Getting Started

This section introduces you to the fundamentals of image processing with MATLAB. It describes the types of supported images, and how MATLAB represents them. It also explains the basics of working with color, coordinate systems, and file formats.

Understanding Images in MATLAB

MATLAB supports four basic types of images:

- Indexed images
- Intensity images
- Binary images
- RGB images

The basic data type in MATLAB is the rectangular matrix, an ordered set of real or complex elements. This object is naturally suited to the representation of images, real-valued, ordered sets of color or intensity data. (The toolbox does not support complex-valued images.)

In this guide the word pixel denotes a single element in an image matrix. You can select a single pixel from an image matrix using normal matrix subscripting. For example,

\[
I(2, 15)
\]

returns the value of the pixel at row 2 and column 15 of the image I. Pixel is derived from picture element and usually denotes a single dot on a computer display. By default, MATLAB scales images to fill the display axes; therefore, an image pixel may use more than a single pixel on the screen. (To assign a single screen pixel to each image pixel, use the truesize function described in the Reference section of this guide.)

Indexed Images

Typical color images require two matrices, a colormap and an image matrix. The colormap is an ordered set of values that represent the colors in the image. For each image pixel, the image matrix contains a corresponding index into the colormap. (The elements of the image matrix are floating-point integers, or flints, which MATLAB stores as double-precision values.)
The size of the colormap matrix is \( n \)-by-3 for an image containing \( n \) colors. Each row of the colormap matrix is a 1-by-3 red, green, blue (RGB) color vector

\[
color = [R \ G \ B]
\]

that specifies the intensity of the red, green, and blue components of that color. \( R, G, \) and \( B \) are real scalars that range from 0.0 (black) to 1.0 (full intensity). MATLAB translates these values into display intensities when you display an image and its colormap.

When MATLAB displays an indexed image, it uses the values in the image matrix to look up the desired color in the colormap. For instance, if the image matrix contains the value 18 in matrix location (86,198), then the color for pixel (86,198) is the color from row 18 of the colormap.
Outside MATLAB, indexed images with \( n \) colors often contain values from 0 to \( n-1 \). These values are indices into a colormap with 0 as its first index. Since MATLAB matrices start with index 1, you must increment each value in the image, or *shift up* the image, to create an image that you can manipulate with toolbox functions.

The statements

\[
X = [1 1 2 1 3 \\
    1 1 2 3 1 \\
    2 2 3 2 2 \\
    1 3 2 1 1];
\]

\[
map = [.4 .4 .4; 0 .6 1; 1 0 0];
\]

create a 4-by-5 image consisting of \( X \), a 3-color image matrix, and \( map \), the associated colormap. The image contains a light blue cross and red diagonal line on a gray background. To see this image, enter the matrices and use

\[
\text{imshow}(X, \text{map})
\]
Intensity Images

MATLAB stores an intensity image as a single matrix. The intensity matrix contains double precision values ranging from 0.0 to 1.0, with each element of the matrix corresponding to an image pixel. The elements in the intensity matrix represent various intensities, or gray levels, where the intensity 0.0 represents black and the intensity 1.0 represents full intensity, or white. Intensity images are ideal for black and white photographs, gray-scale camera images, and medical images.

```
load trees
I = ind2gray(X,map);
imshow(I,64)
```

Intensity Image Matrix

Create a simple intensity image using

\[ I = \frac{(0:15)}{15} \]

\[ I = \]

Columns 1 through 7
0 0.0667 0.1333 0.2000 0.2667 0.3333 0.4000

Columns 8 through 14
0.4667 0.5333 0.6000 0.6667 0.7333 0.8000 0.8667

Columns 15 through 16
0.9333 1.0000

This represents a gray scale stripe.
To display the image above with 64 discrete gray levels, use

```matlab
imshow(I,64)
axis on
```

You can also display intensity images with other ramp (linear) colormaps. See “Displaying Images” for more information.

Many systems outside MATLAB store intensity images as indexed images with an implied gray scale colormap, usually of length 256. You can also apply this scheme to images you use with MATLAB. The indexed version can also use a gray scale colormap of any length. When you store intensity images in this manner, you can work with them in the same way as any other indexed image.

