logan head phantom, which can be generated using the phantom function. The phantom image illustrates many of the qualities that are found in real-world tomographic imaging of human heads. The bright elliptical shell along the exterior is analogous to a skull, and the many ellipses inside are analogous to brain features.

Cont..

Example

1. Create a Shepp-Logan head phantom image.

P = phantom(256);

imshow(P)

Output

Cont..

2. Compute the Radon transform of the phantom brain for three different sets of theta values. R1 has 18 projections, R2 has 36 projections, and R3 has 90 projections.

theta1 = 0:10:170; [R1,xp] = radon(P,theta1);

theta2 = 0:5:175; [R2,xp] = radon(P,theta2);

theta3 = 0:2:178; [R3,xp] = radon(P,theta3);

Cont..

Example

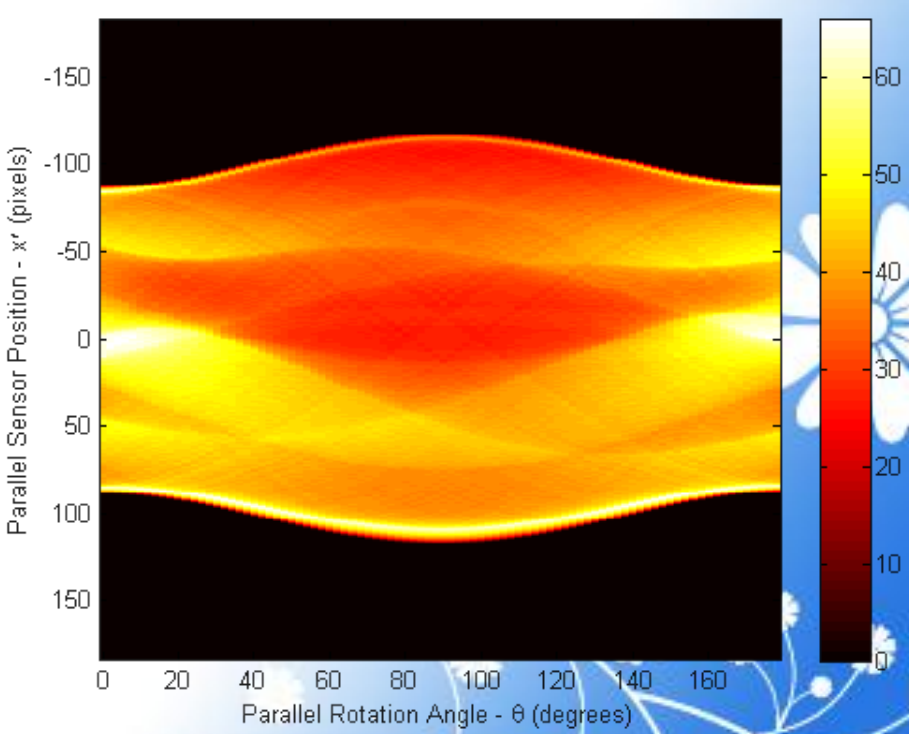
3. Display a plot of one of the Radon transforms of the Shepp-Logan head phantom. The following figure shows R3, the transform with 90 projections

figure,imagesc(theta3,xp,R3);

colormap(hot); colorbar

xlabel('\theta'); ylabel('x\prime');

Output



Cont..

Example

4. Reconstruct the head phantom image from the projection data created in step 2 and display the results.

```
I1 = iradon(R1,10);
I2 = iradon(R2,5);
I3 = iradon(R3,2);
imshow(I1)
figure, imshow(I2)
figure, imshow(I3)
```

Output

I1

Cont..

I2

I3

Reconstructing a Head Phantom Image

Example

1. Generate the test image and display it.

P = phantom(256);

imshow(P)

Output

Cont..

2. Compute fan-beam projection data of the test image, using the FanSensorSpacing parameter to vary the sensor spacing.

```
D = 250;
dsensor1 = 2;
F1 = fanbeam(P,D,'FanSensorSpacing',dsensor1);
dsensor2 = 1;
F2 = fanbeam(P,D,'FanSensorSpacing',dsensor2);
dsensor3 = 0.25
[F3, sensor_pos3, fan_rot_angles3] = fanbeam(P,D,...
'FanSensorSpacing',dsensor3);
```

Cont..

Example

3.Plot the projection data F3.

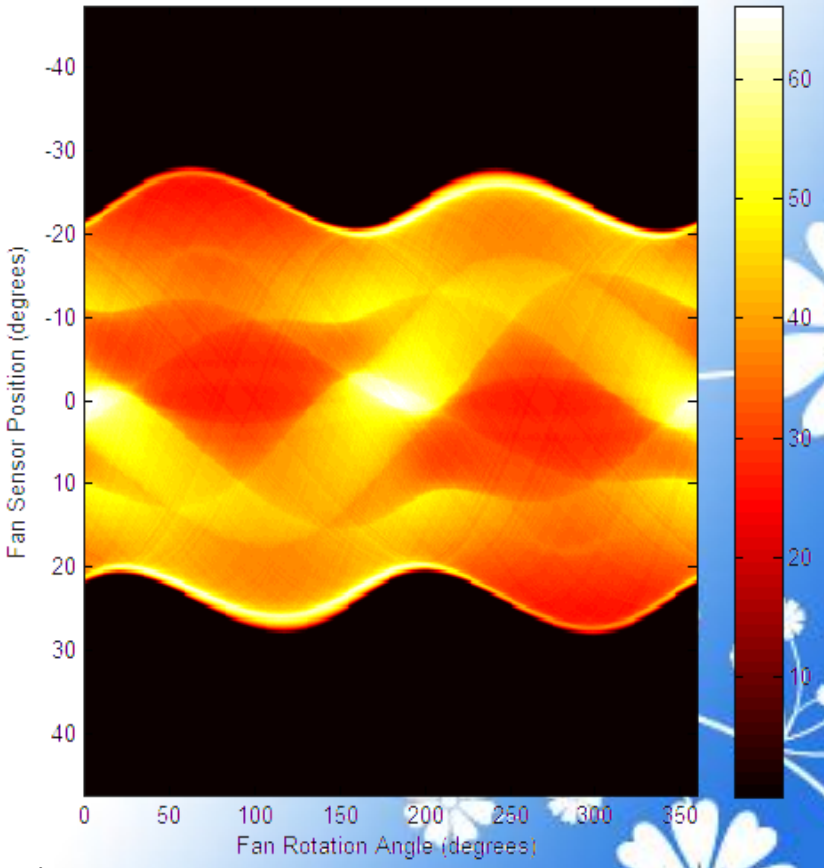
figure, imagesc(fan_rot_angles3, sensor_pos3, F3)

colormap(hot); colorbar

xlabel('Fan Rotation Angle (degrees)')

ylabel('Fan Sensor Position (degrees)')

Output



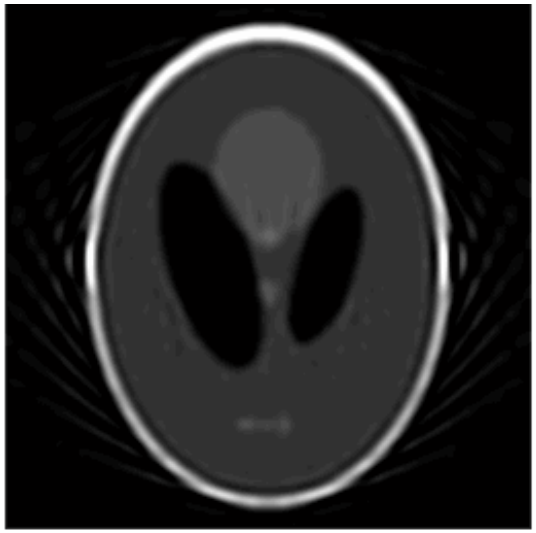
Cont..

