भारतीय सूचना प्रौद्योगिकी संस्थान भागलपुर

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHAGALPUR

(An Institute of National Importance under Act of Parliament)

LAB MANUAL FOR COMPUTER VISION & IMAGE PROCESSING LAB

NAME :- SAGAR SONI

ROLL NO :- 2001092

BRANCH :- ECE

EXPERIMENT – 01

Write an Octave program to do the following change in an Image:

i. Complement of image.

Answer-

1(a). **Aim-**

To create a program to Complement of image using octave software.

Apparatus Used-

- A PC with Octave software
- GNU editor

Theory-

In the complement of a binary image, zeros become ones and ones become zeros. Black and white are reversed.

In the complement of a grayscale or color image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images). The difference is used as the pixel value in the output image. In the output image, dark areas become lighter and light areas become darker. For color images, reds become cyan, greens become magenta, blues become yellow, and vice versa.

Approach:

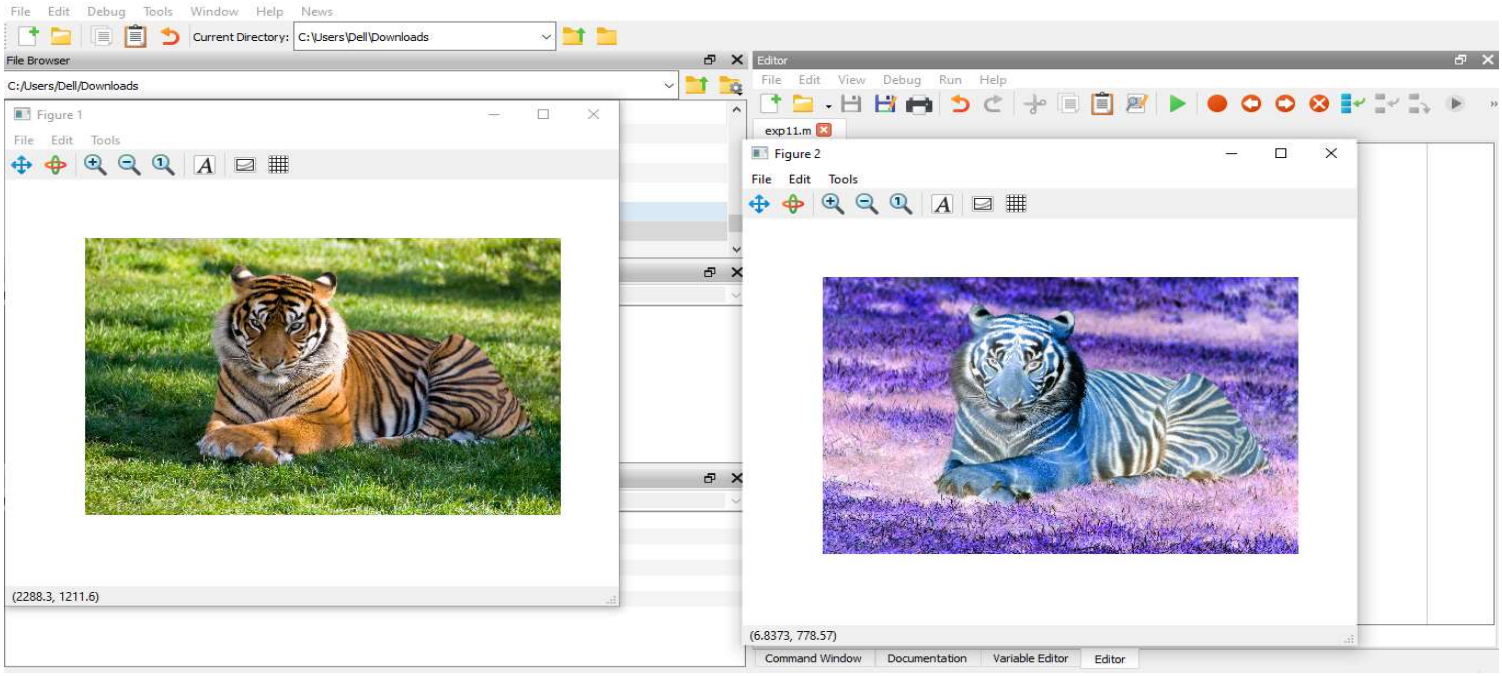
a) Read the image using imread function.
b) Complement the image using imcomplement function.
c) Display the image using imshow.

