LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.01**

**A.1 Aim:**

To explore the fundamentals of Image Processing using basic Matlab functions and basic

mathematical image operations.

**A.2 Prerequisite:**

1. Understanding of fundamental programming functions/commands and environment of Matlab (Refer the Matlab manual),

2. Availability of Soft copy of your Photograph for experiment.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Use following Matlab functions/structures for programming.

clc, clear all, ‘; ’ operator, Imread( ), Imwrite( ), Imshow( ), size( ), resize( ), If statement, While, for loop.

1. Explore the Matlab workspace and understand the matrix representation of an Image
2. Write program segment to Perform following mathematical operations on one’s Photograph.
3. Addition of two images
4. Subtraction of one image with other
5. Multiplication of a value with an Image
6. Division of an Image

**A.4 Theory:**

**Introduction of MATLAB:**

MatLab is a programming language for mathematical operations It provides development environment for mathematical programming. It is very much useful for matrix manipulation hence it is good choice for signal processing and image processing applications.

Some commands and programs in MATLAB:

**[1] Declaring matrices:** In Matlab, there is no need to declare variables; new variables are simply introduced as they are required.

>>A = [1 2 3; 4 5 6; 7 8 9]

The above MATLAB command declares following matrix:



Keep following things in mind while declaring variables/matrices:

* No spaces
* Don’t start with a number
* Variable names are case sensitive

**[2] Populating matrix elements with zeros:**

Matlab also provides a number of functions which can be used to populate a new matrix with particular values. For example to make a matrix full of zeroes we can use the function **zeros(m, n)** which creates an m\*n matrix of zeros as follows:

>>B = zeros(3,3)



**[2] Knowing size of matrix:**

Syntax:

**[rows, cols] = size(A);**

This function gives size of matrix Above command gives result: rows=3, cols=3

**[3] Reading Image file :** We can use following command to read image file:

**myImage=imread(‘File name with path’)**

If name of the image file is test.bmp and if it is in /home/chv folder above commands can be written as:

**myImage=imread(‘/home/chv/test.bmp’)**

The image filename can be given as a full file path or as a file path relative to the Matlab current directory. The current directory can be changed from the main Matlab interface window or by cd (change directory command). The supported file formats include ‘bmp’, ‘gif’, ‘jpg’ and ‘png’.

After giving above command image data is available in **myImage** variable. You can use any variable name.

**[4] Displaying image :** After reading image data using above function, we can display images in Matlab using **imshow** function. This function simply takes the array storing the image values as its only parameter.

Syntax:

**imshow(<variable name>)**

Example:

**imshow(myImage);**

**[5] Knowing size of image in pixels:**

Size of the image in pixels can be found out by following command:

**[Rows, Cols] = size(myImage)**

**[6] Image resizing :** Image resizing can be done by following command:

**imresize(Image,{Parameters});**

For example:

Consider that we read the image in variable myImage using imread function than we can resize the image stored in this variable by following command imresize(myImage,[256,256],’nearest’);

This command will convert image of any size into image of 256x256 using nearest neighbor technique.

[**7]Converting Color image into Grayscale image:** Color image can be converted into Grayscale image by Matlab command rgb2gray.

Example:

myGrayImage=rgb2gray(myImage)

**[8] Flow control in MATLAB:** If statements are simply used to make decisions in Matlab

Syntax:

*if <condition> then*

*<do some work>*

*else*

*<do some other work>*

*end*

**[9] Loops in MATLAB:** There is for loop and while loop in MATLAB.

**While loop**

While loop repeat a piece of work as long as a condition holds true. The while loop in Matlab uses the following syntax:

*while <condition>*

*<perform some work repeatedly…>*

*end*

For loop in MATLAB is very popular for image processing because it is particularly useful for iterating through the members of a matrix.

The MATLAB for loop uses the following syntax:

*for index = <start>:<finish>*

*<Perform some work…>*

*end*

**B. Mathematical operation on Images**

Digital images can be represented in matrix format. All Mathematical operations performed on matrix is possible on digital images. For example **Addition of two images** can be given by:

Eqn:eqnadd1

Or if it is simply desired to add a constant value *C* to a single image then:

Eqn:eqnadd2

**The subtraction of two images** is performed straightforwardly in a single pass. The output pixel values are given by:

Eqn:eqnsub1

Or if the operator computes absolute differences between the two input images then:

Eqn:eqnsub2

Or if it is simply desired to subtract a constant value *C* from a single image then:

Eqn:eqnsub3

**The multiplication of two images** is performed in the obvious way in a single pass using the formula:

Eqn:eqnmul1

Scaling by a constant is performed using:

Eqn:eqnmul2

**The division of two images** is performed in the obvious way in a single pass using the formula:

Eqn:eqndiv1

Division by a constant is performed using:

Eqn:eqndiv2

**A.5 Procedure/Algorithm:**

**A.5.1 TASK 1:**

1. Create new file in matlab

2. Write a program to Read Image file and display the read image file.

3. Execute the program

4. Check the Workspace and get familiarized with newly Image variable created with

Imread () function, Image matrix its content, size, availability of 3 colour planes.

