American Journal of Engineering Research (AJER)
E-ISSN: 2320-0847 p-ISSN: 2320-0936
Volume-8, Issue-2, pp-158-172
www.ajer.org
Research Paper Open Access

Get information from the image using MATLAB

Ahmed Mohamed Ali Karrar, Mahoro Adidja, Hamzeh Abdillahi Robleh, Jun Sun*
School of Internet of Things & Engineering Jiangnan University - china
Corresponding Author: Ahmed Mohamed Ali Karrar

ABSTRACT: *Extracting data from images in MATLAB is one of the most important and useful processes for users. This is the main part of this paper. We have analyzed the image in general. We have reviewed the following(Image Coordinate Systems, Image Types in the Toolbox, Images Processing on a GPU, Import Image Data from the Workspace into the Image Viewer, Export Image Data from the Image Viewer App to the Workspace, Save Image Data, Image Information tool, Write Image Data to File in Graphics Format, Get Pixel Information in Image Viewer App, Pixel Values, Contour Plot of Image Data, Display Colors and The integer MATLAB returns represent the number of bits per screen pixel).*

KEYWORDS: *MATLAB, Images in MATLAB, Image processing, Image processing toolbox, Image Information tool.*

Date of Submission: 03-02-2019 Date of acceptance: 19-02-2019

I. INTRODUCTION

MATLAB is being used as a platform for laboratory exercises and the problems classes in the Image Processing half of the Computer Graphics and Image Processing. This paper describes how to get information from the image using Matlab.

MATLAB could be a data analysis and visualization tool designed to create matrix manipulation as simple as possible. In addition, it's powerful graphics capabilities and its own programming language. The basic MATLAB distribution may be expanded by adding a range of toolboxes, the one relevant to present.

MATLAB's basic data structure is the matrix1. In MATLAB a single variable is a 1 x 1 matrix, a string is a 1 x n matrix of chars. An image is an x m matrix of pixels [1].

1.2 Image Processing Toolbox Product Description:

Image Processing Toolbox provides a comprehensive set of reference-standard algorithms and workflow apps for image process, analysis, image, and algorithm development [2]. You'll be able to perform image segmentation, image improvement, noise reduction, geometric transformations, image registration, and 3D image process [3].

Image Processing Toolbox apps allow you to automatize common image process workflows [1]. You'll be able to interactively section image information, compare image registration techniques, and batch-process giant datasets [4]. Image functions and apps allow you to explore pictures, 3Dvolumes, and videos; modify contrast; produce histograms, and manipulate regions of interest (ROIs) [5].

You can accelerate your algorithms by running them on multicore processors and GPUs.Many tool box functions support C/C++ code generation for desktop prototyping and embedded
Vision system readying [2].

1.3 Images in MATLAB

The basic data structure in MATLAB is the array, an ordered set of real or complex elements. This object is suited to the illustration of images, real-valued ordered sets of color or intensity data.

MATLAB stores most images as two-dimensional matrices, during which every part of the matrix corresponds to a single discrete pixel in the displayed image. Pixel is derived from picture element and usually denotes a single dot on a computer display.) as an example, a picture composed of two hundred rows and three hundred columns of diff erent colored dots would be keep in MATLAB as a 200-by-300 matrix.

Some images, like true color images, represent images employing a three-dimensional array. In true color images, the primary plane within the 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 operating with images in MATLAB almost like operating with the other form of numeric knowledge and makes the complete power of MATLAB accessible for image processing applications [6].

1.4 Image Coordinate Systems

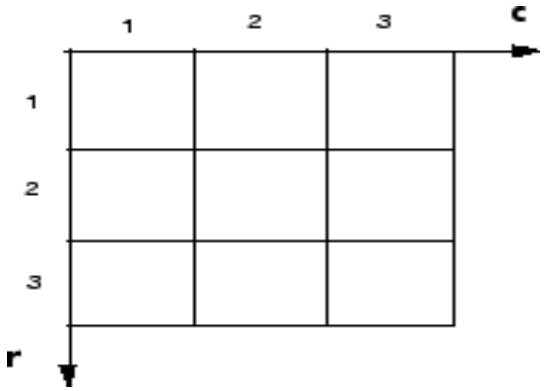
MATLAB stores most images as two-dimensional arrays (i.e., matrices), during which every component of the matrix corresponds to a single pixel in the displayed image. To access locations in images, the Image Processing Toolbox uses many diff erent image coordinate systems as conventions for representing Images as arrays.

• "Pixel Indices" as a result of images are arrays; you'll be able to use standard MATLAB indexing.

• "Spatial Coordinates" you'll be able to contemplate locations in images as positions on a plane using Cartesian coordinates.

Pixel Indices

Often, the foremost convenient technique for expressing locations in an image is to use pixel indices. The image is treated as a grid of distinct elements, ordered from prime to bottom, and left to right, as illustrated by the following figure [7]



1.4.1 Pixel Indices

For pixel indices, the row will increase downward, whereas the column will increase to the proper. Pixel indices are integer values and vary from one to the length of the row or column.

There is a one-to-one correspondence between pixel indices and subscripts for the first 2 matrix dimensions in MATLAB. As an example, the data for the pixel in the fifth row, the second is stored in the matrix element (5, 2). You use normal MATLAB matrix subscription to access values of individual pixels. As an example, the MATLAB code

I (2, 15)

Returns the value of the pixel at row 2, column 15 of the image I. Similarly, the MATLAB code

RGB(2,15,:)

Returns the R, G, B values of the pixel at row 2, column 15 of the image RGB.

The correspondence between pixel indices and subscripts for the first 2 matrix dimensions in MATLAB [7] makes the relationship between an image's data matrix the way the image is displayed easy to understand [8].

1.4.2 Spatial Coordinates

Another technique for expressing locations in an image is to use a system of unceasingly variable coordinates instead of distinct indices. This permits you to think about an image as covering a sq. patch, as an example. In a spatial coordinate system like this, locations in an image are positions on a plane, and that they are delineate in terms of x and y (not row and column as within the picture element categorization system). From this Cartesian perspective, an (x, y) location like (3.2,5.3 is meaningful and is distinct from pixel (5,3).