Procedure-

1) Open Octave software from desktop.
2) In octave we go to editor window an create a script .
3) Write our code in script and save it with path and give it a name in your pc.
4) To run an existing script in Octave, you have to be in the same directory as the script file and type in the name of the file without the . m in Octave.
5) You can check your code error in the command window.

Code and output-

```
% Read the binaryImagei=imread('tiger1.jpeg');

% Display the imageimshow(i);

% Complement the image using functioni=imcomplement(i);

% Display the complemented imagefigure, imshow(i);

% Read the colored imagea=imread('tiger1.jpeg');

% Display the imagefigure, imshow(a);

% Complement the image using functiona=imcomplement(a);

% Display the complemented imagefigure, imshow(a);
```



ii. Flipping upside down and left-right.

1(b). **Aim-**

To create a program to Flipping upside down and left-right using octave software.

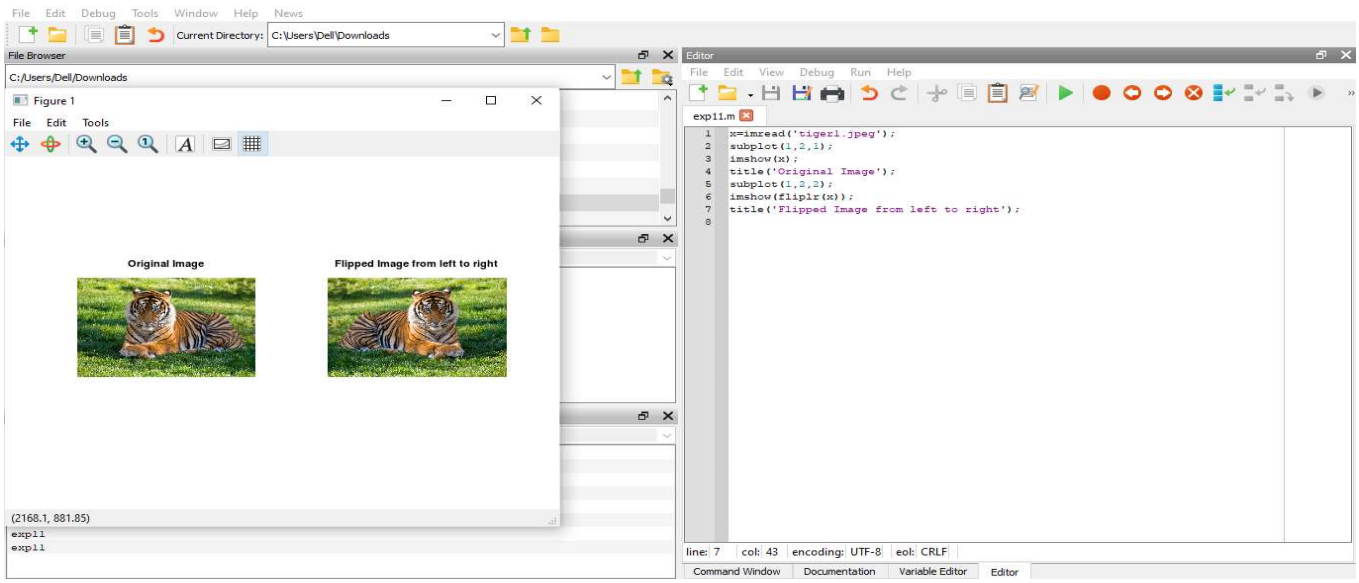
Theory-

A flip (mirror effect) is done by reversing the pixels horizontally or vertically. For instance, for a horizontal flip, the pixel situated at coordinate (x, y) will be situated at coordinate (width - x - 1, y) in the new image.