1. Modify the above written program and use following functions on the same image and

observe the changes in workspace.

a. Imwrite b. size c. resize

d. ending each matlab statement with/without ‘;’ semicolon sign.

e. clc, clearall.

5. Modify the above program to experiment the use of following in matlab

a. if statement b. For loop c. While loop

6. Observe the output and complete PART B of lab manual.

7. Save and close the file and name it as **EXP1\_Task1\_your Roll no.m**

**TASK 2:**

1. Create a new matlab file.
2. Read your input images.
3. Add two images and observe the output (both workspace and displayed output) and write your comments.
4. Subtract one image from another image and observe output (both workspace and displayed output) write your comments.
5. Divide one image by 10, 50, 255 and observe output(both workspace and displayed output).
6. Multiply one image by 10, 20, 255 and observe the output(both workspace and displayed output).
7. Modify the above programs to show all input and output images on single output window.
8. Complete PART B of lab manual.
9. Save and close the file and name it as **EXP1\_Task2\_your Roll no. m**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

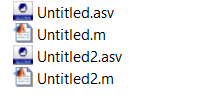
***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.B032 | Name: Naman Garg |
| Class : B Tech CS B | Batch : B2 |
| Date of Experiment: 1-07-2020 | Date of Submission |
| Grade : |  |

**B.1 Software Code written by student:**

**TASK 1:::**

**1. //created a new matlab file by clicking newfile**

****

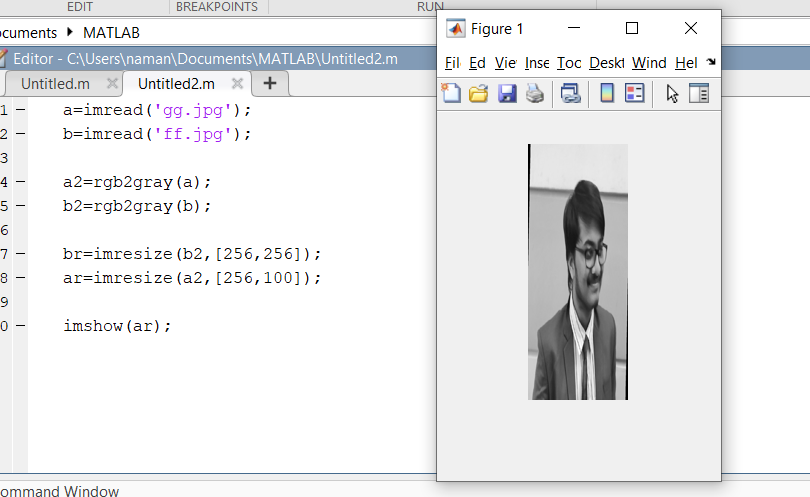
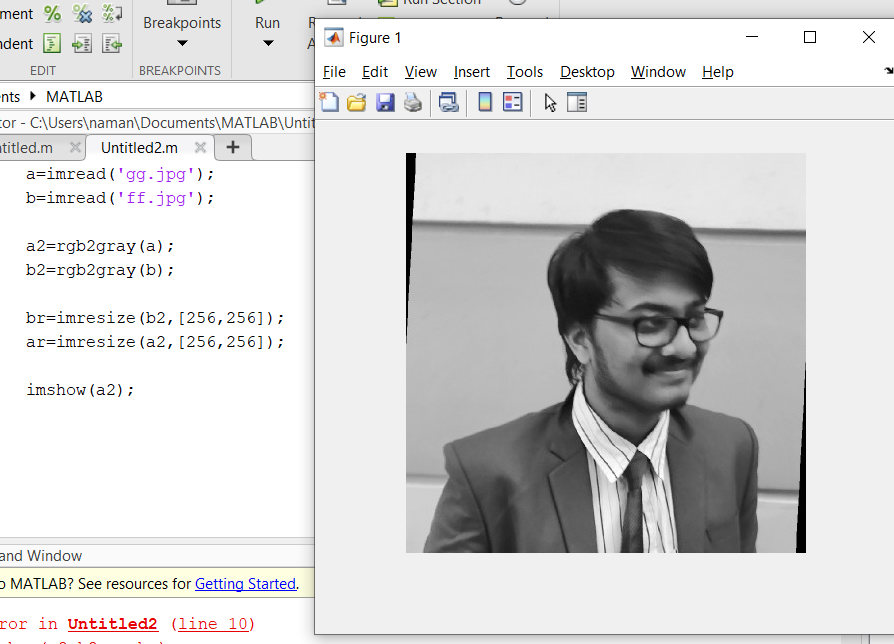
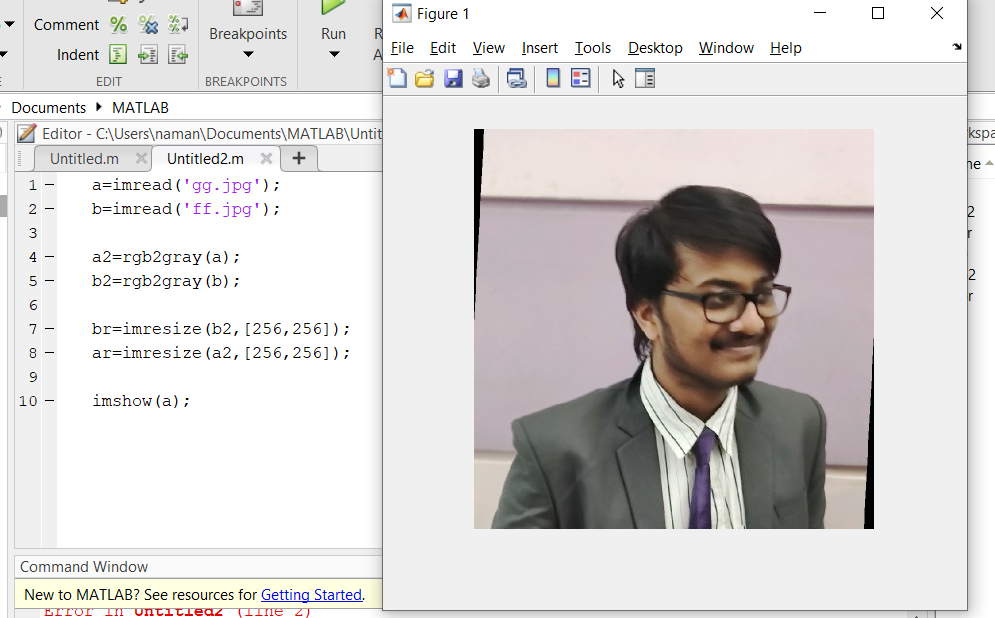
**2. //added 2 images to the workspace and read them.**