4. Reconstruct the image from the fan-beam projection data using ifanbeam.

```
output_size = max(size(P));

Ifan1 = ifanbeam(F1,D, ...
    'FanSensorSpacing',dsensor1,'OutputSize',output_size);
figure, imshow(Ifan1)

Ifan2 = ifanbeam(F2,D, ...
    'FanSensorSpacing',dsensor2,'OutputSize',output_size);
figure, imshow(Ifan2)

Ifan3 = ifanbeam(F3,D, ...
    'FanSensorSpacing',dsensor3,'OutputSize',output_size);
figure, imshow(Ifan3)
```

Cont..

Ifan1



Ifan2

Ifan3

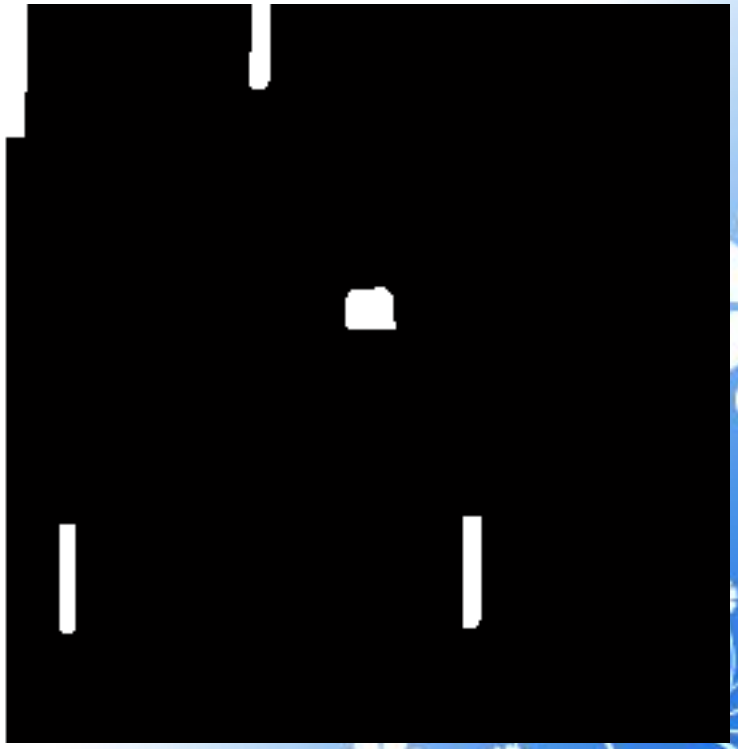
Morphological Operations

Morphological Opening

Example

```
BW1 = imread('circbw.tif')

SE = strel('rectangle',[40 30]);

BW2 = imerode(BW1,SE);

imshow(BW2)
```

Output



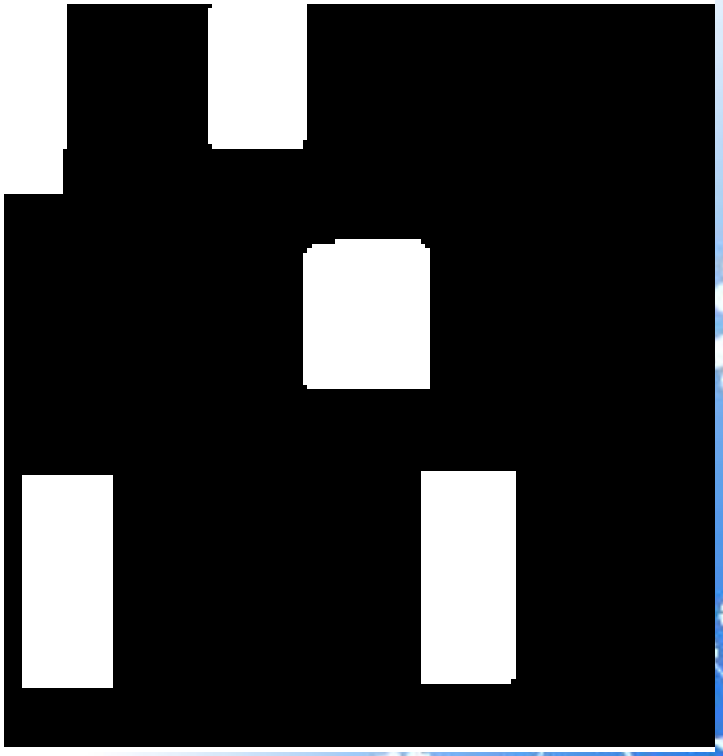
Cont..

Example

BW3 = imdilate (BW2,SE);

imshow(BW3)

Output

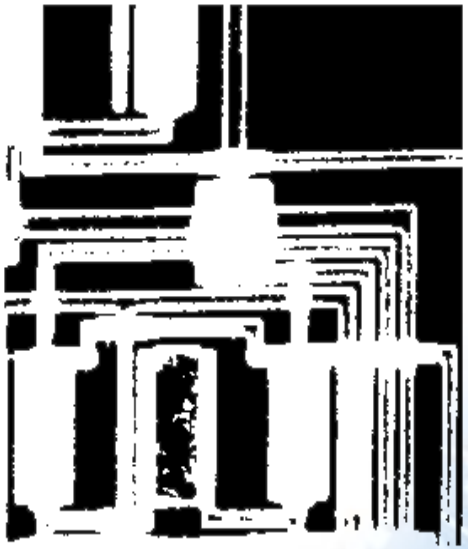


Skeletonization

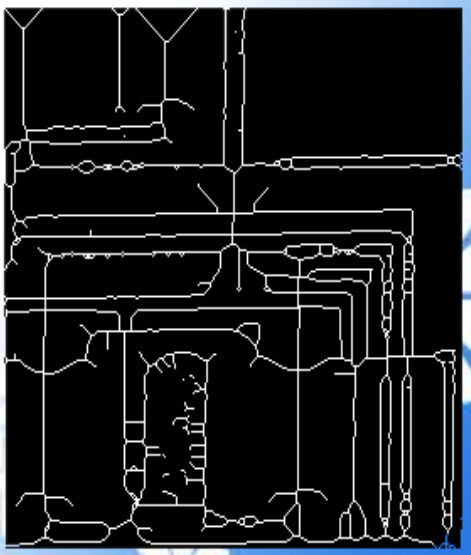
Example

```
BW1 = imread('circbw.tif');

BW2 = bwmorph(BW1,'skel',Inf);

imshow(BW1)

figure, imshow(BW2)
```

Output



Original Image



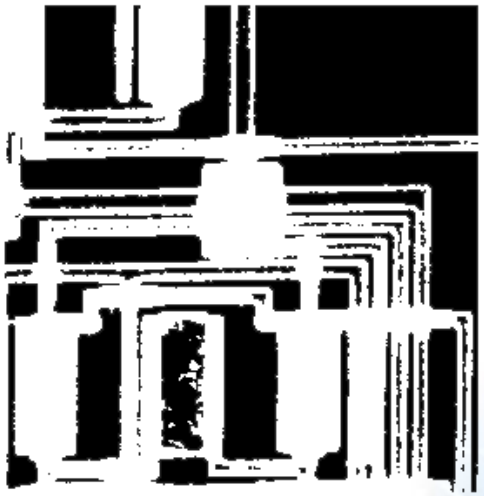
Skeletonization of Image

Perimeter Determination

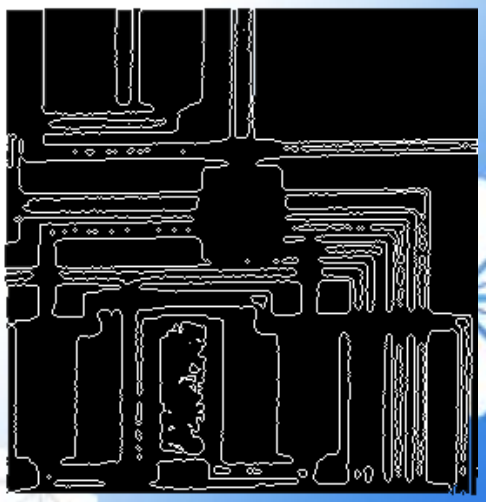
Example

```
BW1 = imread('circbw.tif');

BW2 = bwperim(BW1);

imshow(BW1)

figure, imshow(BW2)
```