The Image Processing Toolbox defines 2 kinds of spatial coordinate systems

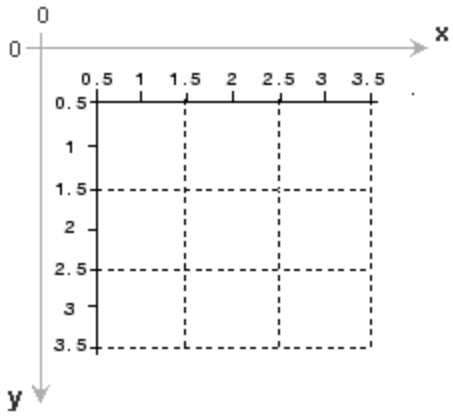
•Intrinsic Coordinates A spatial coordinate system that corresponds to pixel indices.

• World coordinates a spatial coordinate system that relates the image to some other coordinate space [7].

1.4.3 Intrinsic Coordinates

By default, the toolbox uses a spatial coordinate system for an image that corresponds to the image's pixel indices. it's referred to as the intrinsic coordinate system and is illustrated within the following figure.

Notice that y will increase downward as a result of this orientation is in step with the manner within which digital images are sometimes viewed [7].



1.4.3.1 Intrinsic Coordinate System

The intrinsic coordinates (x,y) of the center point of any pixel are identical to the column and row indices for that pixel.as an example, the centre point of the pixel in row five, column3 has spatial coordinates x = 3.0, y = 5.0. This correspondence simplifies many toolbox functions considerably. Be aware, however, that the order of coordinate specification (3.0,5.0) is reversed in intrinsic coordinates relative to pixel indices (5,3).
Many functions primarily work with spatial coordinates rather than pixel indices, however as long as you're victimization the default spatial coordinate system (intrinsic coordinates), you'll be able to specify locations in terms of their columns (x) and rows (y).

When observing the intrinsic coordinate system, note that the higher left corner of the image is found at (0.5,0.5), not at (0,0), and the lower right corner of the image is located at (numCols + 0.5, numRows + 0.5), wherever numCols and numRows are the numbers of rows and columns within the image. In contrast, the upper left pixel is pixel (1,1) and the lower right pixel is pixel (numRows, numCols). The middle of the upper left pixel is (1.0, 1.0) and the center of the lower right pixel is (numCols, numRows). In fact, the middle coordinates of every pixel are integer valued. the middle of the pixel with indices (r, c) — wherever r and c are integers by definition — falls at the point x = c, y = r within the intrinsic coordinate system [7].

1.5 World Coordinates

In some things, you may need to use a world coordinate system (also known as a nondefault spatial coordinate system). as an example, you may shift the origin by specifying that upper left corner of an image is the point (19.0,7.5), rather than (0.5,0.5). Or, you may need to specify a coordinate system during which each constituent covers a5-by-5 meter patch on the ground. There are many ways that to define a world coordinates system:

1.5.1 Define World Coordinates Using XData and YData Properties

To define a world coordinate system for an image, specify the XData and YData image properties for the image. The XData and YData image properties are two-element vectors that management the vary of coordinates spanned by the image. once you do that, the MATLAB axis coordinates become identical to the world (nondefault) coordinates. If you are doing not specify XData and YData, the axes coordinates are identical to the intrinsic coordinates of the image. By default, for an image A, XData is [1 size(A,2)], and YData is [1 size(A,1)]. With this default, the world coordinate system and intrinsic coordinate system coincide perfectly. (Another way to define a world coordinate system is to use spatial referencing.

As an example, if A is a 100 row by 200 column image, the default XData is [1 200], and the default YData is [1 100]. The values in these vectors are actually the coordinates for the centre points of the first and last pixels (not the pixel edges), therefore the actual coordinate vary spanned is slightly larger. for example, if XData is [1 200], the interval in X spanned by the image is [0.5 200.5].
It's additionally attainable to set XData or YData such that the x-axis or y-axis is reversed. You'd do that by inserting the larger worth initial. (For example, set the YData to [1000 1].) This is a common technique to use with geospatial data.

Several toolbox functions accept this XData and YData as arguments and return coordinates within the world coordinate system: bwselect, imcrop, impixel, roipoly, and imtransform [9].

II. IMAGE TYPES IN THE TOOLBOX

The Image Processing Toolbox software defines four basic types of images, summarized within the following table. These image sorts confirm the approach MATLAB interprets array elements as pixel intensity values.

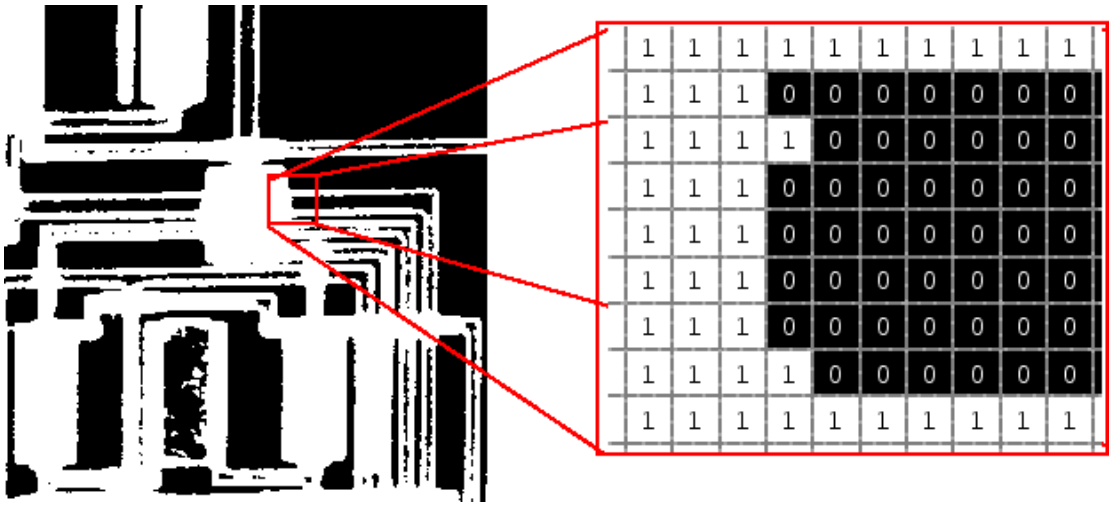
All images in Image Processing Toolbox are assumed to possess real, no sparse, numeric or logical values unless otherwise specific.