a=imread('gg.jpeg');

b=imread('ff.jpeg');



2.



if c<5

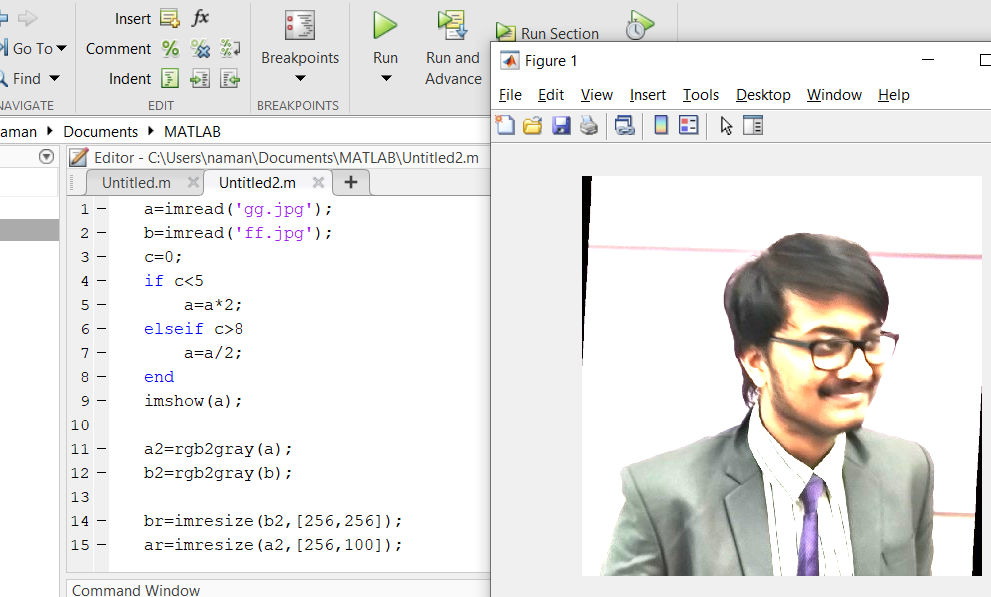
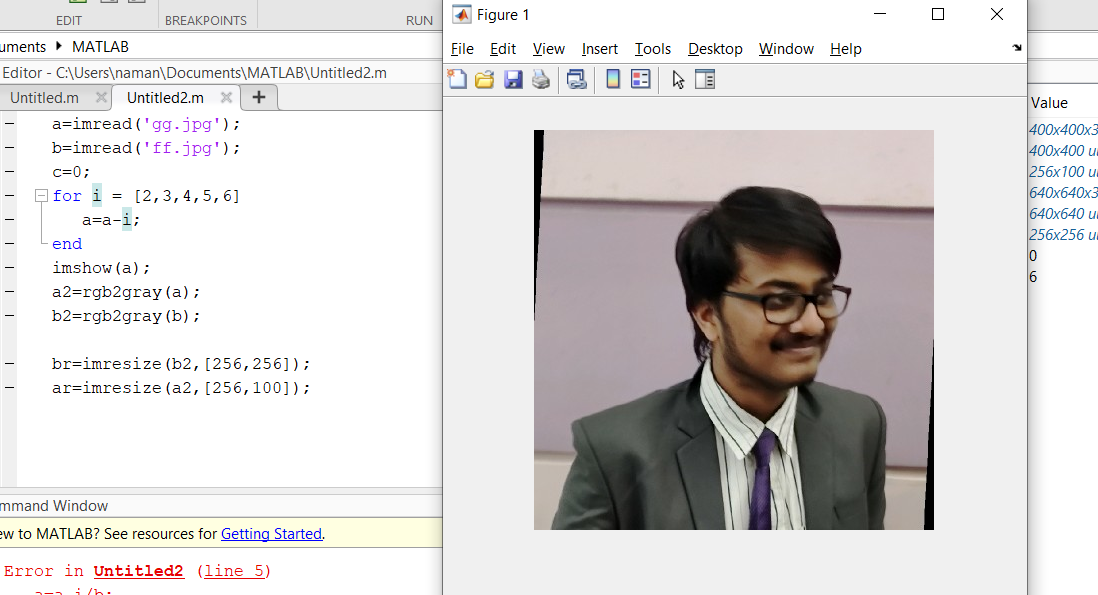
a=a\*2;