Output



Original Image



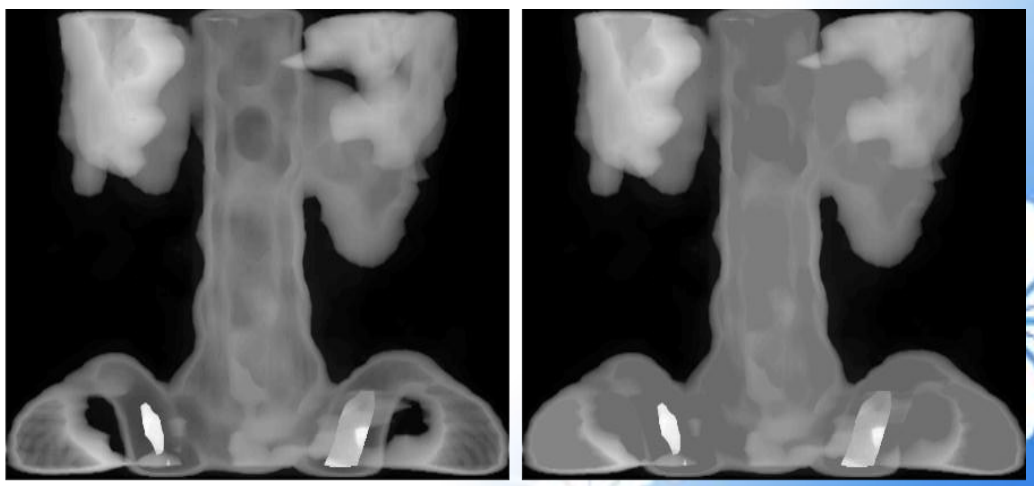
Perimeters Determined

Filling Holes

Example

```
[X,map] = imread('spine.tif');

I = ind2gray(X,map);

Ifill = imfill(I,'holes');

imshow(I);figure, imshow(Ifill)
```

Output



Original

After Filling Holes

Lookup Table

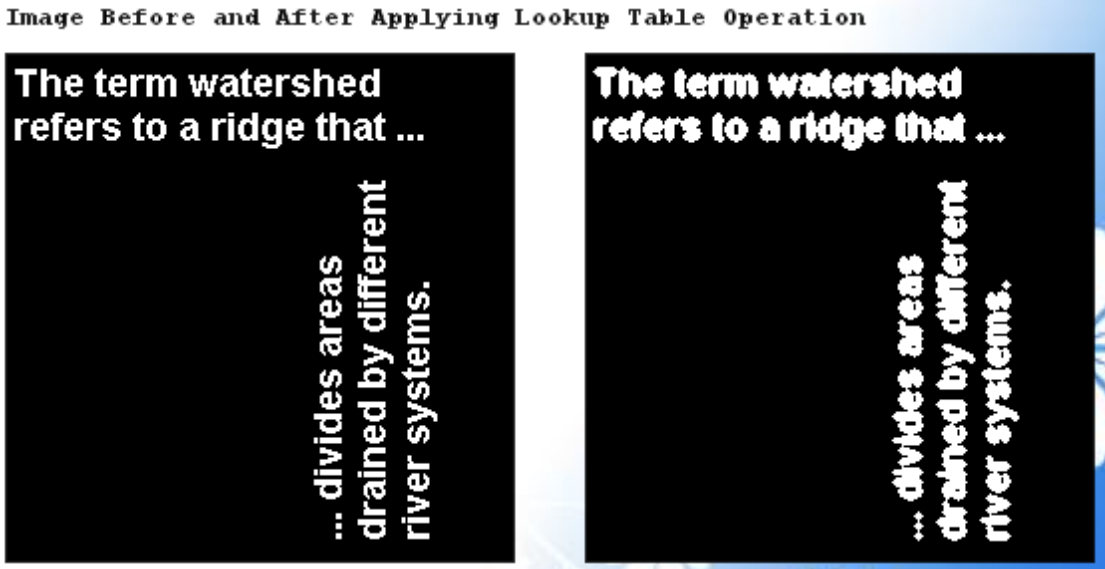
Once you create a lookup table, you can use it to perform the desired operation by using the applylut function.

Cont..

Example

```
f = @(x) sum(x(:)) >= 3;

lut = makelut(f,3);

BW1 = imread('text.png');

BW2 = applylut(BW1,lut);

imshow(BW1)

figure, imshow(BW2)
```

Output



Analyzing and Enhancing Images

Displaying a Contour Plot of Image Data

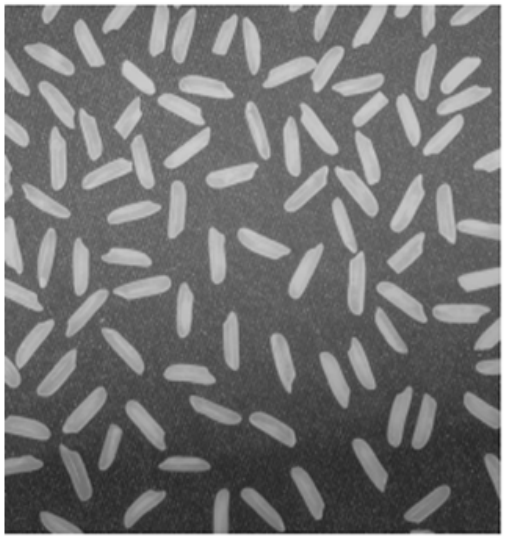
You can use the toolbox function imcontour to display a contour plot of the data in a grayscale image.

Cont..

Example

```
I = imread('rice.png');

imshow(I)

figure, imcontour(I,3)
```

Output



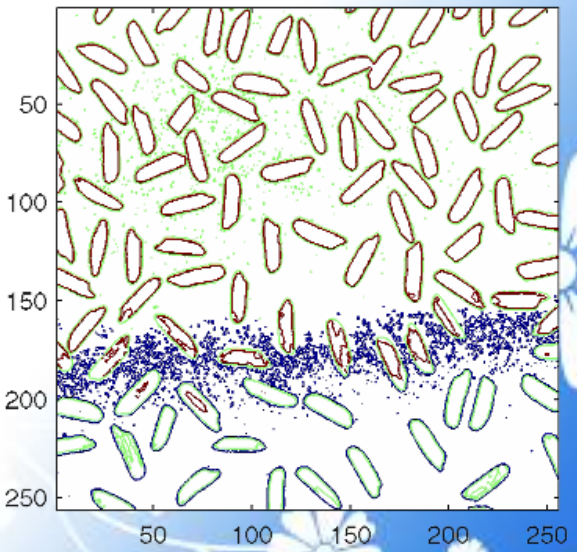
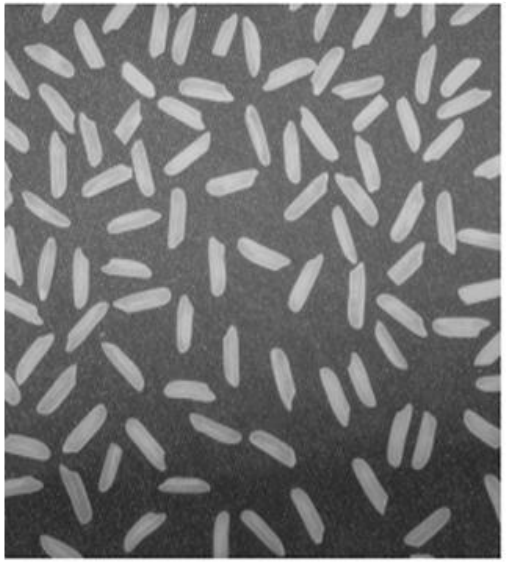