Image type	Interpretation
Binary Images	Image data are stored as an m-by-n logical array. Array values of 0 and 1 are interpreted as black and white, respectively.
Indexed Images	Image data are stored as an m-by-n numeric matrix whose elements are direct indices into a color map. Each row of the color map specifies the red, green, and blue components of a single color. • For single or double arrays, integer values range from [1, p]. • For logical, uint8, or uint16 arrays, values range from [0, p-1]. The colormap is a c-by-3 array of class double.
Grayscale Images (Also known as an intensity image)	Image data are stored as an m-by-n numeric array whose elements specify intensity values. • For single or double arrays, values range from [0, 1]. • For uint8 arrays, values range from [0,255]. • For uint16, values range from [0, 65535]. • For int16, values range from [-32768, 32767].
Truecolor Images (Also known as an RGB image)	Image data are stored as an m-by-n-by-3 numeric array whose elements specify the intensity values of one of the three color channels. For RGB images, the three channels represent the red, green, and blue signals of the image. • For single or double arrays, RGB values range from [0, 1]. • For uint8 arrays, RGB values range from [0,255]. • For uint16, RGB values range from [0, 65535]. There are other models, called color spaces, that describe colors using three color channels. For these color spaces, the range of each data type may differ from the range allowed by images in the RGB color space. For example, pixel values in the L*a*b* color space of data type double can be negative or greater than 1. For more information, see "Understanding Color Spaces and Color Space Conversion" on page 15-20.

2.1 Binary Images

In a binary image, every pixel assumes one in every of solely 2 separate values: one or zero. A binary image is hold on as a logical array. By convention, this documentation uses the variable name BW to refer to binary images.

The following figure shows a binary image with a close-up read of a some of the pixel values [10].



Pixel Values in a Binary Image

2.2 Indexed Images

An indexed image consists of an array and a colormap matrix. An indexed image uses direct mapping of pixel values within the array to colormap values. By convention, this documentation uses the variable name X to ask the array and map to ask the colormap. The colormap matrix is AN m-by-3 array of class double containing floating-point values within the vary [0,1]. every row of map specifies the red, green, and blue components of a single color.

The pixel values within the array are direct indices into a colormap. the colour of every image pixel is set determined by using the corresponding value of X as an index into map. The relationship the values within the image matrix and also the colormap depends on the class of the image matrix:

• If the image matrix is of class single or double, the colormap ordinarily contains whole integer values within the vary [1, p], wherever p is that the length of the colormap. The value one points to the first row within the colormap, the value two points to the second row, and so on.

• If the image matrix is of class logical, uint8 or uint16, the colormap ordinarily contains whole integer values within range [0, p–1]. The value zero points to the first row within the colormap, the value one points to the second row, and so on.

A colormap is usually hold on with an indexed image and is automatically loaded with the image after you use the imread function after you read the image and also the colormap into workspace as separate variables, you need to keep track of the association between the image and colormap. However, you're not limited to using the default colormap you will use any colormap that you simply select[10].

2.3 Grayscale Images

A grayscale image (also called gray-scale, gray scale, or gray-level) could be a information matrix whose values represent intensities at intervals some range. MATLAB stores a grayscale image as a personal matrix, with every element of the matrix corresponding to one image pixel. By convention, this documentation uses the variable name I to ask grayscale images.

The matrix will be of class uint8, uint16, int16, single, or double. Whereas grayscale images are seldom saved with a colormap, MATLAB uses a colormap to show them. For a matrix of class single or double, using the default grayscale colormap, the intensity zero represents black and also the intensity one represents white. For a matrix of type uint8, uint16, or int16, the intensity intmin (class(I)) represents black and also the intensity intmax(class(I)) represents white [10].

2.4 Truecolor Images

A truecolor image is an image within which every pixel is specific by 3 values — one every for the red, blue, and green components of the pixel's color. MATLAB store truecolor images as AN m-by-n-by-3 data array that defines red, green, and blue color parts for every individual lpixel. Truecolor images don't use a colormap. The colour of every pixel is set by the mixture of the red, green, and blue intensities hold on in every color plane at the pixel's location.

Graphics file formats store truecolor images as 24-bit images, wherever the red, green, and blue components are eight bits every. This yields a possible of sixteen million colours. The precision with that a real-life image will be replicated has led to the commonly used term truecolor image. A truecolor array will be of class uint8, uint16, single, or double. in an exceedingly truecolor array of class single or double, every color component is a value between 0 and 1. A pixel whose color components are (0,0,0) is displayed as black, and a a pixel whose color components are (1,1,1) is displayed as white. The 3 color components for every pixel are stored along the third dimension of the data array.

for instance, the red, green, and blue color components of the pixel (10,5) are stored in RGB (10,5,1), RGB (10,5,2), and RGB (10,5,3), respectively [10].

III. IMAGES PROCESSING ON A GPU (GPU COMPUTING WITH IMAGE PROCESSING TOOLBOX):

to require advantage of the performance advantages off ered by a contemporary graphics process unit (GPU), certain Image Processing Toolbox functions are enabled to perform image processing operations on a GPU. This may give GPU acceleration for complicated image processing workflows. These techniques will be enforced solely or in combination to satisfy design requirements and performance goals.

To run image processing code on a graphics processing unit (GPU), you need to have the Parallel Computing Toolbox software. To perform an image processing operation on a GPU, follow these steps:

• Move the data from the CPU to the GPU. Use the gpuArray function to transfer AN array from MATLAB to the GPU.

• Perform the image processing operation on the GPU. Any toolbox function that accepts a gpuArray object as AN input will work on a GPU. As an example, you'll pass a gpuArray to the imfilter function to perform the filtering operation on a GPU. For a list of all the toolbox functions that are GPU-enabled.
• Move the data back onto the CPU from the GPU. Use the gather function to retrieve AN array from the GPU and transfer the array to the MATLAB workspace as a regular MATLAB array.
When operating with a GPU, note the following:
• Performance improvements can depend on the GPU device.
• There is also little diff erences within the results came back on a GPU from those came back on a CPU [10].