Code and output-

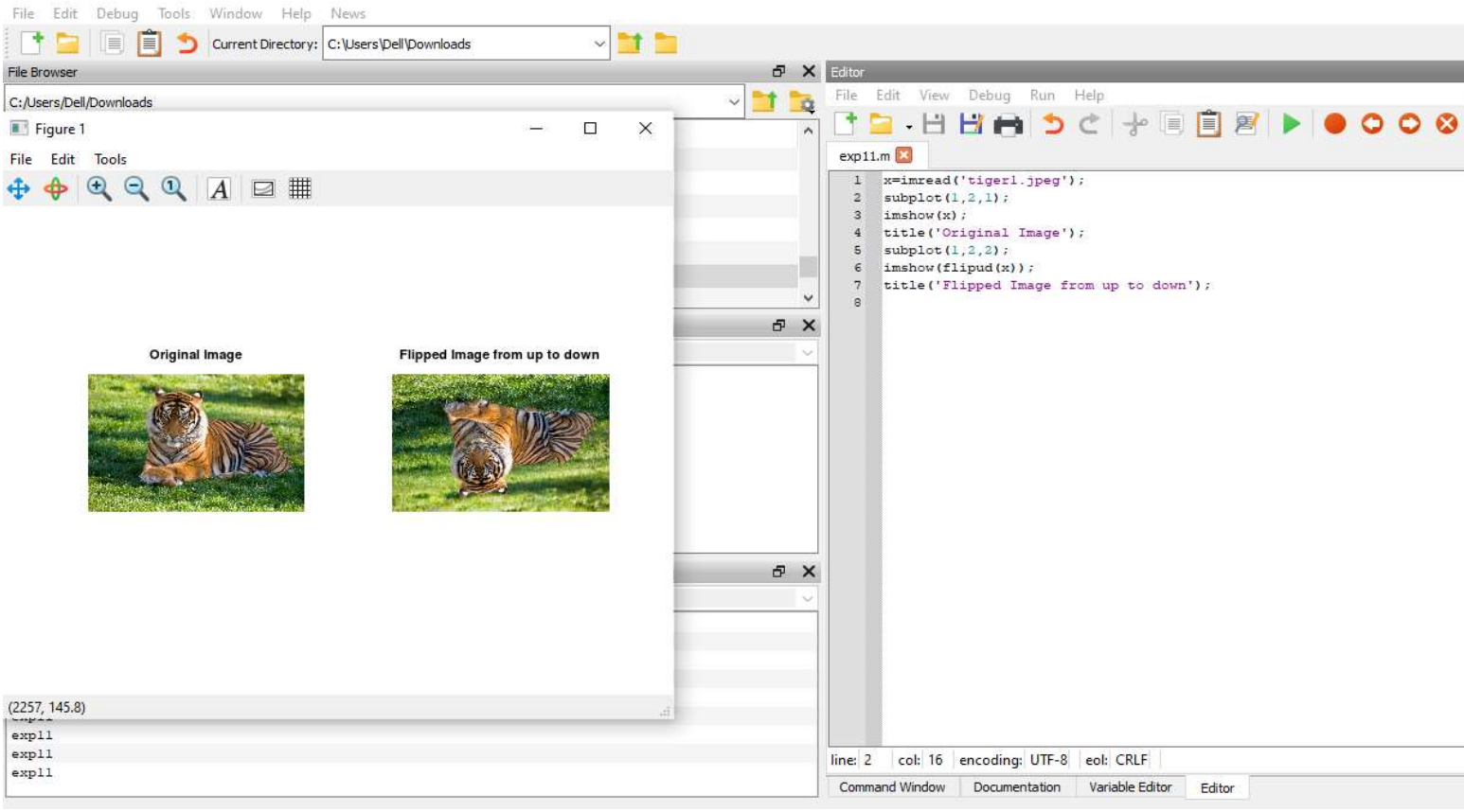
Code for flipping left-right:

```
x=imread('tiger1.jpeg');

subplot(1,2,1);

imshow(x);

title('Original Image');

subplot(1,2,2);

imshow(fliplr(x));

title('Flipped Image from left to right');
```



Code for flipping up-down:

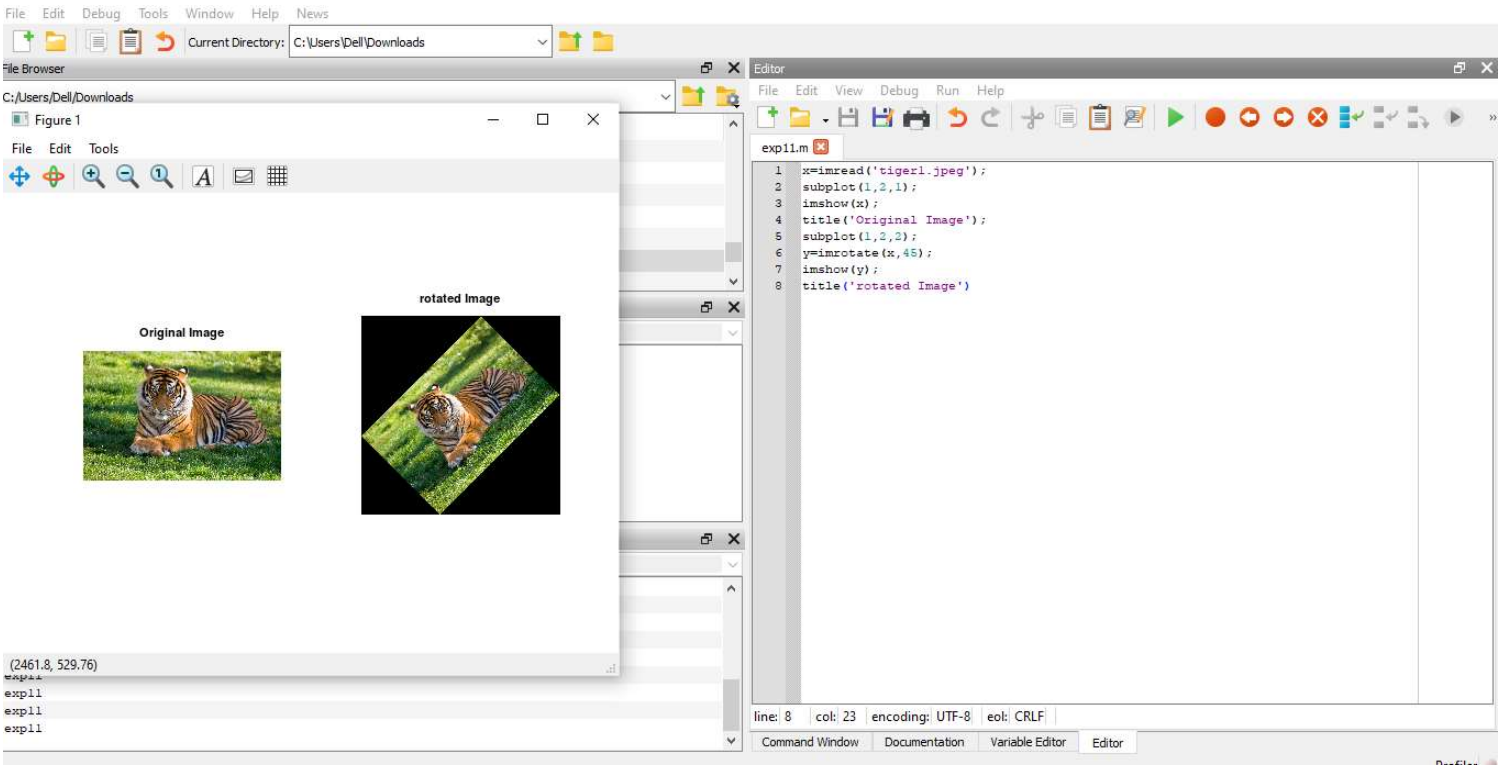
```
warning off
x=imread(‘tiger1.jpeg’);
subplot(1,2,1);
imshow(x);
title('Original Image');
subplot(1,2,2);
imshow(flipud(x));
title('Flipped Image from up to down');
```



iii. Rotation by 45 degrees.

Code and output-

```
x=imread('tiger1.jpeg');
subplot(1,2,1);
imshow(x);
title('Original Image');
subplot(1,2,2);
y=imrotate(x,45);
imshow(y)
title('rotated image')
```



EXPERIMENT – 02

Write an Octave program to perform following actions on Image:

i. RGB to HSV image
ii. HSV to Gray image
iii. RGB to Gray image
iv. RGB to Binary image
v. Change Contrast of original image (Very high & Very Low).