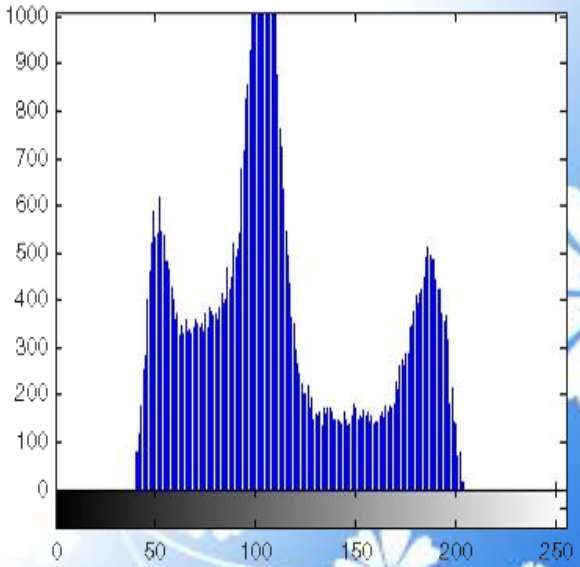
Image Histogram Using imhist

Example

```
I = imread('rice.png');

imshow(I)

figure, imhist(I)
```

Output





Detecting Edges Using the edge Function

In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image.

To find edges, you can use the edge function.

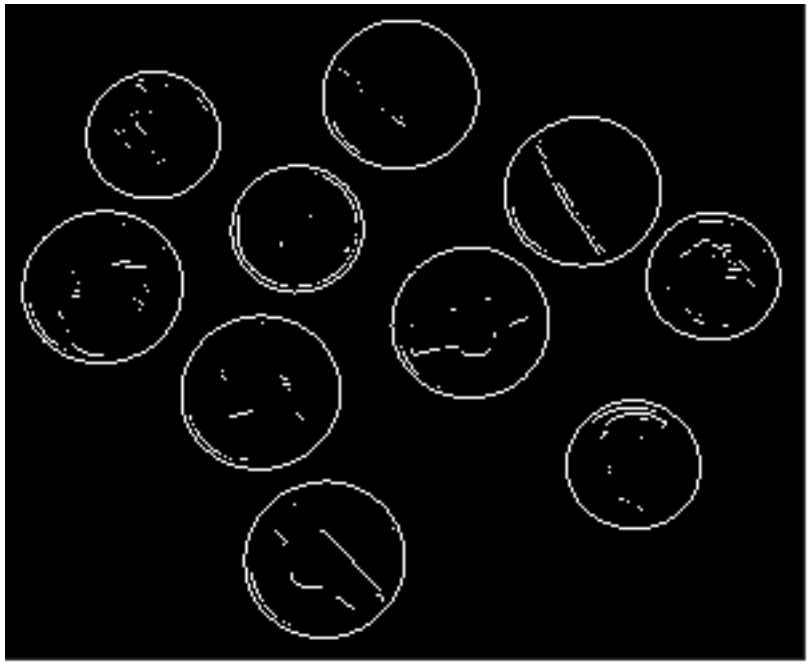
Cont..

Example

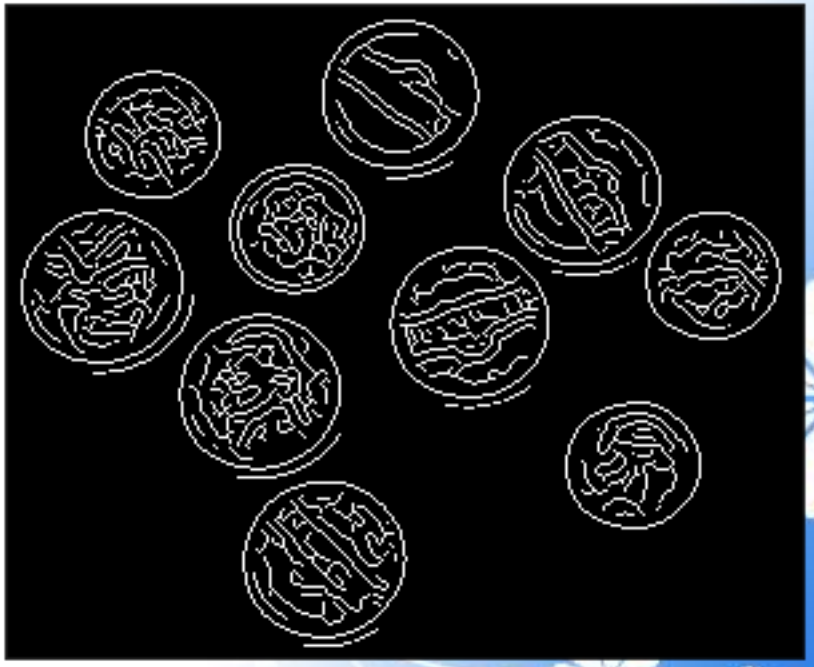
```
I = imread('coins.png');
imshow(I)
BW1 = edge(I,'sobel');
BW2 = edge(I,'canny');
imshow(BW1)
figure, imshow(BW2)
```

Output

Cont..



Sobel Filter



Canny Filter

Tracing Object Boundaries in an Image

Example

1. Read image and display it.

I = imread('coins.png');

imshow(I)

Output

Cont..

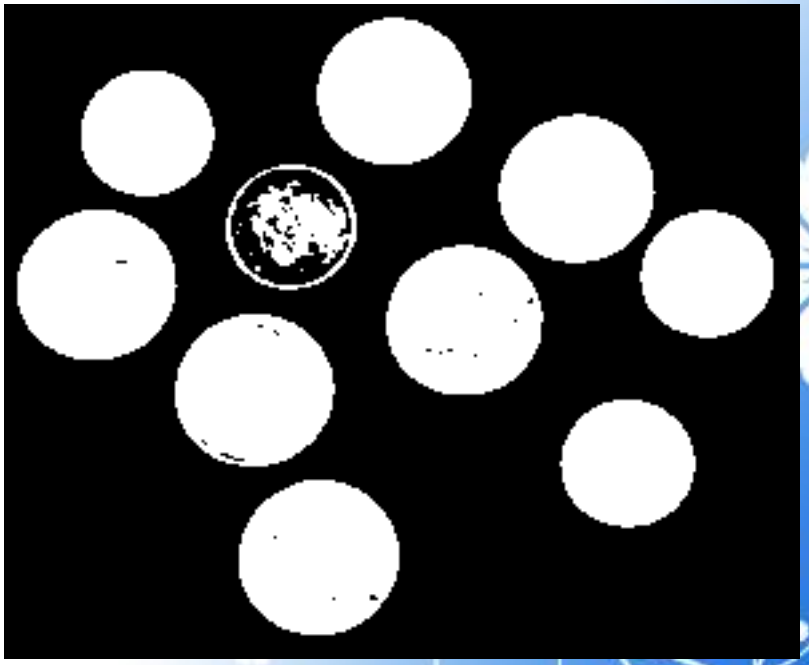
Example

2. Convert the image to a binary image.

BW = im2bw(I);

imshow(BW)

Output



Cont..

3. Determine the row and column coordinates of a pixel on the border of the object you want to trace.

Example

```
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)))
```

Cont..

4. Call bwtraceboundary to trace the boundary from the specified point.

Example

boundary = bwtraceboundary(BW,[row, col] , 'N');

Cont..