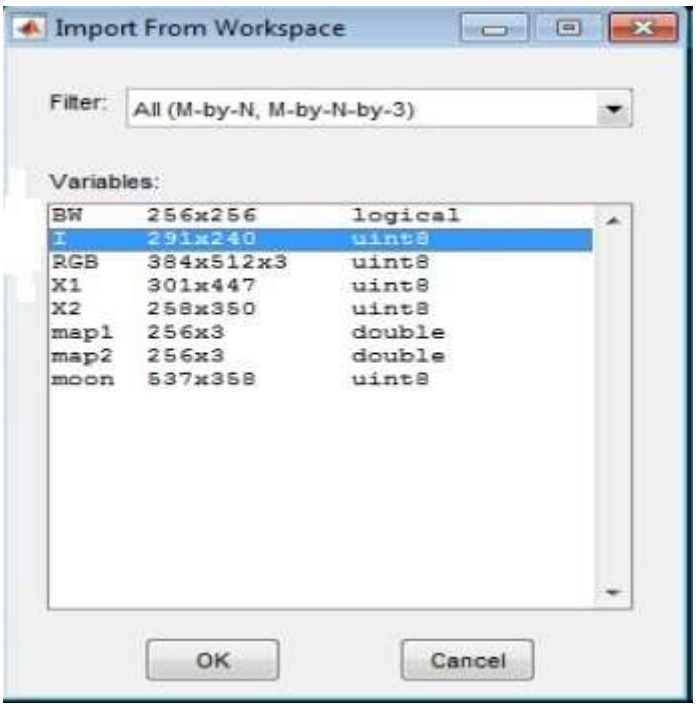
3.1 Display an Image in a Figure Window:
Overview
To display image data, use the imshow function. The following example reads an image into the MATLAB workspace and displays the image in a MATLAB figure window.
moon = imread('moon.tif');
imshow(moon);
The imshow function displays the image in a MATLAB figure window, as shown within the following figure.
Figure???
You can additionally pass imshow the name of a file containing an image.
imshow('moon.tif');

This syntax will be helpful for scanning through images. Note, however, that once you use this syntax, imread doesn't store the image data within the MATLAB workspace. If you would like to bring the image into the space, you need to use the getimage function that retrieves the image data from the current image object. This instance assigns the image data from moon.tif to the variable moon, if the figure window during which it's displayed is currently active.
moon = getimage;

3.2 Import Image Data from the Workspace into the Image Viewer App

To import image data from the MATLAB workspace into the Image Viewer, use the Import from workspace option on the Image Viewer File menu. In the dialog box, shown below, you choose the workspace variable that you simply need to import into the space.

The following figure shows the Import from Workspace dialog box. you'll use the Filter menu to limit the images enclosed within the list to certain image varieties, i.e., binary, indexed, intensity (grayscale), or truecolor.

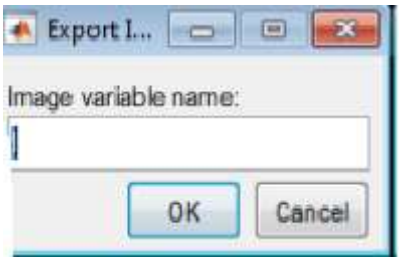


3.3 Export Image Data from the Image Viewer App to the Workspace

To export the image displayed within the Image Viewer to the MATLAB workspace, you'll use the Export to workspace option on the Image Viewer File menu. (Note that once exporting data, changes to the display range won't be preserved.) In the dialog box, exporting data, changes to the display range won't be preserved.) In the dialog box, shown below, you specify the name you would like to assign to the variable within the workspace.

By default, the Image Viewer pre-fills the variable name field with BW biological warfare, for binary images, RGB, for truecolor images, and that I for grayscale orindexed images.

If the Image Viewer contains AN indexed image, this dialog box also contains a field wherever you'll specify the name of the associated colormap



Using the getimage Function to Export Image Data

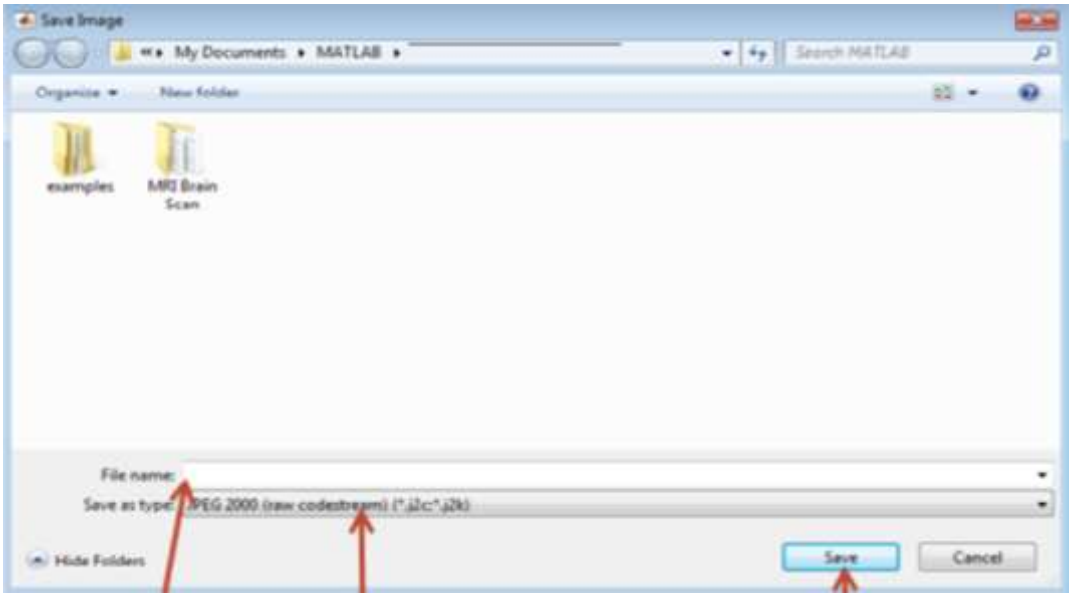
You'll additionally use the getimage function to bring image data from the Image Viewer into the MATLAB workspace.