elseif c>8

a=a/2;

end

imshow(a);

**  
**

for i = [2,3,4,5,6]

a=a-i;

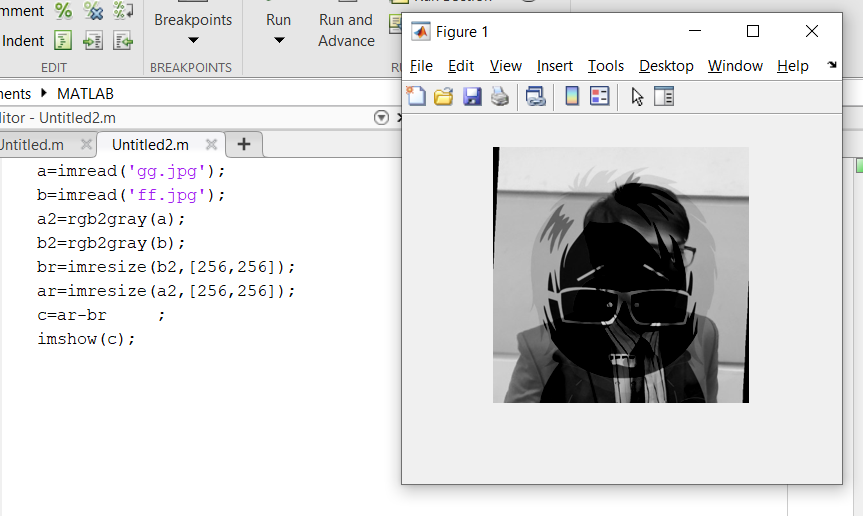
end

imshow(a);

