What Is the Image Processing Toolbox?

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment. The toolbox supports a wide range of image processing operations, including:

• Spatial image transformations

• Morphological operations

• Neighborhood and block operations

• Linear filtering and filter design

• Transforms

• Image analysis and enhancement

• Image registration

• Deblurring

• Region of interest operations

Many of the toolbox functions are MATLAB M-files, a series of MATLAB statements that implement specialized image processing algorithms. You can view the MATLAB code for these functions using the statement

type function_name

You can extend the capabilities of the Image Processing Toolbox by writing your own M-files, or by using the toolbox in combination with other toolboxes, such as the Signal Processing Toolbox and the Wavelet Toolbox.

For a list of the new features in Version 3.0, see the Release Notes documentation.

Images in MATLAB and the Image Processing Toolbox

The basic data structure in MATLAB is the array, 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.

MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image. (Pixel is derived from picture element and usually denotes a single dot on a computer display.) For example, an image composed of 200 rows and 300 columns of different colored dots would be stored in MATLAB as a 200-by-300 matrix. Some images, such as RGB, require a three-dimensional array, where the first plane in the third dimension represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities.

This convention makes working with images in MATLAB similar to working with any other type of matrix data, and makes the full power of MATLAB available for image processing applications. For example, you can select a single pixel from an image matrix using normal matrix subscripting.

I(2,15)

This command returns the value of the pixel at row 2, column 15 of the image I.

Storage Classes in the Toolbox

By default, MATLAB stores most data in arrays of class double. The data in these arrays is stored as double precision (64-bit) floating-point numbers. All MATLAB functions work with these arrays.

For image processing, however, this data representation is not always ideal. The number of pixels in an image may be very large; for example, a 1000-by-1000 image has a million pixels. Since each pixel is represented by at least one array element, this image would require about 8 megabytes of memory.

To reduce memory requirements, MATLAB supports storing image data in arrays as 8-bit or 16-bit unsigned integers, class uint8 and uint16. These arrays require one eighth or one fourth as much memory as double arrays.

Image Types in the Toolbox

The Image Processing Toolbox supports four basic types of images:

• Index images

• Intensity images

• Binary images

• RGB images

This section discusses how MATLAB and the Image Processing Toolbox represent each of these image types.

Indexed Images

An indexed image consists of a data matrix, X, and a colormap matrix, map. The data matrix can be of class uint8, uint16, or double. The colormap matrix is an m-by-3 array of class double containing floating-point values in the range [0,1]. Each row of map specifies the red, green, and blue components of a single color. An indexed image uses direct mapping of pixel values to colormap values. The color of each image pixel is determined by using the corresponding value of X as an index into map. The value 1 points to the first row in map, the value 2 points to the second row, and so on.

A colormap is often stored with an indexed image and is automatically loaded with the image when you use the imread function. However, you are not limited to using the default colormap—you can use any colormap that you choose. The figure below illustrates the structure of an indexed image. The pixels in the image are represented by integers, which are pointers (indices) to color values stored in the colormap