The getimage function retrieves the image data (CData) from the current image object. You need to use the toolbox function imgca to urge the image object displayed within the Image Viewer. The subsequent example assigns the image data from moon.tif to the variable moon if the figure window during which it's displayed is currently active.

moon = getimage(imgca);

3.4 Save Image Data Displayed in Image Viewer

To save the image data displayed in the Image Viewer, choose the Save as option from the Image Viewer File menu. The Image Viewer opens the Save Image dialog box, shown within the following figure. Use this dialog box to navigate your filing system to determine where to save the image file and specify the name of the file. Choose the graphics file format you would like to use from among several common image file formats listed within the Files of kind menu. If you are doing not specify a file name extension, the Image Viewer adds AN extension to the file associated with the file format selected, such as .jpg for the JPEG format.



1 2 3

(1-Specify file name 2 – select file format 3-click save)

Changes you create to the show vary won't be saved. If you'd prefer to preserve your changes, use imcontrast.

IV. IMAGE INFORMATION TOOL

Use the image info function to create an Image Information tool. The tool displays information about the basic attributes and metadata of the target image in a separate figure.

4.1Get Image Information in Image Viewer App:

To get information about the image displayed within the Image Viewer, use the Image Information tool. The Image data tool will give 2 varieties of information about an image:

. Basic information : Includes width, height, class, and image type. For grayscale and indexed images, this data additionally includes the minimum and maximum intensity values.

. Image metadata: Displays all the metadata from the graphics file that contains the image. this can be the identical data returned by the imfinfo function or the dicominfo function.

Note:

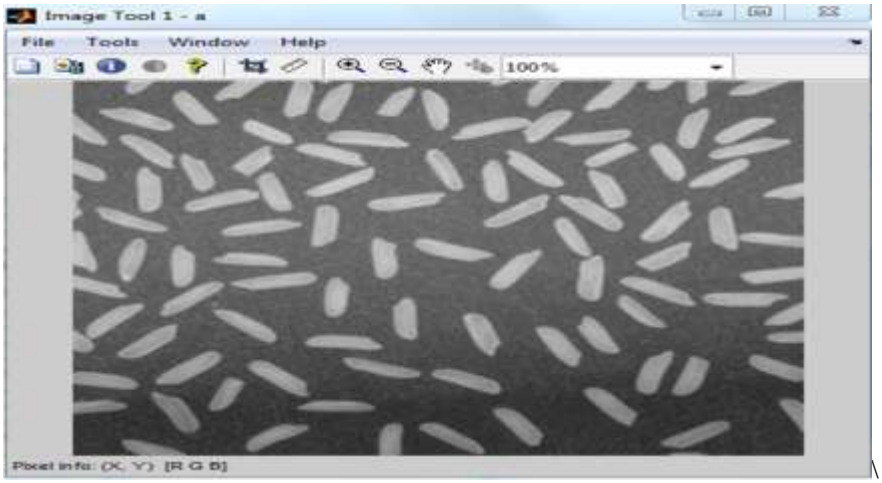
The Image Information tool will display image metadata only if you specify the file name containing the image to Image Viewer, e.g., a=imread('RIC.jpg');

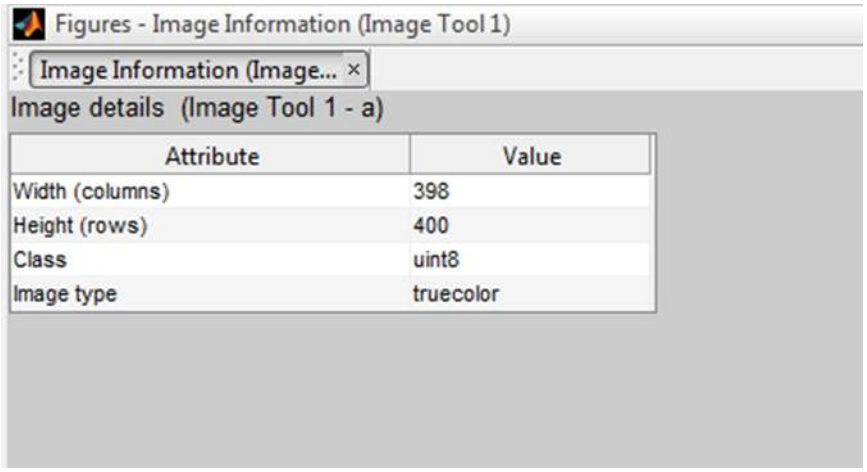
For example, view an image in the Image Viewer.

imtool(a)

Start the Image Information tool by clicking the Image Information button within the Image Viewer toolbar or choosing the Image Information option from the Tools menu within the Image Viewer. (Another possibility is to open a figure window victimization imshow so decision imageinfo from the command.)

The following figure shows the Image Viewer with the Image Information tool. If you specify a file name once your decision the imtool function, the Image Information tool displays each basic image Information, as shown within the figure [11].





4.2info: Image metadata:

Image metadata, such as a structure returned by the functions imfinfo, dicominfo, nitfinfo, interfileinfo, or analyze75info. info can also be a user-created structure.

The table lists the basic image attribute info included within the Image info tool display. Note that the tool contains either four or six fields, depending on the type of image.

Attribute Name	Value
Width (columns)	Number of columns in the image
Height (rows)	Number of rows in the image
Class	Data type used by the image, such as 'uint8' Note: For single or int16 images, imageinfo returns a 'Class' value of 'double', because the image object converts the CData of these images to double.
Image type	One of the image types identified by the Image Processing Toolbox™ software: 'intensity' 'truecolor', 'binary', or 'indexed'.
Maximum intensity or index	For grayscale images, this value represents the lowest intensity value of any pixel. For indexed images, this value represents the lowest index value into a color map. This field is not included for 'binary' or 'truecolor' images.
Maximum intensity or index	For grayscale images, this value represents the highest intensity value of any pixel. For indexed images, this value represents the highest index value into a color map. This field is not included for 'binary' or 'truecolor' images.

imageinfo gets info regarding image attributes by querying the image object's CData. The image object converts the CData for single or int16 images to class double. In these cases, imageinfo(H) displays a'Class' attribute of 'double', although the image is of class single or int16.

V. GET INFORMATION ABOUT GRAPHICS FILES

to get info a couple of graphics file and its contents, use the imfinfo function. you'll use imfinfo with any of the formats supported by MATLAB. Use the imformats function to see that formats are supported.