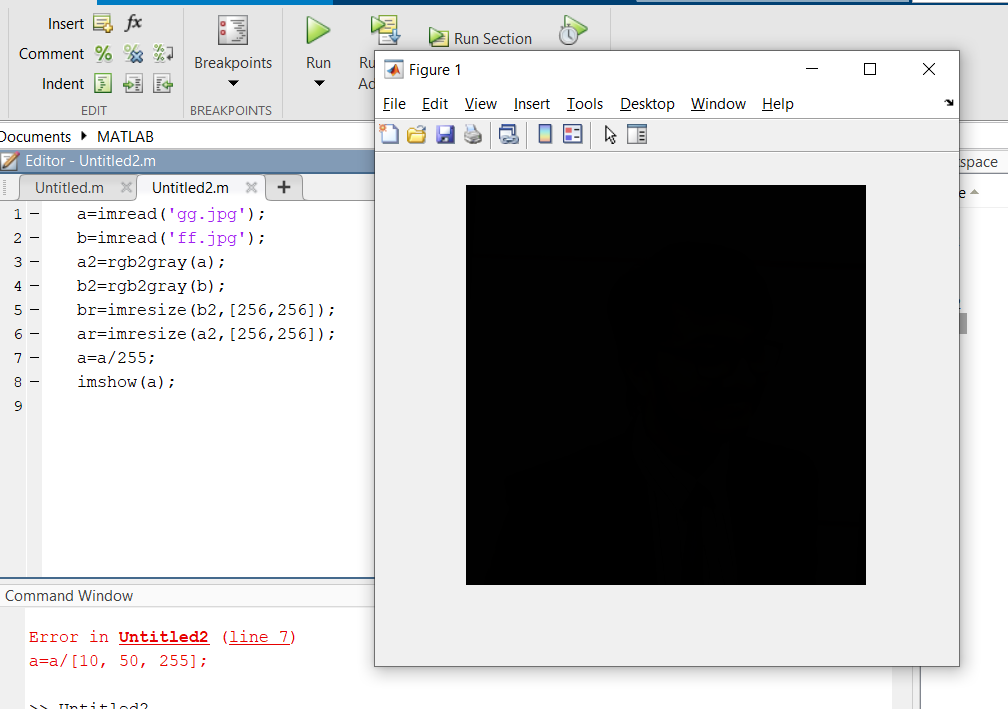
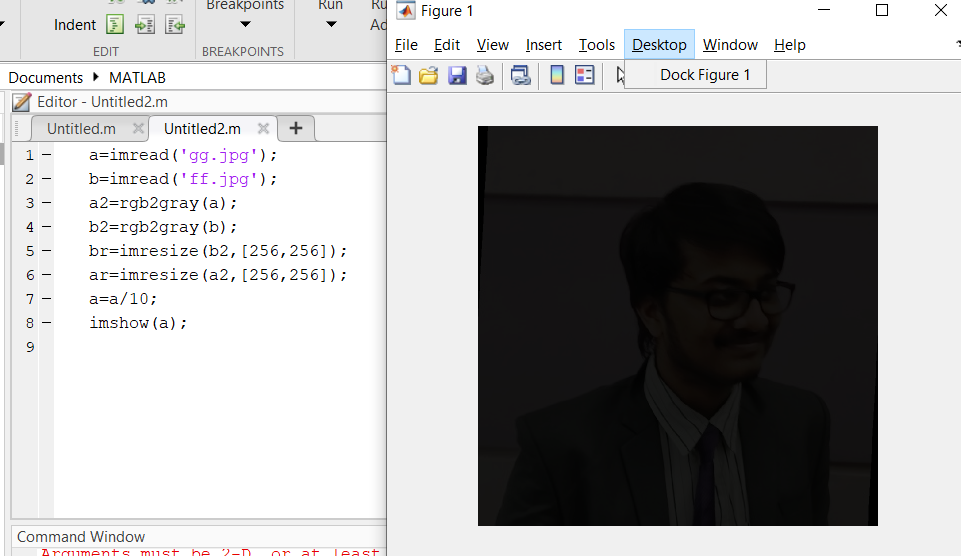
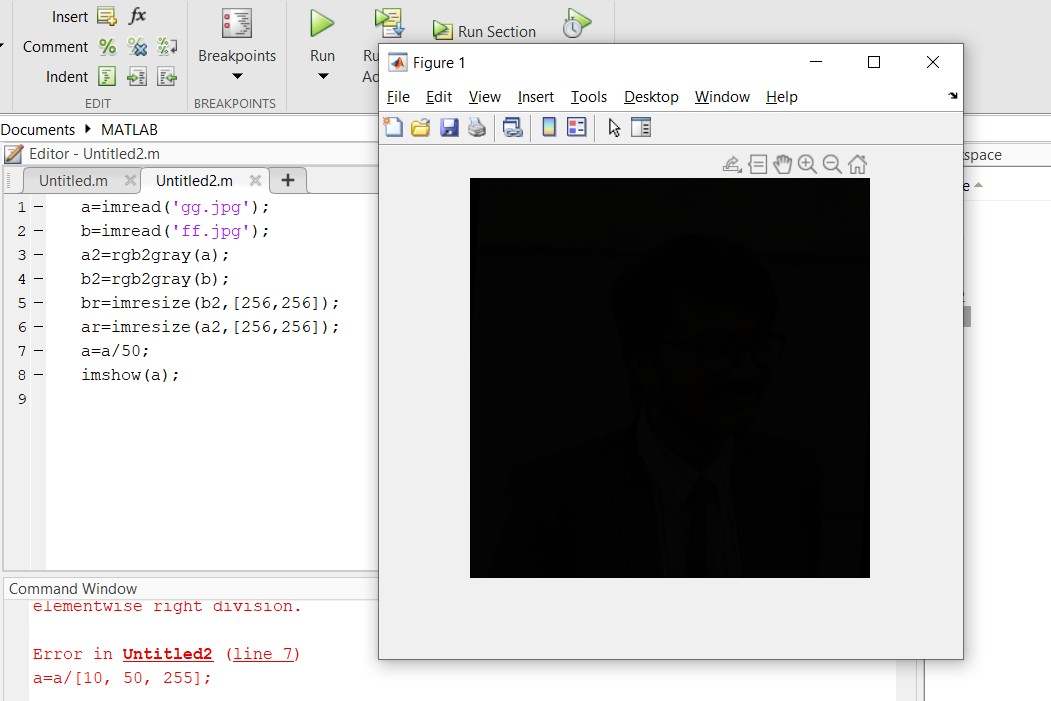
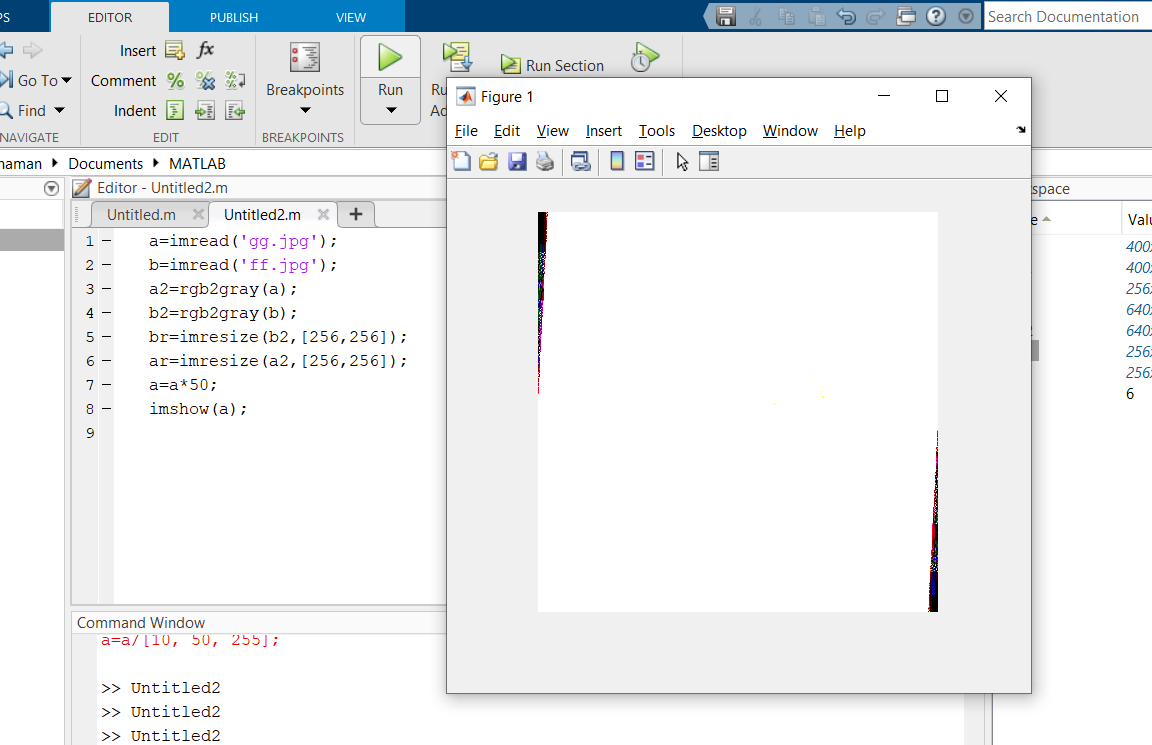
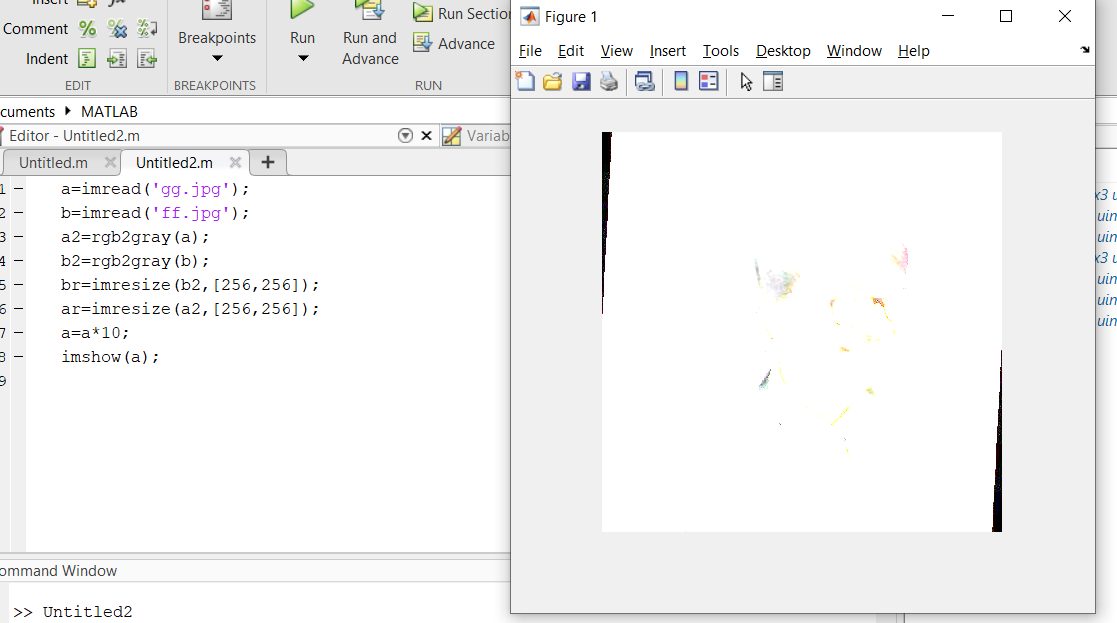
**TASK 2**