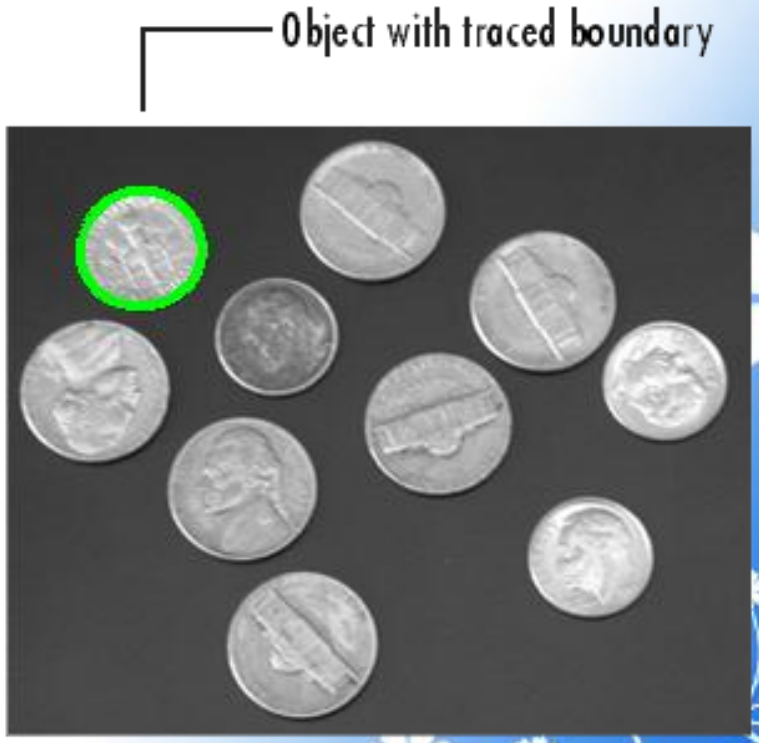
5. Display the original grayscale image and use the coordinates returned by bwtraceboundary to plot the border on the image.

Cont..

Example

```
imshow(I)
hold on;
plot(boundary(:,2),boundary(:,1),'g
','LineWidth',3);
```

Output



Cont..

6. To trace the boundaries of all the coins in the image, use the bwboundaries function.

Example

```
BW_filled = imfill(BW,'holes');
boundaries =
bwboundaries(BW_filled);
```

Cont..

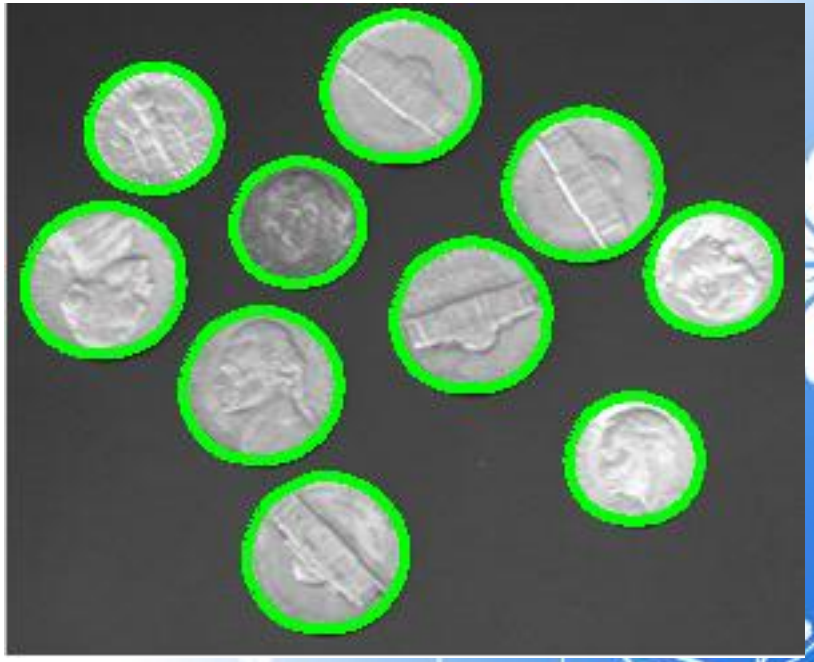
7. Plot the borders of all the coins on the original grayscale image using the coordinates returned by bwboundaries.

Cont..

Example

```
for k=1:10
   b = boundaries{k};
plot(b(:,2),b(:,1),'g','LineWidth',3);
end
```

Output



Texture Functions

Example

1. Read in the image and display it.

I = imread('eight.tif');

imshow(I)

Output

Cont..

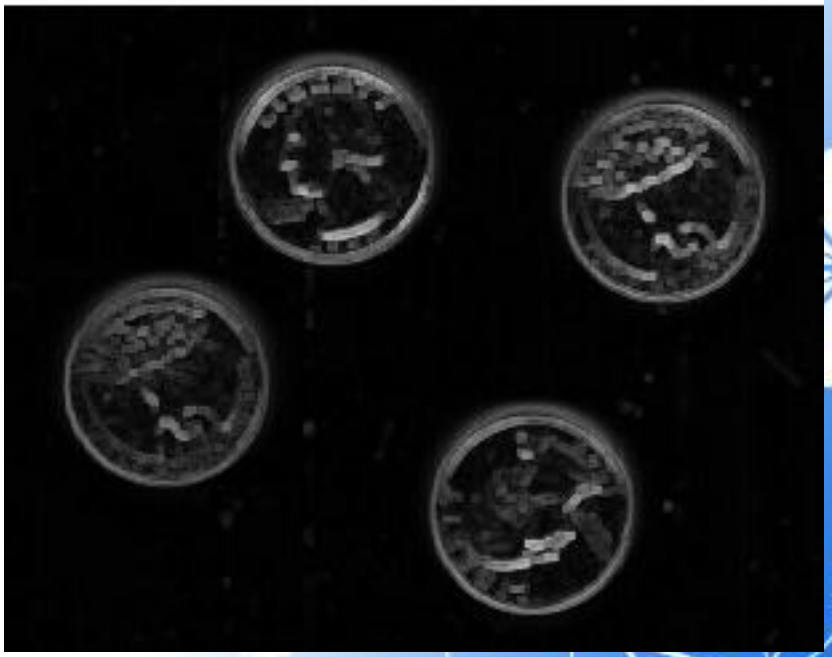
Example

2. Filter the image with the rangefilt function and display the results.

K = rangefilt(I);

figure, imshow(K)

Output



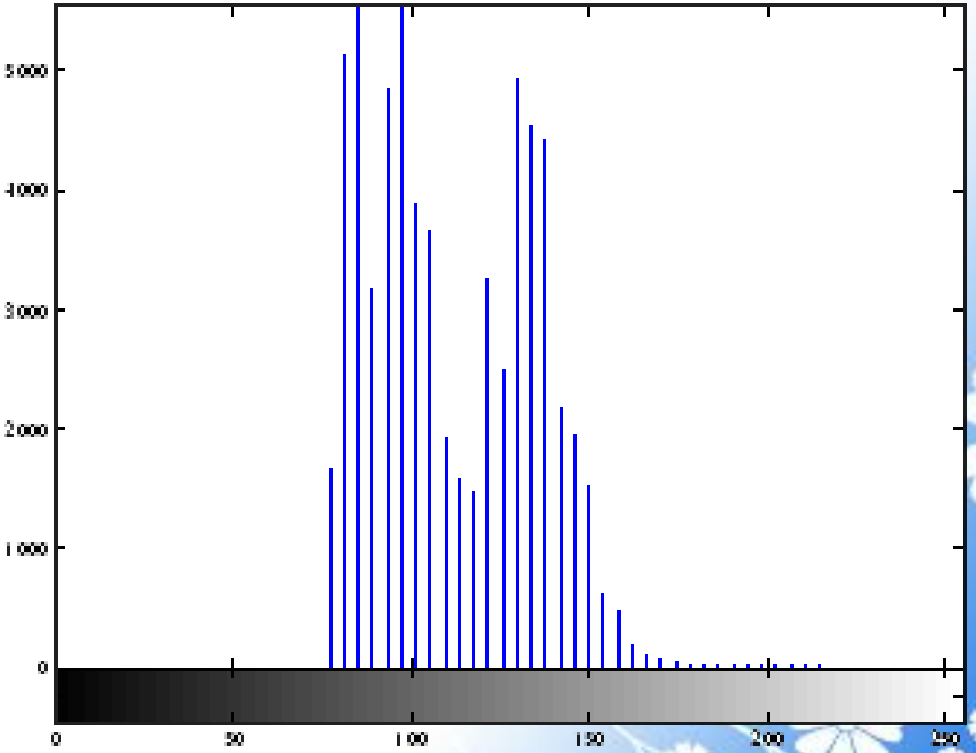
Understanding Intensity Adjustment

Intensity adjustment is an image enhancement technique that maps an image's intensity values to a new range.