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

• Name of the file.
• File format
• Version number of the file format.
• File modification date.
• File size in bytes
• Image width in pixels
• Image height in pixels
• Number of bits per pixel
• Image type: TrueColor (RGB), grayscale (intensity), or indexed

The format of the file is inferred from its contents.

If filename is a HDF, ICO, GIF, or CUR file containing quite one image, then info could be a structure array with one element for every image within the file. As an example, info (3) would contain information about the third image in the file.

info = imfinfo(filename,fmt) additionally looks for a file named filename.fmt, if MATLAB cannot find a file named filename.

info = imfinfo(URL) returns information regarding the image at the specified internet resource, URL.

Examples:

Return Information about Graphics File

Find information about the example image, ngc6543a.jpg.

```
info = imfinfo('ngc6543a.jpg');
```

The info structure contains the subsequent information fields: name, FileModDate, File Size, Format, FormatVersion, Width, Height, BitDepth, ColorType, Format Signature, Number of Samples, CodingMethod, CodingProcess, and Comment.

To display information from the structure, as an example CodingMethod, type info.CodingMethod within the command window.

```
info.CodingMethod
ans =
```

'Huffman'
To show all the properties within the structure, type info in the command window.
Input Arguments:
filename — Name of graphics file
character vector | string scalar
Name of graphics file, specified as a character vector or string scalar.
Depending on the location of the file, filename can take on one of these forms.

Location	Form
Current folder or folder on the MATLAB path	Specify the name of the file in filename. Example: 'myImage.jpg'
File in a folder	If the file is not in the current folder or in a folder on the MATLAB path, then specify the full or relative path name. Example: 'C:\myFolder\myImage.ext' Example: '\imgDir\myImage.ext'
URL	if the file is located by an internet URL, then filename must contain the protocol type such as, http://. Example: 'http://hostname/path_to_file/my_image.jpg'
Remote Location (Amazon S3™, Windows Azure® Blob Storage, and HDFS™)	If the file is stored at a remote location, then filename must contain the full path of the file specified as an internationalized resource identifier (IRI) of the form: hdfs:///path_to_file. Example: 's3://bucketname/path_to_file/my_image.jpg'

VI. WRITE IMAGE DATA TO FILE IN GRAPHICS FORMAT

This example shows a way to write image data from the MATLAB workspace to a file in one of the supported graphics file formats using the imwrite function.
Load image data into the workspace. this instance hundreds the indexed image X from a MAT file, clown. Mat, along with the associated colormap map.
load clown
whos
Name Size Bytes Class Attributes
X 200x320 512000 double
caption 2x1 four char
map 81x3 1944 double

Export the image data as a bitmap file using imwrite, specifying the name of the variable and the name of the output file you want to create. Include an extension in the filename, imwrite ttempts to infer the desired file format from it. For example, the file extension .bmp specifies the Microsoft Windows Bitmap format. You'll additionally specify the format expressly as associate argument to imwrite.
imwrite(X,map,'clown.bmp')
Use format-specific parameters with imwrite to regulate aspects of the export process.
For example, with PNG files, you'll specify the bit depth. To illustrate, read an image into the workspace in TIFF format and note its bit depth.
I = imread('cameraman.tif');
s = imfinfo('cameraman.tif');
s.BitDepth
ans = 8
Write the image to a graphics file in PNG format, specifying a bit depth of 4.
imwrite(I,'cameraman.png','Bitdepth',4)
Check the bit depth of the newly created file.
newfile = imfinfo('cameraman.png');
newfile.BitDepth
ans = 4
imwrite uses the subsequent rules to work out the storage class used in the output image.

Storage Class of Image	Storage Class of Output Image File
logical	If the output image file format supports 1-bit images, imwrite creates a 1-bit image file. If the output image file format specified does not support 1-bit images, imwrite exports the image data as a uint8 grayscale image.
uint8	If the output image file format supports unsigned 8-bit images, imwrite creates an unsigned 8-bit image file.
Uint16	If the output image file format supports unsigned 16-bit images (PNG or TIFF), imwrite creates an unsigned 16-bit image file. If the output image file format does not support 16-bit images, imwrite scales the image data to class uint8 and creates an 8-bit image file.
Uint16	Partially supported; depends on file format.
single	Partially supported; depends on file format.
double	MATLAB scales the image data to uint8 and creates an 8-bit image file, because most image file formats use 8 bits.

When a file contains multiple images that are connected in a way, you can call image Processing algorithms directly.
If you're working with a large file, you will need to do block processing to reduce memory usage.

7. Get Pixel Information in Image Viewer App

Determine Individual Pixel Values in Image Viewer:

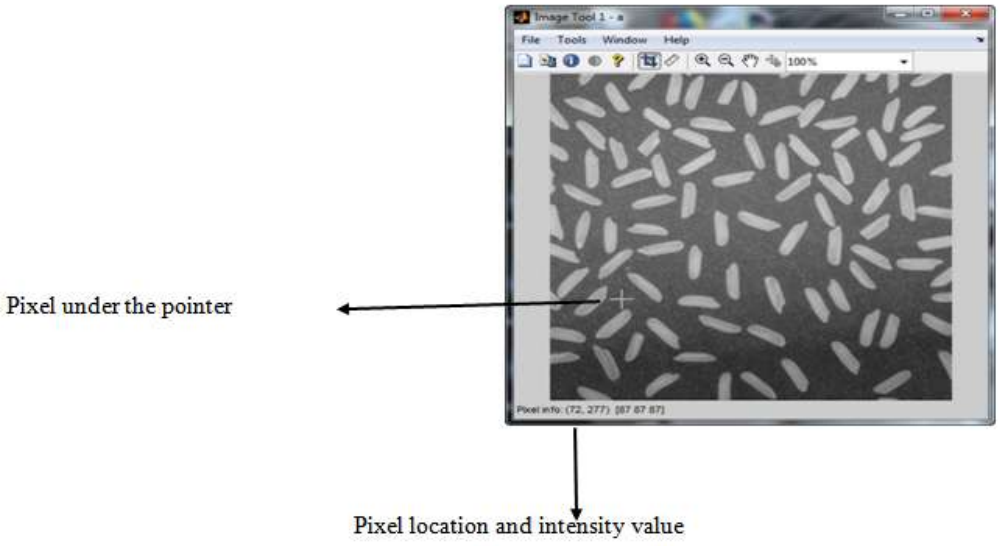
The Image Viewer displays info concerning the location and value of individual pixels in an image within the bottom left corner of the tool. (You also can acquire this info by gap a figure with imshow and then calling impixelinfo from the command line.)