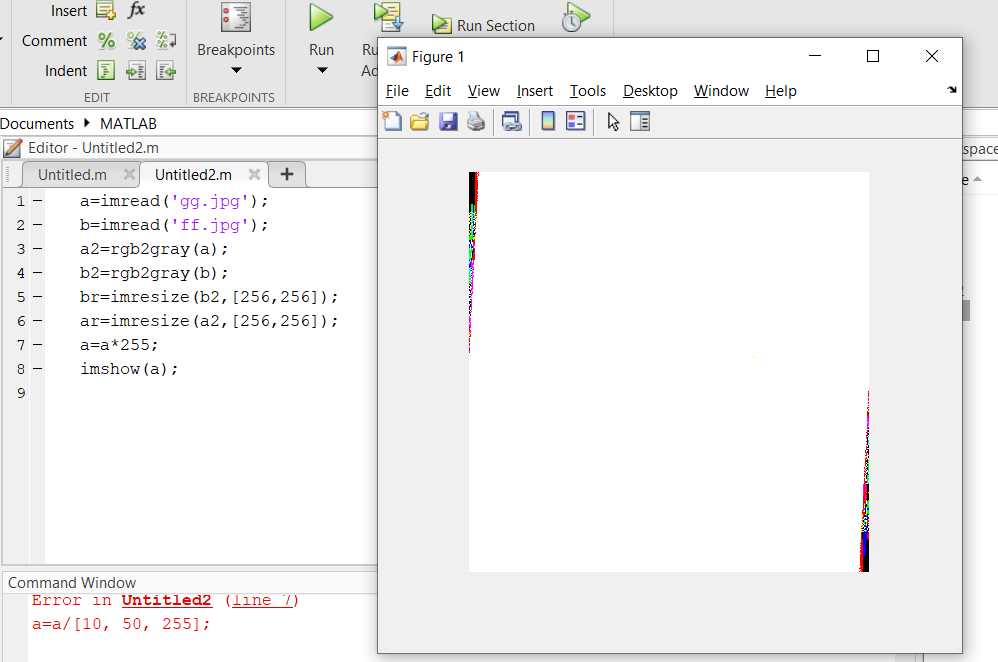
**//subtracting 2 images; the original one and this one::**