If you store intensity images in indexed image format, or obtain images in this format from another application, you can easily convert them back to gray scale. For zero-based colormaps, use

```matlab
ind2gray(X+1,gray(256))
```

For colormaps beginning with index 1, use

```matlab
ind2gray(X,gray(256))
```

If the image comes with its own colormap, say `map`, use

```matlab
ind2gray(X+1,map) or ind2gray(X,map)
```

See “Changing Image Types,” later in this chapter, for more details on using `ind2gray`.
Binary Images

A binary (black and white) image is a special kind of intensity image. Binary images contain only two levels, 0 (black) and 1 (white). MATLAB stores these values in the image matrix as double-precision numbers.

load trees
I = ind2gray(X,map);
BW = edge(I);
imshow(~BW,2)

Binary Image Matrix

Binary images can represent logical, or boolean, matrices. You can create a boolean matrix using MATLAB's logical operators, such as ==, <, and >. Edge extraction and region of interest operations also create binary images. Because of their special nature, binary images are useful as masks for filtering operations.
The command

\[
\text{BW} = \begin{bmatrix}
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{bmatrix};
\]

creates a simple 5-by-5 binary image. To display this image, enter the matrix and use

\[
\text{imshow(BW,2)}
\]

Like intensity images, binary images use gray scale colormaps. As long as the colormap defines at least two colors, its length does not affect image display.
RGB Images

The red, green, blue (RGB) image is a standard way to represent color data. Many color scanners, for example, produce RGB images. In MATLAB, the red, green, and blue components of an RGB image reside in three separate intensity matrices, each having the same row and column dimensions as the original RGB image. The intensities of corresponding pixels from each matrix combine to create the actual pixel color at a given location.

For example, assume the red component matrix for an image contains the value 0.1238 at location (112,86), the green matrix contains 0.9874 at that location, and the blue matrix contains 0.2543 at the same location. Then the color for the pixel at (112,86) is

\[
\text{color} = [0.1238\ 0.9874\ 0.2543];
\]

```matlab
load trees
[R,G,B] = ind2rgb(X,map)
imshow(R,G,B)
```

RGB Image Matrices

For some operations, it is helpful to convert an indexed image to RGB format. To filter an indexed image, for example, you can separate the image into its red, green, and blue components, then filter each one separately. Otherwise, MATLAB simply applies the filter to the indices in the indexed image matrix.
The statements

\[
R = [.4 .4 0 .4 1  \\
   .4 .4 0 1 .4  \\
   0 0 1 0 0  \\
   .4 1 0 .4 0  \\
];
\]

\[
G = [.4 .4 .6 .4 0  \\
   .4 .4 .6 0 .4  \\
   .6 .6 0 .6 .6  \\
   .4 0 .6 .4 .4  \\
];
\]

\[
B = [.4 .4 1 .4 0  \\
   .4 .4 1 0 .4  \\
   1 1 0 1 1  \\
   .4 0 1 .4 0  \\
];
\]

create an RGB image of a blue cross and red line on a gray background. To display this image, enter these matrices and use

\[
\text{imshow}(R,G,B)
\]

Systems other than MATLAB often store the red, green, and blue components of an RGB image as the first three bytes of a 4-byte integer, where the fourth byte is padding. In this format, each component has a range of 0 to 255, so there are up to \(256^3\) colors available. To work with
such images in MATLAB, split the red, green, and blue components into separate matrices and scale them to create intensity matrices.

Image Decks

For some applications, the object of interest is a collection of images related by time or view. Magnetic resonance imaging (MRI) slices or movie frames are examples of this type of object. MATLAB packs images of this type into an image deck. Each image within the deck is called an image slice. All the images in a deck are the same size.

You can collect any of the supported image types into a deck. Indexed, intensity, and binary images can reside in a single deck, while RGB images reside in separate red, green, and blue decks.

To demonstrate how MATLAB stores image decks, assume you have a deck D consisting of five image slices (D1, D2, D3, D4, and D5). MATLAB stores the deck D as if you created it with

\[
D = [D1,D2,D3,D4,D5];
\]

If the size of each image slice is m-by-n, D is a matrix of size m-by-(5+n).

The function \texttt{imslice} indexes an image deck for insertion or extraction. You can use \texttt{imslice} to extract subdecks or individual image slices from a deck, or insert them into a deck. For example, the command below extracts the second slice of the deck D, where \texttt{siz} is a vector listing the image deck dimensions, \texttt{[row\_size column\_size num\_slices]}.

\[
A = D(\texttt{imslice(siz,2)})
\]

To insert the matrices A1 and A2 into image planes 3 and 4 within D, use

\[
D(\texttt{imslice(siz,3:4)}) = [A1,A2];
\]
The commands below display a medical image deck. Note that `montage` displays images in a row-wise manner. The first slice in the deck appears in the first position of the first row, the next slice in the second position of the first row, and so on.

```matlab
load mri
colormap(map)
montage(D,siz);
```
Changing Image Types

The Image Processing Toolbox provides a collection of functions that lets you easily convert any image to another image type. These functions have mnemonic names, where ind represents an indexed image, gray represents an intensity image, bw represents a black and white image, rgb represents an RGB image, and im represents any image type. Each name consists of the starting image format, the number 2, and the target format. For example, ind2gray converts an indexed image to intensity format.