The pixel value and location info represent the pixel under the current location of the pointer. The Image Viewer updates this info as you move the pointer over the image.

For example, read an image within the Image Viewer.

imtool('RIC.jpg')

The following figure shows the Image Viewer with constituent location and value displayed in the Pixel Information tool.



8. Image Region Properties

A region in an image has properties like space, center of mass, orientation, and bounding box.

Distance Transform of a Binary Image

The distance transform of a binary image shows the distance from each pixel to a nonzero pixel. There are other ways to live the distance between 2 pixels.

Intensity Profile of images

The intensity profile of an image is the set of intensity values taken from regularly spaced points on a line or path within the image.

Contour Plot of Image Data

A contour could be a path in a picture on that intensity values are constant. Contour plots will show the outline of objects in an image or represent a 3D shape in a 2-D plane

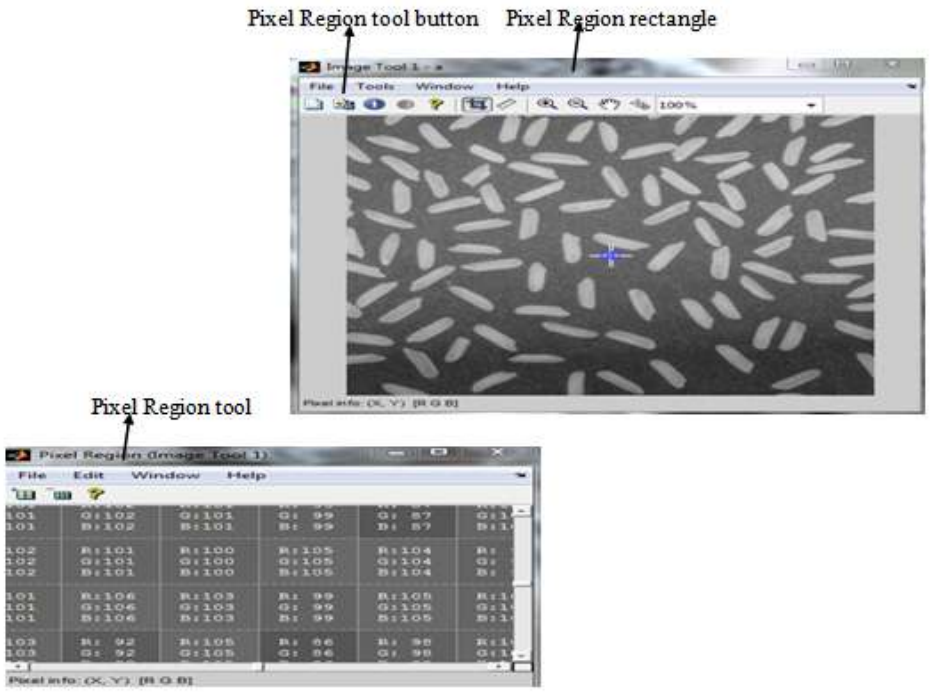
VII. SAVING THE PIXEL VALUE AND LOCATION INFORMATION

To save the pixel location and value information displayed, right-click a pixel in the image and select the Copy pixel info option. The Image Viewer copies the x- and y-coordinates and also the pixel value to the clipboard.

To paste this pixel information into the MATLAB workspace or another application, right click and choose Paste from the context menu.

VIII. DETERMINE PIXEL VALUES IN AN IMAGE REGION

to view the values of pixels in a very specific region of an image displayed within the Image Viewer, use the pixel Region tool. The pixel Region tool superimposes rectangle, called the pixel region rectangle, over the image displayed within the Image Viewer. This rectangle defines the group of pixels that are displayed, in extreme close-up read, within the pixel Region tool window. the subsequent figure shows the Image Viewer with the pixel Region tool. Note however the pixel Region tool includes the value of every pixel within the display [12].



IX. SAVING THE CHANGED IMAGE DATA

By default, if you close up the Image Viewer, it doesn't save the changed image data. to save these changed values, use the Save As option from the Image Viewer File menu to store the modified data in a very file or use the Export to workspace option to save the modified in a work space variable.

X. PIXEL VALUES

To determine the values of 1 or more pixels in an image and return the values in a variable, use the impixel function. You'll be able to specify the pixels by passing their coordinates as input arguments otherwise you can choose the pixels interactively using a mouse. impixel returns the value specified pixels in a variable in the MATLAB workspace [13]. You can get pixel value data interactively using the Image Tool.

Determine Values of Individual Pixels in Images

This example shows the way to use impixel interactively to urge pixel values.

Display an image.

```
s=imread('7777777.jpg')
imshow(s)
```

Call impixel. When called with no input arguments, impixel associates itself with the image in the current axes.
pixel values = impixel
Select the points you would like to examine in the image by clicking the mouse. impixel places a star at every point you choose.
imshow(s)

When you are finished choosing points, press Return. impixel returns the pixel values in n-by-3 array, where n is the number of points you selected. impixel removes the stars used to indicate selected points.

pixels =

255	255	255
255	255	255
117	110	94
255	255	255

Relationship of Pixel Values to Display Range

The Adjust Contrast tool accomplishes this distinction stretching by modifying the CLim property of the axes object that contains the image. The CLim property controls the mapping of image pixel values to show intensities.

By default, the Image Viewer sets the CLim property to the default display range according to the data type. for instance, the display range of an image of class uint8 is from zero to 255. After you use the Adjust Contrast tool, you change the contrast within the image by ever-changing the display range that affects the mapping between image pixel values and also the black-to-white range. You produce a window over the range that defines that pixel within the image map to the black within the display range by shrinking the range from the bottom up.

Contour Plot of Image Data

A contour may be a path in an image on that the image intensity values are equal to a constant. You'll create a contour plot of the data in a grayscale image using imcontour. This function is comparable to the contour function in MATLAB; however it automatically sets up the axes therefore their orientation and ratio match the image. To label the level of the contours, use the clabel function.