****

****

**As we can observe, the images are negated which means that all the whiter parts of the images are turned darker whereas the already darker parts remain unchanged as they were already 0 in value**

**** 

****

**THE image gets darker as the denominator increases.**

**AND THE IMAGE GETS LIGHTER AS THE MULTIPLICAND INCREASES;**

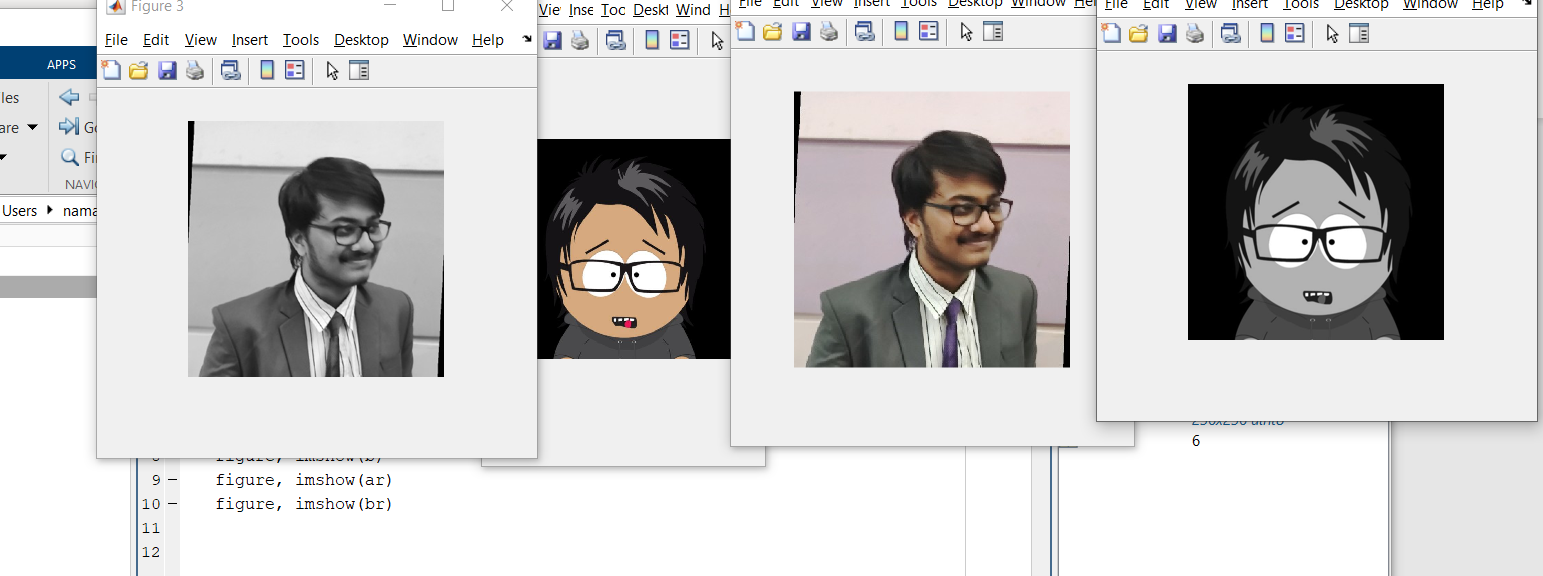
**HOWEVER THE COMPLETELY DARK I.E 0 VLAUE PIXELS ARE UNCHANGED IN THE MULTIPLICATION AS IT WILL RESULT IN 0 ONLY**

imshow(a)

figure, imshow(b)

figure, imshow(ar)

figure, imshow(br)

****

**B.3 Observations and learning:**

I observed that the matlab tools help us automate the image manipulation workflow, for example instead of using photoshop to one-by-one resize/edit photos in a set certain way I can now just write a matlab script that can increase the brightness or resize them to a set predetermined size

Many other applications of matlab image processing might have to do with the fact that, if we have a certain image that we want to find as a subset of another image we can make that part of the image visible and rest completely black then subtract the original image from it

Thus we can figure out if said image is a subset of the original image this method can be used to find trivially doctored photos.

