|  |
| --- |
| **Experiment 1:**  **Introduction to MATLAB/Octave**    Objective: At the end of this experiment, the student would be able to explain the basics functions/commands of MATLAB/Octave Image Processing and Computer Vision Toolbox.  Instructions for MATLAB/Octave: The student can use the Octave Online available at <https://octave-online.net/> for conducting this experiment. It is recommended to create a user login so as to access your previous files. No seperate installation is necessary and your files can be accessed remotely.  Alternatively, the student can also use MATLAB for completing this experiment. However, a standalone installation of MATLAB is required.  Theory:    1.      Image representation in MATLAB/Octave    Sight/vision is one of the greatest powers of a human being. Our eyes can tell us the shape, size, color of any and everything which comes in front of it. An Image is a 2 D light intensity function *f(x,y)*. A digital image *f(x,y)* is discretized both in spatial coordinates and brightness. It can be considered as a matrix whose row, column indices specify a point in the image and the element value identifies the intensity value at that points. These elements are referred to as pixels. An image in MATLAB/Octave is stored as a 2D matrix (of size *m x n*) where each element of the matrix represents the intensity of light/color of that particular pixel. Hence, for a binary image, the value of each element of the matrix is either 0 or 1 and for a grayscale image each value lies between 0 and 255. A color image is stored as an *m x n x 3* matrix where each element is the RGB value of that particular pixel (hence it’s a 3D matrix). You can consider it as three 2D matrices for red, green and blue intensities.   2.    Different type of images (grayscale, binary)  **Binary Image:** An image that consists of only black and white pixels. Technically these types of images are called as Black and White Image. (Although it makes me sad to break my reader’s heart but till now what you called black and white images have some other technical name).    **Grayscale Image:** In daily  language  what  we  refer  to  as  black-and-white  (as  in  old  photos)  are  actually grayscale. It contains intensity values ranging from a minimum (depicting absolute black) to a maximum (depicting absolute white) and in between varying shades of gray. Typically, this range is between 0 and 255.    **Color Image:** We all have seen this! Such an image is composed of the three primary colors, Red, Green and Blue, hence also called an RGB image.    **RGB  Image:**  All  colors  which  we  see  around  us  can  be  made  by  adding  red,  blue  and  green components in varying proportions. Hence, any color of the world can uniquely be described by its RGB value, which stands for Red, Blue and Green values. This triplet has each value ranging from 0 to 255,  with  0  obviously  meaning  no component  of  that particular  color and  255  meaning  full component. For example, pure red color has RGB value [255 0 0], pure white has [255 255 255], pure black has [0 0 0] and   has RGB value [55 162 170].  3. MATLAB/Octave datatype  By default, MATLAB stores its variable in double precision floating point representation, referred to as ‘*double*’. However, there exists several other datatype such as uint8, int8 etc. As discussed earlier, an 8-bit grayscale image contains grayscale values ranging from 0-255. If we read this image in MATLAB as a matrix ‘*im*’, the datatype of this matrix is by default ‘*uint8*’ for unsigned 8-bit integer. Note that even though the default datatype for MATLAB is ‘*double*’, the image matrix ‘*im*’ is represented as ‘*uint8*’ because it is an 8-bit grayscale image. Therefore, the image matrix datatype depends on the image being read.  In addition to *double, uint8*, there exists several other datatypes such as *logical, uint16*, etc  which is used for the saving respective images.     You can verify the datatype of your image using the function ‘*whos*’.  During the course of this lab, you will learn different operations that can be applied on this image matrix ‘*im*’. Although, you could continue working with ‘*im*’ matrix stored as ‘*uint8*’, it is highly recommended to convert this matrix to double precision. The reason being, the unsigned 8-bit integer (*uint8*) would not be able to store decimal quantities. You can use ‘*im2double*’ for this conversion. For this conversion, there is a significant difference between the function ‘*double*’ and ‘*im2double*’.  The following example illustrates this concept. Let us define a matrix  *f= [-0.5,0.5;0.75,1.5];*  By default, this matrix is stored as '*double*'. We can convert this matrix to *uint8* using two different inbuilt functions *uint8* and *im2uint8*.  *g = uint8(f);*  *g1 = im2uint8(f);*  Note the difference in the values of *g* and *g1*. It is because *im2uint8* performs necessary scaling to recognize data as the valid image data.  Before proceeding further, please snswer the following mulitple choice questions.  1.      Suppose you read a 16- bit grayscale image in MATLAB/Octave, the corresponding datatype of the image matrix will be  A)    uint8                     B) int16                                   C) double        D) uint16    2.      Suppose you read a binary image in MATLAB/Octave, the corresponding datatype of the image matrix will be  A)      int8                        B) logical                                C) double        D) uint16  4.  Brief description of basic functions  If you wish to read an image, you should use the inbuilt function ‘imread’. For instance, the following command reads an image titled ‘toto.jpg’ into your MATLAB workspace.  *im = imread(‘toto.jpg’);*  Note that in this case, this image must be stored in your present working directory of MATLAB. You ‘pwd’ to    know this directory. In case, if you wish to read an image from any other directory, you must specify the entire          path.  *im = imread('D:\test\toto.jpg');*  If you wish to check the dimensions of the image matrix im, you can use the function ‘*size*’.  Note that in Online Octave, you must be singed in to upload the image.  You can use ‘*imshow*’ to display the image. For instance, the following commands displays the image stored in the matrix im  *imshow(im);*  Please answer the following multiple choice question before proceeding further*:*  Suppose you wish to read a grayscale image saved at the following location in your computer: ‘D:\User\test1.png’, which is same as the present working directory of matlab. The following command will read this image into MATLAB workspace;  A) im = imread(test1)                              B). im = imread(test1.png)  C) im  = imread(‘test1.png’)                    D). im = imread(test1,png)  **Exercises:**  1.      Create a grayscale image and display : Create a 8-bit grayscale image of size 128 x 128 where the first 64 rows are pure black and rest 64 rows are pure white. Display this image.    2.      Crop a grayscale image  Suppose that the image created in Question 1 is referred as im1. Crop this image such that only the first 64 rows of im1 is stored in another image matrix im2. Display im2.    3.      Crop and replace a grayscale image  Create a 8-bit grayscale image im3 of size 64 x 128 which is pure gray. Put this image im3 in the first 64 rows of the image im1 (Question 1). Display the final resulting image.      4.      Create a binary image. Create a 256 x 256 binary image where the center pixel is 1 and all others are zero. Display this image. Note that binary image should only contain either 1 or 0. (Hint: Use *logical*) |