International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 2

Image Processing in MATLAB 9.3

Shete S. G.
School of Mathematical Sciences,
S.R.T.M.University, Nanded, Maharashtra, India

Ghadge Nagnath G.
Assistant Professor, Department of
Computer Science, Mahatma Basweshwar,
Mahavidyalaya, Latur, Maharashtra, India

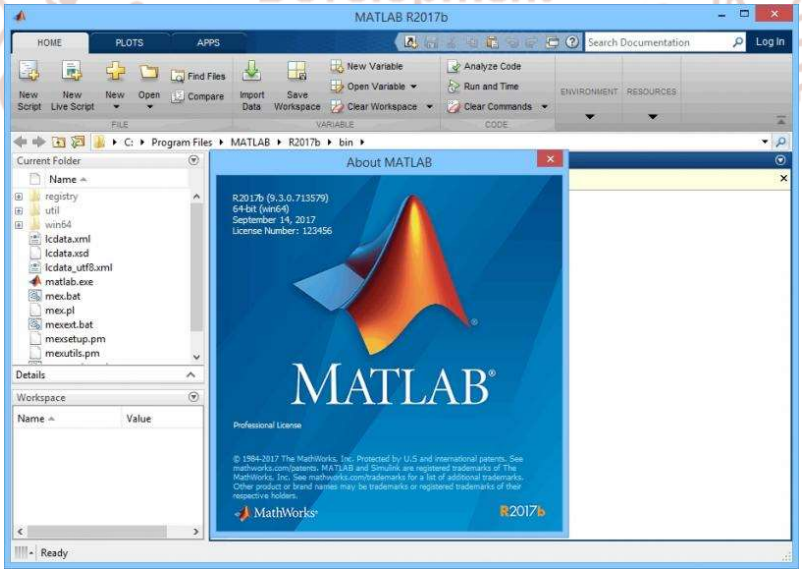
ABSTRACT

In this paper we introduce how to handle different kinds of image formats in MATLAB 9.3 by using Matlab Workspace & its Various Commands. Also we illustrated example of processing the images.

Keywords: *Image Processing, Image Formats, reading Images*

Introduction

This worksheet is an introduction on how to handle images in MATLAB 9.3. When working with images in MATLAB 9.3, there are many things to keep in mind such as loading an image, using the right format, saving the data as different data types, how to display an image, conversion between different image formats, etc. This worksheet presents some of the commands designed for these operations. Most of these commands require you to have the *Image processing tool box* installed with MATLAB 9.3. To find out if it is installing, type ver at the MATLAB 9.3 prompt. This gives you a list of what tool boxes that are installed on your system.



For further reference on image handling in MATLAB 9.3 you are recommended to use MATLAB 9.3's help browser. There is an extensive on-line manual for the Image processing tool box that you can access via MATLAB 9.3's help browser.

The first sections of this worksheet are quite heavy. The only way to understand how the presented commands work is to carefully work through the examples given at the end of the worksheet. Once you can get these examples to work, experiment on your own using your favorite image!

Fundamentals

A digital image is composed of *pixels* which can be thought of as small dots on the screen. A digital image is an instruction of how to color each pixel. We will see in detail later on how this is done in practice. A typical size of an image is 512-by-512 pixels. Later on in the course you will see that it is convenient to let the dimensions of the image to be a power of 2. For example, $2^4.=16$ In the general case we say that an image is of size *m*-by-*n* if it is composed of *m* pixels in the vertical direction and *n* pixels in the horizontal direction.

Let us say that we have an image on the format 512-by-1024 pixels. This means that the data for the image must contain information about 524288 pixels, which requires a lot of memory! Hence, *compressing* images is essential for efficient image processing. You will later on see how Fourier analysis and Wavelet analysis can help us to compress an image significantly. There are also a few "computer scientific" tricks (for example entropy coding) to reduce the amount of data required to store an image.

Image formats supported by MATLAB 9.3

The following image formats are supported by MATLAB 9.3:

- BMP
- PNG
- SCT
- GIF
- HDF
- JPEG
- PCX
- TIFF
- XWB

Most images you find on the Internet are JPEG-images which is the name for one of the most widely used compression standards for images. If you have stored an image you can usually see from the suffix what format it is stored in. For example, an image named myimage.jpg is stored in the JPEG format and we will see later on that we can load an image of this format into MATLAB 9.3.

Working formats in MATLAB 9.3

If an image is stored as a JPEG-image on your disc we first read it into MATLAB 9.3. However, in order to start working with an image, for example perform a wavelet transform on the image, we must convert it into a different format. This section explains four common formats.