**B.4 Conclusion:**

LEARNT THE USAGE OF:

clc, clear all, ‘; ’ operator, Imread( ), Imwrite( ), Imshow( ), size( ), resize( ), If statement, While, for loop.

FOUND USEFUL USE OF THE MATLAB WORKSPACE.

Learnt how to perform::

Addition of two images

Subtraction of one image with other

Multiplication of a value with an Image

Division of an Image

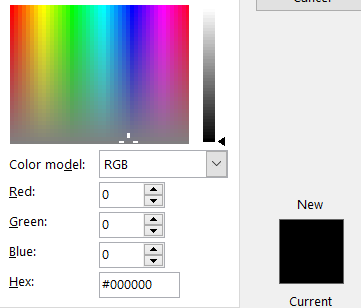
**B.5 Question of Curiosity**

Q1: List out possible real life applications of mathematical operations you have performed on the images

The operators are applied in a pixel-by-pixel way, i.e. the value of a pixel in the output image depends only on the values of the corresponding pixels in the input images.

As discussed above, the images can be subtracted to find sub-set images.

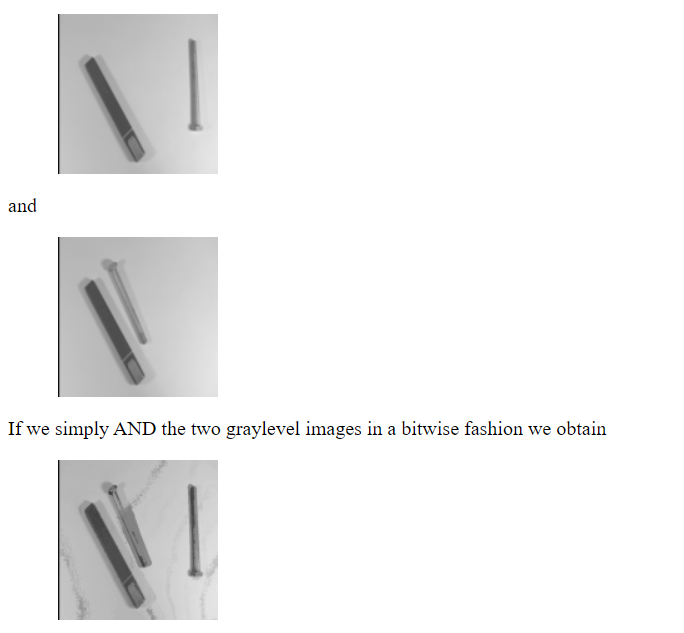
Multiplying or any other way of increasing the pixel values results in a brighter or more saturated picture

Whereas division or reducing the pixel values results in a darker image which makes sense as we observe in the color gradient picker. 

Q2: What output you can get if you perform following logical operations on images: AND/NAND, OR/NOR, XOR/XNOR and logical NOT.

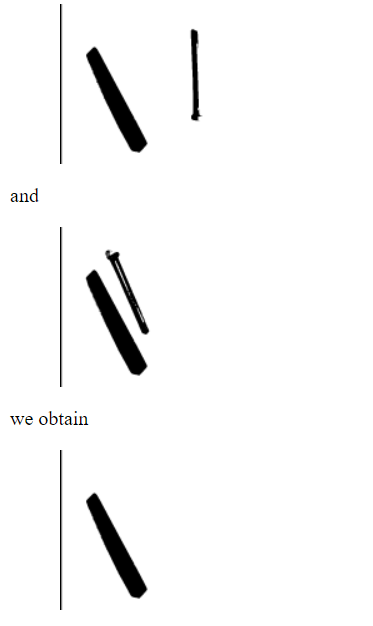
And:

The 2 images will me merged.

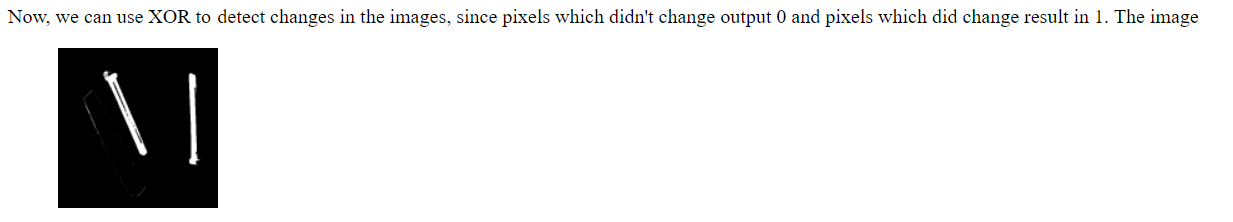


And for nand its similar but first we negate one image

OR:



XOR::



Not::

The image is inverted darker pixels become lighter.