Example

```
I = imread('pout.tif');
imshow(I)
figure, imhist(I,64)
```

Cont..

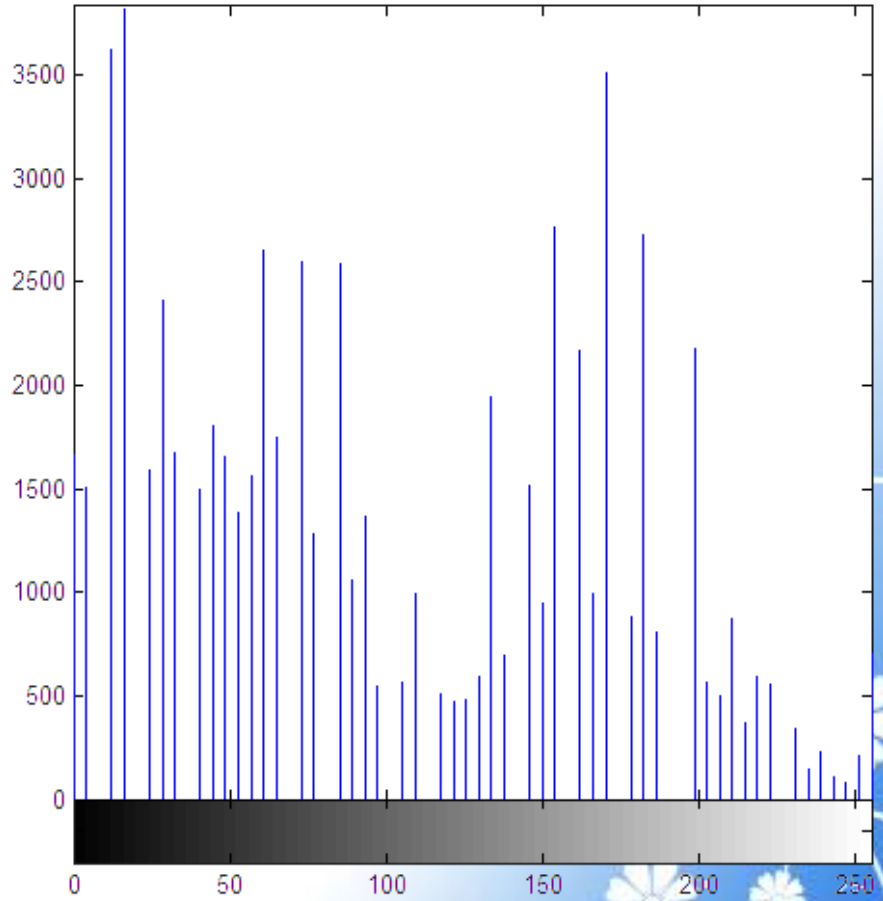


Adjusting Intensity Values to a Specified Range

You can adjust the intensity values in an image using the imadjust function, where you specify the range of intensity values in the output image.

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

J = imadjust(I);

imshow(J)

figure, imhist(J,64)
```

Cont..



Removing Noise By Median Filtering

Example

1. Read in the image and display it.

I = imread('eight.tif');

imshow(I)

Output

Cont..

Example

2. Add noise to it.

J = imnoise(I,'salt & pepper',0.02);

figure, imshow(J)

Output

Cont..

Example

3. Filter the noisy image with an averaging filter and display the results.

K = filter2 (fspecial ('average',3), J)/255;

figure, imshow(K)

Output

Cont..

Example

4. Now use a median filter to filter the noisy image and display the results.

L = medfilt2(J,[3 3]);

figure, imshow(L)

output

ROI-Based Processing

This chapter describes how to define a region of interest (ROI) and perform processing on the ROI

Creating a Binary Mask

Example

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

h_im = imshow(img);

e = imellipse(gca,[55 10 120 120]);

BW = createMask(e,h_im);
```

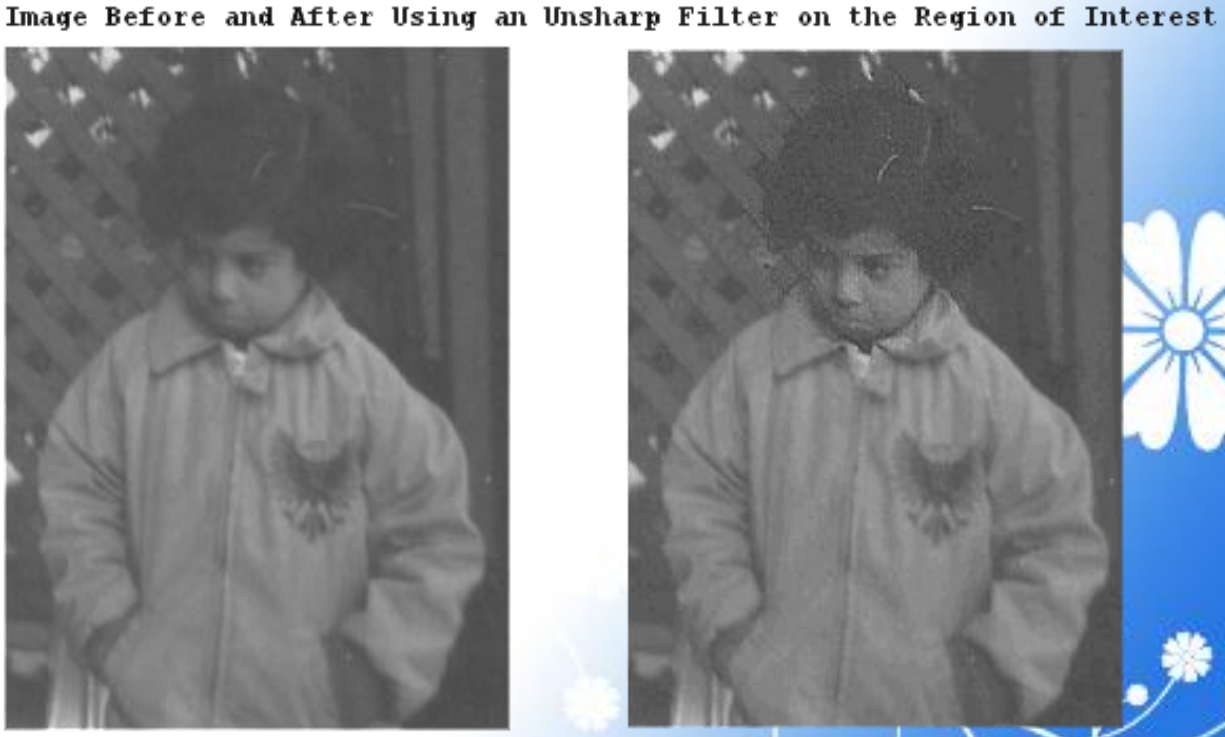
Output

Filtering a Region in an Image

Example

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

h = fspecial('unsharp');