Intensity image (GSI)

This is the equivalent to a "Gray Scale Image (GSI)" and this is the image we will mostly work with in this course. It represents an image as a matrix where every element has a value corresponding to how bright/dark the pixel at the corresponding position should be colored. There are two ways to represent the number that represents the brightness of the pixel: The double class (or data type). This assigns a floating number ("a number with decimals") between 0 and 1 to each pixel. The value 0 corresponds to black and the value 1 corresponds to white. The other class is called uint8 which assigns an integer between 0 and 255 to represent the brightness of a pixel. The value 0 corresponds to black and 255 to white. The class uint8 only requires roughly 1/8 of the storage compared to the class double. On the other hand, many mathematical functions can only be applied to the double class. We will see later how to convert between double and uint8.

Binary image

This image format also stores an image as a matrix but can only color a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white.

Indexed image

This is a practical way of representing color images. (In this course we will mostly work with Gray Scale Image (GSI) but once you have learned how to work with a Gray Scale Image (GSI) you will also know the principle how to work with color images.) An indexed image stores an image as two matrices. The first matrix has the same size as the image and one number for each pixel. The second matrix is called the *color map* and its size may be different from the image. The numbers in the first matrix is an instruction of what number to use in the color map matrix.

RGB image

This is another format for color images. It represents an image with three matrices of sizes matching the

image format. Each matrix corresponds to one of the colors red, green or blue and gives an instruction of how much of each of these colors a certain pixel should use.

Multiframe image

In some applications we want to study a sequence of images. This is very common in biological and medical imaging where you might study a sequence of slices of a cell. For these cases, the multiframe format is a convenient way of working with a sequence of images. In case you choose to work with biological imaging later on in this course, you may use this format.

How to convert between different formats

The following table shows how to convert between the different formats given above. *All these commands require the Image processing tool box!*

Image format conversion

(Within the parenthesis you type the name of the image you wish to convert.)

Operation:	**Matlab Command**
Convert between intensity format to indexed	format.gray2ind()
Convert between indexed format to intensity format.	ind2gray()
Convert between indexed format to RGB format.	ind2rgb()
Convert a regular matrix to intensity format by scaling .	mat2gray()
Convert between RGB format to intensity format.	rgb2gray()
Convert between RGB format to indexed format.	rgb2ind()

The command mat2gray is useful if you have a matrix representing an image but the values representing the gray scale range between, let's say, 0 and 1000. The command mat2gray automatically re scales all entries so that they fall within 0 and 255 (if you use the uint8 class) or 0 and 1 (if you use the double class).

How to convert between double and uint8

When you store an image, you should store it as a uint8 image since this requires far less memory than double. When you are processing an image (that is performing mathematical operations on an image) you should convert it into a double. Converting back and forth between these classes is easy.

I=im2double(I);
converts an image named I from uint8 to double.

I=im2uint8(I);
converts an image named I from double to uint8.

How to read files

When you encounter an image you want to work with, it is usually in form of a file (for example, if you down load an image from the web, it is usually stored as a JPEG-file). Once we are done processing an image, we may want to write it back to a JPEG-file so that we can, for example, post the processed image on the web. This is done using the imread and imwrite commands. *These commands require the Image processing tool box!*

Reading and writing image files

Operation:	**MATLAB 9.3 command:**
Read an image. (Within the parenthesis you type the name of the image file you wish to read. Put the file name within single quotes ' '.)	imread()
Write an image to a file. (As the first argument within the parenthesis you type the name of the image you have worked with. As a second argument within the parenthesis you type the name of the file and format that you want to write the image to. Put the file name within single quotes ' '.)	imwrite(,)

Make sure to use semi-colon ; after these commands, otherwise you will get LOTS OF number scrolling on your screen... The commands imread and imwrite

support the formats given in the section "Image formats supported by MATLAB 9.3" above.

Loading and saving variables in MATLAB 9.3

This section explains how to load and save variables in MATLAB 9.3. Once you have read a file, you probably convert it into an intensity image (a matrix) and work with this matrix. Once you are done you may want to save the matrix representing the image in order to continue to work with this matrix at another time.

This is easily done using the commands save and load. Note that save and load are commonly used MATLAB 9.3 commands, and works independently of what tool boxes that are installed.