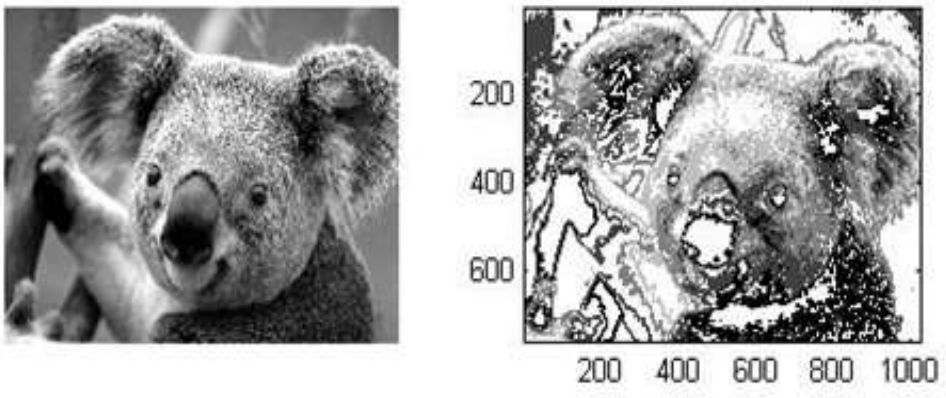
Create Contour Plot of Image data

This example shows the way to create a contour plot of an image.
Read grayscale image and display it. The example uses an example image of koala.

```
img = imread('koala.jpg');
imshow(I)
```

Create a contour plot of the image using imcontour.

```
imcontour(gray_img)
```



XI. DISPLAY COLORS

The number of bits per screen pixel determines the display's screen bit depth. The screen bit depth determines the screen color resolution, which is what percentage distinct colours the display will produce.

Most computer displays use eight, 16, or twenty four bits per screen pixel. Looking on your system, you would possibly be able to opt for the screen bit depth you would like to use. In general, 24-bit display mode produces the simplest results. If you wish to use a lower screen bit depth, 16-bit is usually preferred to 8-bit. However, detain mind that a 16-bit display has sure limitations, such as

• An image might need finer gradations of color than a 16-bit display will represent. If a color is untouchable, MATLAB uses the closest approximation.

• There are solely thirty-twoof grey available. If you're operating primarily with grayscale images, you would possibly regain display results using 8-bit show mode, that provides up to 256 shades of grey to determine the bit depth of your system's screen, enter this command at the MATLAB prompt [13].

```
get(0,'ScreenDepth')
ans =
32
```

11.1The integer MATLAB returns represent the number of bits per screen pixel:

Value	Screen Bit Depth
8	8-bit displays support 256 colors. An 8-bit display can produce any of the colors available on a 24-bit display, but only 256 distinct colors can appear at one time. (There are 256 shades of gray available, but if all 256 shades of gray are used, they take up all the available color slots.)
16	16-bit displays usually use 5 bits for each color component, resulting in 32 (i.e., 25) levels each of red, green, and blue. This supports 32,768 215) distinct colors (of which 32 are shades of gray). Some systems use the extra bit to increase the number of levels of green that can be displayed. In this case, the number of different colors supported by a 16-bit display is actually 64,536 (i.e. 216).
24	24-bit displays use 8 bits for each of the three-color components, resulting in 256 (i.e., 28) levels each of red, green, and blue. This supports 16,777,216 (i.e., 224) different colors. (Of these colors, 256 are shades of gray. Shades of gray occur where R=G=B.) The 16 million possible colors supported by 24-bit display can render a lifelike image.
32	32-bit displays use 24 bits to store color information and use the remaining 8 bits to store transparency data (alpha channel). For information about how MATLAB supports the alpha channel, see “Add Transparency to Graphics Objects” (MATLAB).

Regardless of the number of colours your system will display, MATLAB will store and process images with terribly high bit depths: 224 colours for uint8 RGB images, 248 colours for uint16 RGB images, and 2159 for double RGB images. These images are displayed best on systems with 24-bit color; however sometimes look fine on 16-bit systems moreover. [14].

XII. CONCLUSION

This paper is an introduction on how to get information from images in Matlab. When working with images in Matlab, there are many things to keep in mind such as loading an image, Image Coordinate Systems, Image Types in the Toolbox, Images Processing on a GPU, Import Image Data from the Workspace into the Image Viewer, Export Image Data from the Image Viewer App to the Workspace, Save Image Data, Image Information tool, Write Image Data to File in Graphics Format, Get Pixel Information in Image Viewer App, Pixel Values, Contour Plot of Image Data, Display Colors and The integer MATLAB returns represent the number of bits per screen pixel . This paper presents some of the commands designed for these operations. Most of these commands require you to have the Image processing toolbox installed with Matlab.Extracting and getting data from images is one of the important things some users need. Through our review, it is clear to us the flexibility and accuracy of the MATLAB program in extracting data from images.

REFERENCES

[1]. Dr. Tim Morris. Image Processing with MATLAB
[2]. http://www.directindustry.com/prod/mathworks/product-12865-130905.html
[3]. https://www.mathworks.com/products/image.html
[4]. https://ww2.mathworks.cn/help/images/index.html
[5]. http://mbmelectrical.info/2018/04/11/matlab-workshop/
[6]. https://www.mathworks.com/help/images/images-in-matlab.html
[7]. https://www.mathworks.com/help/images/image-coordinate-systems.html
[8]. https://www.mathworks.com/help/matlab/creating_plots/working-with-images-in-matlab-graphics.html
[9]. https://www.mathworks.com/help/images/define-world-coordinates-using-xdata-and-ydata-properties.html
[10]. https://www.mathworks.com/help/images/indexed-images.html
https://www.mathworks.com/help/images/image-processing-on-a-gpu.html
[12]. https://www.mathworks.com/help/images/get-image-information-in-image-viewer-app.html
[13]. https://www.mathworks.com/help/images/pixel-values.html
[14]. https://www.mathworks.com/help/images/display-colors-1.html

Ahmed Mohamed Ali Karrar" Get information from the image using MATLAB" American Journal of Engineering Research (AJER), vol.8, no.02, 2019, pp.158-172