I2 = roifilt2(h,I,BW);

imshow(I)

figure, imshow(I2)
```

Output



Filling an ROI

Filling is a process that fills a region of interest (ROI) by interpolating the pixel values from the borders of the region. This process can be used to make objects in an image seem to disappear as they are replaced with values that blend in with the background area.

Cont..

Example

1. Read an image into the MATLAB workspace and display it.

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

2.Call roifill to specify the ROI you want to fill.

I2 = roifill;

Output

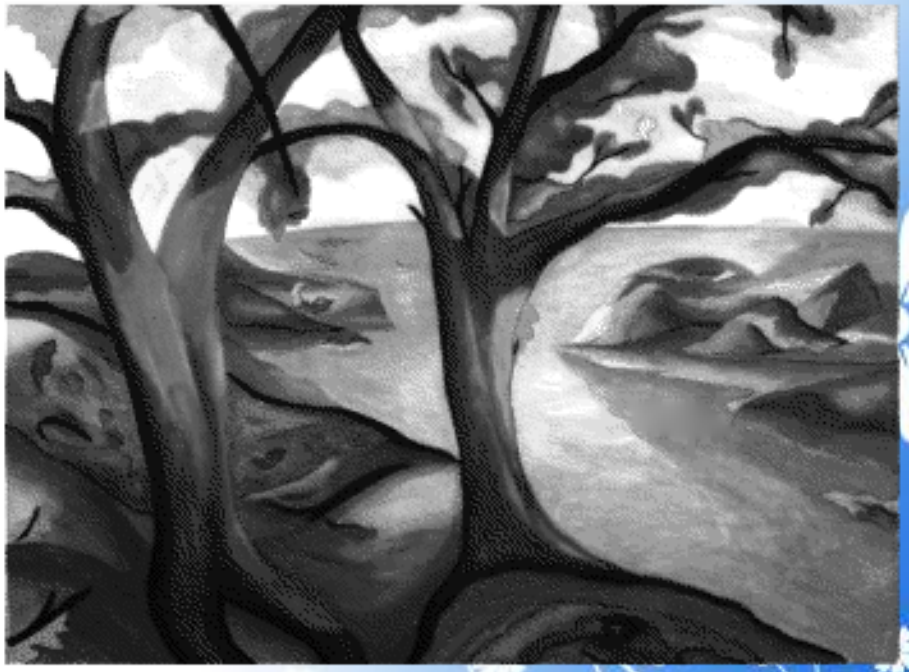
Cont..

Example

3. Perform the fill operation. Double-click inside the ROI or right-click and select Fill Area.

imshow(I2)

Output



Example: Using the deconvlucy Function to Deblur an Image

Example

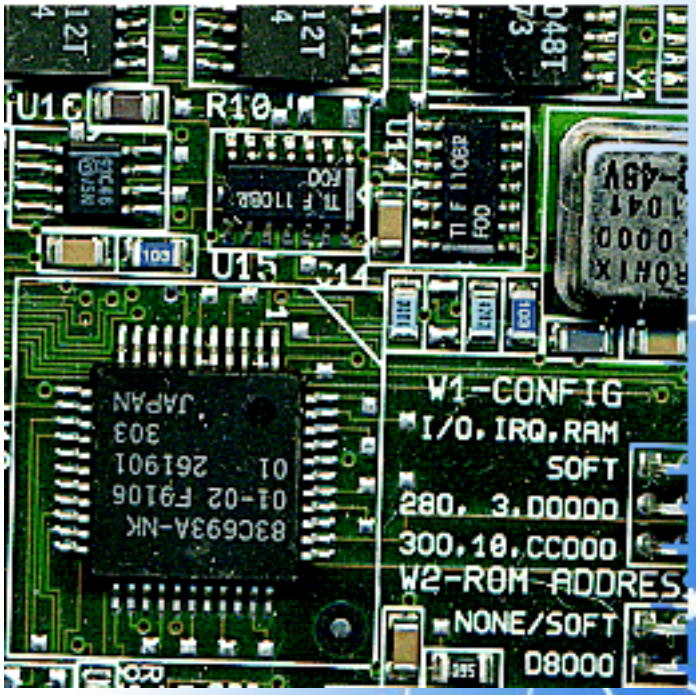
1. Read an image into the MATLAB workspace.

I = imread('board.tif');

I = I(50+[1:256],2+[1:256],:);

figure, imshow(I)

Output



Cont..

Example

2. Create the PSF.

PSF = fspecial('gaussian',5,5);

3. Create a simulated blur in the image and add noise.

Blurred = imfilter(I,PSF,'symmetric','conv');

V = .002;

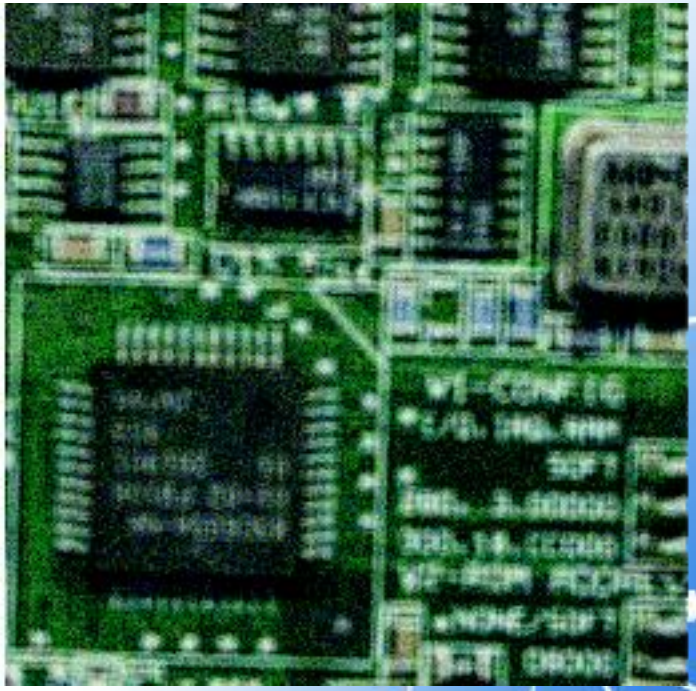
BlurredNoisy = imnoise(Blurred,'gaussian',0,V);

figure, imshow(BlurredNoisy)

title('Blurred and Noisy Image')

Output

Blurred and Noisy Image



Cont..

Example

4. Use deconvlucy to restore the blurred and noisy image

luc1 =

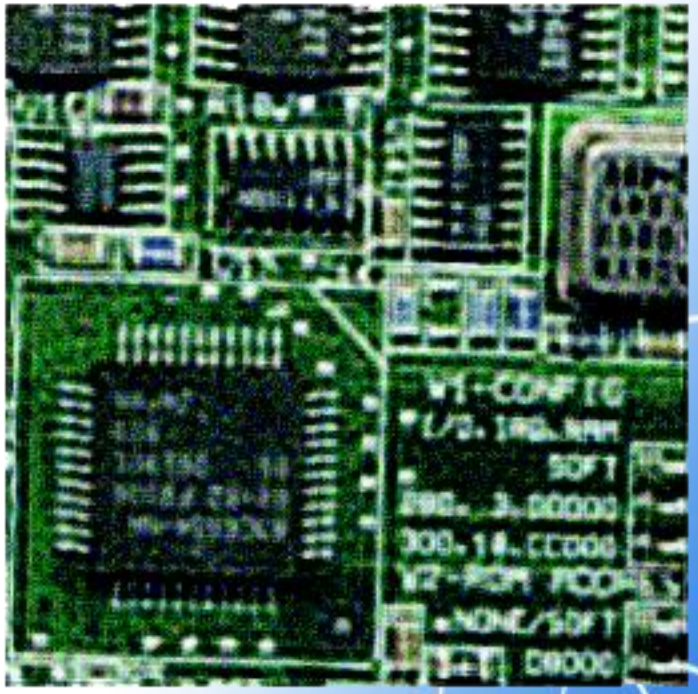
deconvlucy(BlurredNoisy,PSF,5);

figure, imshow(luc1)

title('Restored Image')

Output

Restored Image



Color

This chapter describes the toolbox functions that help you work with color image data. Note that "color" includes shades of gray; therefore much of the discussion in this chapter applies to grayscale images as well as color images.

Reducing Colors Using imapprox

Use imapprox when you need to reduce the number of colors in an indexed image. imapprox is based on rgb2ind and uses the same approximation methods. Essentially, imapprox first calls ind2rgb to convert the image to RGB format, and then calls rgb2ind to return a new indexed image with fewer colors.

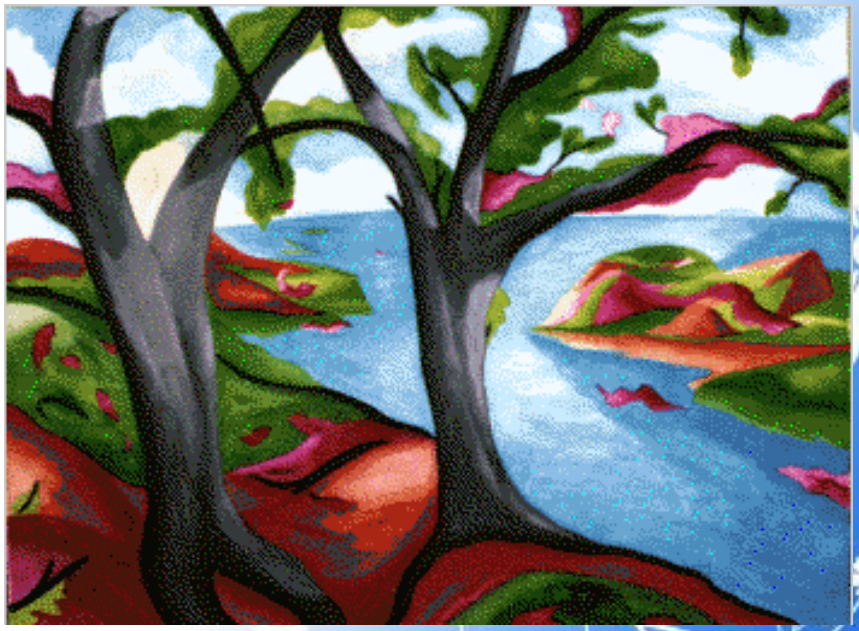
Example