In addition to these standard conversion tools, some functions return a new image type as part of the operation they perform. For example, the region of interest routines all create a binary mask that can overlay an indexed or intensity image for filtering or other specialized operations.

The figure below shows the image conversion techniques available.
Getting Started

The table below summarizes these image conversion techniques:

<table>
<thead>
<tr>
<th>To convert...</th>
<th>To...</th>
<th>Use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexed image</td>
<td>Binary image</td>
<td>roicolor, roipoly, im2bw</td>
</tr>
<tr>
<td>Indexed image</td>
<td>Intensity image</td>
<td>ind2gray</td>
</tr>
<tr>
<td>Indexed image</td>
<td>RGB image</td>
<td>ind2rgb</td>
</tr>
<tr>
<td>Intensity image</td>
<td>Binary image</td>
<td>edge, im2bw, roicolor, roipoly</td>
</tr>
<tr>
<td>Intensity image</td>
<td>Indexed image</td>
<td>grayslice, gray2ind</td>
</tr>
<tr>
<td>Intensity image</td>
<td>RGB</td>
<td>Use the original intensity matrix for all three RGB components.</td>
</tr>
<tr>
<td>RGB image</td>
<td>Indexed image</td>
<td>rgb2ind</td>
</tr>
<tr>
<td>RGB image</td>
<td>Intensity image</td>
<td>rgb2gray</td>
</tr>
<tr>
<td>RGB Image</td>
<td>Binary image</td>
<td>im2bw</td>
</tr>
<tr>
<td>Binary image</td>
<td>Indexed image</td>
<td>gray2ind</td>
</tr>
<tr>
<td>General matrix</td>
<td>Intensity image</td>
<td>mat2gray</td>
</tr>
<tr>
<td>Deck</td>
<td>Image, subdeck</td>
<td>imslice</td>
</tr>
</tbody>
</table>

Understanding Color in MATLAB

As described earlier, MATLAB uses a colormap matrix to define the color or intensity values for an image. A colormap specifies the values to be installed in your computer’s hardware lookup table, a device that defines the colors used for screen displays. Several considerations, including hardware lookup table size and operating system requirements, determine the number of colors you can display at one time.

The default colormap length, determined by the 'MinColorMap' figure property, is 64. The maximum number of display colors available at any time is usually 256, allowing you to display approximately three images with different colormaps if each colormap has no more than 64 colors.
The number of colors MATLAB can control may differ slightly on some platforms. In some cases, it is slightly less than 256. On other platforms, MATLAB can display more than 256 colors. See your MATLAB User's Guide and system documentation for details.
Getting Started

The toolbox function **colormap** installs a specified colormap matrix in the hardware lookup table. Each pixel in the image has an index value into the color indirection table, which in turn indexes the hardware lookup table to determine the actual screen colors. When you use colormap to change the colormap matrix, you indirectly change the hardware lookup table, and thus the actual RGB values that comprise the screen display. MATLAB rearranges the colormap, and sometimes approximates the colors it contains, according to the restrictions of each platform.

If your colormap is longer than the number of available colors, MATLAB eliminates as many colors as needed. MATLAB approximates the removed colors by the closest color in your colormap. If the color for any pixel is removed, that pixel simply uses the closest approximation.

Whenever possible, MATLAB does not affect other figures as you add new figures with different colormaps. If you exceed the limit for your system, however, you may get undesirable results. To avoid color display problems, always try to limit the total number of colors you use to the maximum available on your system.

In addition to the colors in the colormap, MATLAB must also reserve color slots for any Handle Graphics™ objects that have color properties. Examples include figures, axes, lines, uicontrols, and text. The colors that MATLAB allocates for these objects reduce the number of color slots available for a colormap. By default, each figure allocates eight Handle Graphics colors: one for axes (white), one for the figure background (black), and six line colors (yellow, magenta, cyan, red, green, and blue). This results in eight fewer colors for image display.

Graphics functions that use pseudocolor — **mesh**, **surf**, **pcolor**, and others — map a color matrix, **c**, with values in the range \([\text{cmin}, \text{cmax}]\), to an array of indices, **k**, with values in the range \([1, m]\). The values of \text{cmin} and \text{cmax} are either \text{min}(\text{min}(c)) and \text{max}(\text{max}(c)), or are specified by **caxis**. The mapping from \text{c} to \text{k} is linear, with \text{cmin} mapping to 1 and \text{cmax} mapping to \text{m}. The colormap uses the resulting indices to determine the color for each matrix element. See the description of **caxis** in the *MATLAB Reference Guide* for details.

If you are unable to achieve the color results you want, see “Compensating for Long Colormaps” in the “Advanced Topics” section.