Loading and saving variables

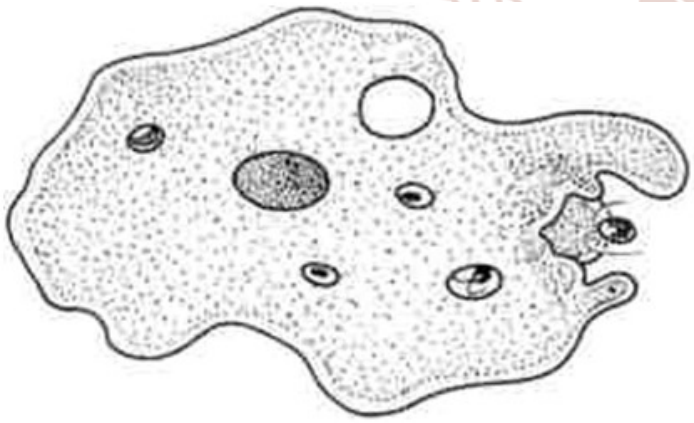
Operation	MATLAB 9.3 Command
Save the Variable X.	save x
Load the Variable X	load x

Examples

In the first example we will down load an image from the web, read it into MATLAB 9.3, investigate its format and save the matrix representing the image.

Example 1

Down load the following image (by clicking on the image using the right mouse button) and save the file as cell1.jpg.



This is an image of a cell taken by google Images from robinsonimagelibrary.com

Now open Matla and make sure you are in the same directory as your stored file. (You can check what files your directory contains by typing ls at the MATLAB 9.3 prompt. You change directory using the command cd.) Now type in the following commands and see what each command does. (Of course, you do not have to type in the comments given in the code after the % signs.)

```
I=imread('cell1.jpg'); % Load the image file and store it as the variable I.
whos % Type "whos" in order to find out the size and class of all stored variables.
save I % Save the variable I.
ls % List the files in your directory.
```

There should now be a file named "I.mat" in you directory containing your variable I.

Note that all variables that you save in MATLAB 9.3 usually get the suffix .mat.

Next we will see that we can display an image using the command imshow. This command requires the image processing tool box. Commands for displaying images will be explained in more detail in the section "How to display images in MATLAB 9.3" below.

```
clear % Clear MATLAB 9.3's memory.
load I % Load the variable I that we saved above.
whos % Check that it was indeed loaded. imshow(I)
% Display the image
I=im2double(I); % Convert the variable into double.
whos % Check that the variable indeed was converted into double
%       The next procedure cuts out the upper left corner of the image
%       and stores the reduced image as Ired.
for i=1:256
for j=1:256
Ired(i,j)=I(i,j);
end
end
whos % Check what variables you now have stored.
imshow(Ired) % Display the reduced image.
```

How to display an image in MATLAB 9.3

Here are a couple of basic MATLAB 9.3 commands (do not require any tool box) for displaying an image.

Displaying an image given on matrix form

Operations	MATLAB 9.3 Command
Display an image represented as the matrix X.	imagesc(X)
Adjust the brightness. s is a parameter such that -1<s<0 gives a darker image, 0<s<1 gives a brighter image.	brighten(s)
Change the colors to gray.	colormap(gray)

Sometimes your image may not be displayed in gray scale even though you might have converted it into a Gray Scale Image (GSI). You can then use the command colormap(gray) to "force" MATLAB 9.3 to use a gray scale when displaying an image.

If you are using MATLAB 9.3 with an Image processing tool box installed, I recommend you to use the command imshow to display an image.

Displaying an image given on matrix form (with image processing tool box)

Operations	MATLAB 9.3 Command
Display an image Represented as a matrix X	Imshow (X)
zoom in	zoom on
Turn off the zoom function.	Zoom off

Conclusion:

Thus here various components of this MATLAB9.3 based image Processing have been discussed. The authors have tried their level best to make the image operations in MATLAB 9.3 user based as possible. The purpose of the image processing is to bring the various image editing functions available in MATLAB9.3 tool box under one common platform and to make it easier for the understanding of any user. Future work can be aimed to expand the set of applications than what has been used here.

References:

1. Digital Image Processing using Matlab By Rafael C. Gonzalez, Richard E. Woods & Steven L. Eddins , Pearson Education, Inc. Pearson Prentice-Hall ISBN - 0-13-008519-7.
2. Digital Image Processing 6th revised & extended edition, Springer-Verlag Berlin Heidelberg 2005, ISBN - 3-540-24035-7.
3. https://in.mathworks.com/support/books/search.html?q=&fq=book-category:image-and-video-processing&page=1
4. Digital Image Processing: A Signal Processing and Algorithmic Approach by D sundarajan, ISBN – 978-981-10-6112-7.