```
load trees
[Y,newmap] = imapprox (X,map, 64);
imshow(Y, newmap);
```

Cont..

Original

After Operation



Dithering

When you use rgb2ind or imapprox to reduce the number of colors in an image, the resulting image might look inferior to the original, because some of the colors are lost. rgb2ind and imapprox both perform dithering to increase the apparent number of colors in the output image. Dithering changes the colors of pixels in a neighborhood so that the average color in each neighborhood approximates the original RGB color.

Cont..

Example

1. Read image and display it.

```
rgb=imread('onion.png');

imshow(rgb);
```

Output

Cont..

Example

2. Create an indexed image with eight colors and without dithering.

[X_no_dither,map]= rgb2ind(rgb,8,'nodither');

figure, imshow(X_no_dither,map);

Output

Cont..

Example

3. Create an indexed image using eight colors with dithering. Notice that the dithered image has a larger number of apparent colors but is somewhat fuzzy-looking.

[X_dither,map]=rgb2ind(rgb,8, 'dither');

figure, imshow(X_dither,map);

Output

Neighborhood and Block Operations

Implementing Linear and Nonlinear Filtering as Sliding Neighborhood Operations

For example, this code computes each output pixel by taking the standard deviation of the values of the input pixel's 3-by-3 neighborhood (that is, the pixel itself and its eight contiguous neighbors).

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

I2 = nlfilter(I,[3 3],'std2');
```

Cont..

This example converts the image to class double because the square root function is not defined for the uint8 datatype.

```
I = im2double(imread('tire.tif'));

f = @(x) sqrt(min(x(:)));

I2 = nlfilter(I,[2 2],f);
```

Cont..

The following example uses **nlfilter** to set each pixel to the maximum value in its 3-by-3 neighborhood.

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

f = @(x) max(x(:));

I2 = nlfilter(I,[3 3],f);

imshow(I);

figure, imshow(I2);
```

Cont..

Each Output Pixel Set to Maximum Input Neighborhood Value

Implementing Block Processing Using the blockproc Function

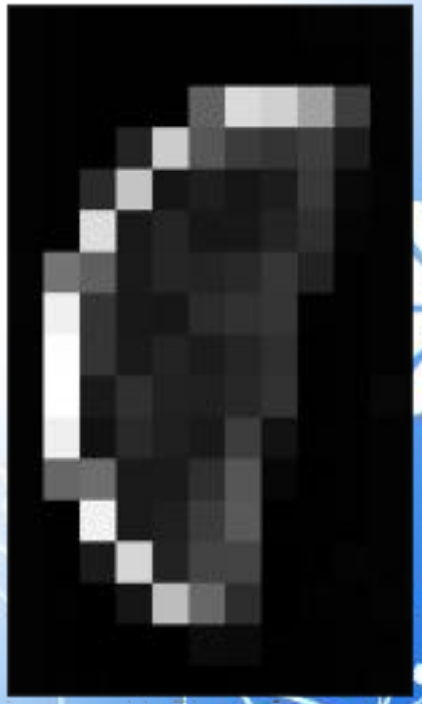
To perform distinct block operations, use the blockproc function. The blockproc function extracts each distinct block from an image and passes it to a function you specify for processing. The blockproc function assembles the returned blocks to create an output image.

Cont..

Example

```
myfun = @(block_struct) ...
   uint8(mean2(block_struct.data)*...
   ones(size(block_struct.data)));
I2 = blockproc('moon.tif',[32 32],
myfun);
figure;
imshow ('moon.tif');
figure;
imshow(I2,[]);
```

Output



Using Column Processing with Distinct Block Operations

For a distinct block operation, colfilt creates a temporary matrix by rearranging each block in the image into a column. colfilt pads the original image with 0's, if necessary, before creating the temporary matrix.

Cont..

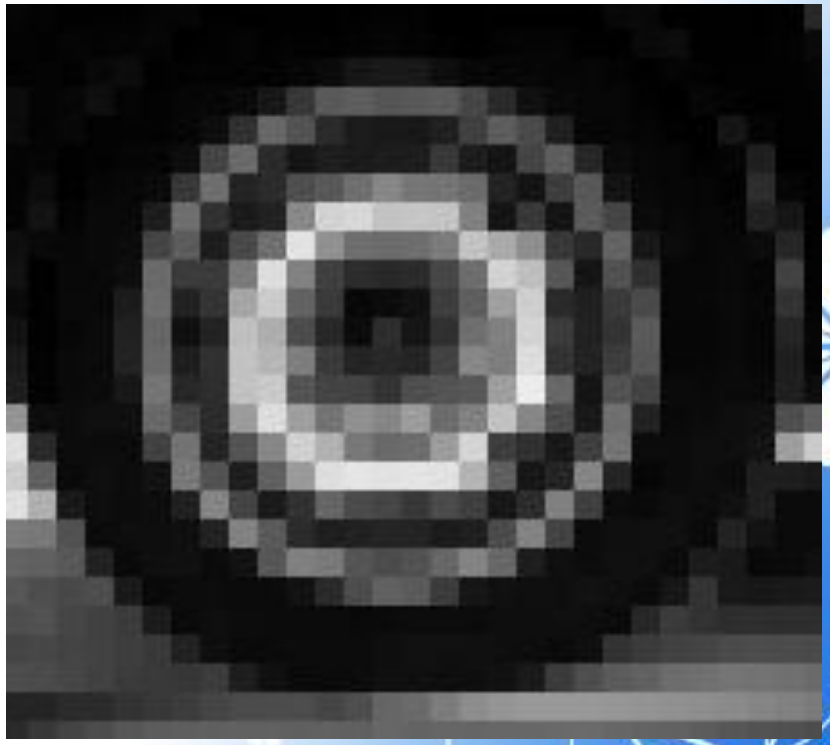
This example sets all the pixels in each 8-by-8 block of an image to the mean pixel value for the block.

```
I = im2double(imread('tire.tif'));

f = @(x) ones(64,1)*mean(x);

I2 = colfilt(I,[8 8],'distinct',f);
```

Cont..

Original

After Operation



Thank You