i. RGB to HSV image

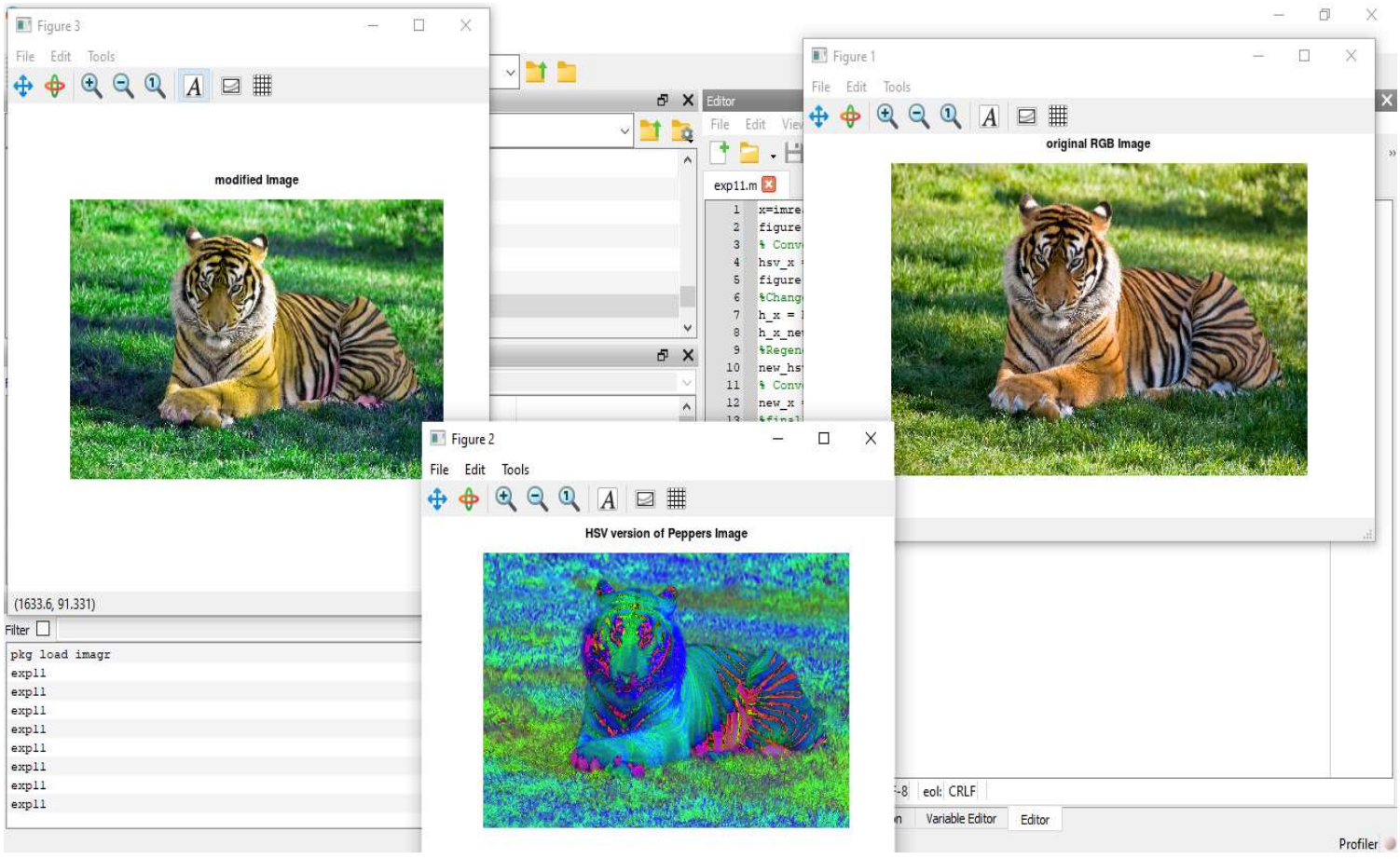
Theory-

HSV = rgb2hsv(RGB) converts the red, green, and blue values of an RGB image to hue, saturation, and value (HSV) values of an HSV image. hsvmap = rgb2hsv(rgbmap) converts an RGB colormap to an HSV colormap.

Code and output-

```
x=imread('tiger1.jpeg');

    figure,imshow(x),title('original RGB Image');

% Convert to HSV

    hsv_x = rgb2hsv(x);

    figure,imshow(hsv_x),title('HSV version of Peppers Image');

%Change  the Hue
```

```
    h_x = hsv_x(:,:,1);
    h_x_new = mod(h_x*1.5,1);  %to make sure range
%Regenerate HSV Image
    new_hsv_x = cat(3,h_x_new,hsv_x(:,:,2),hsv_x(:,:,3));
% Convert HSV back to RGB
    new_x = hsv2rgb(new_hsv_x);
%finally display modified hue image
    figure,imshow(new_x);title('modified Image')
```



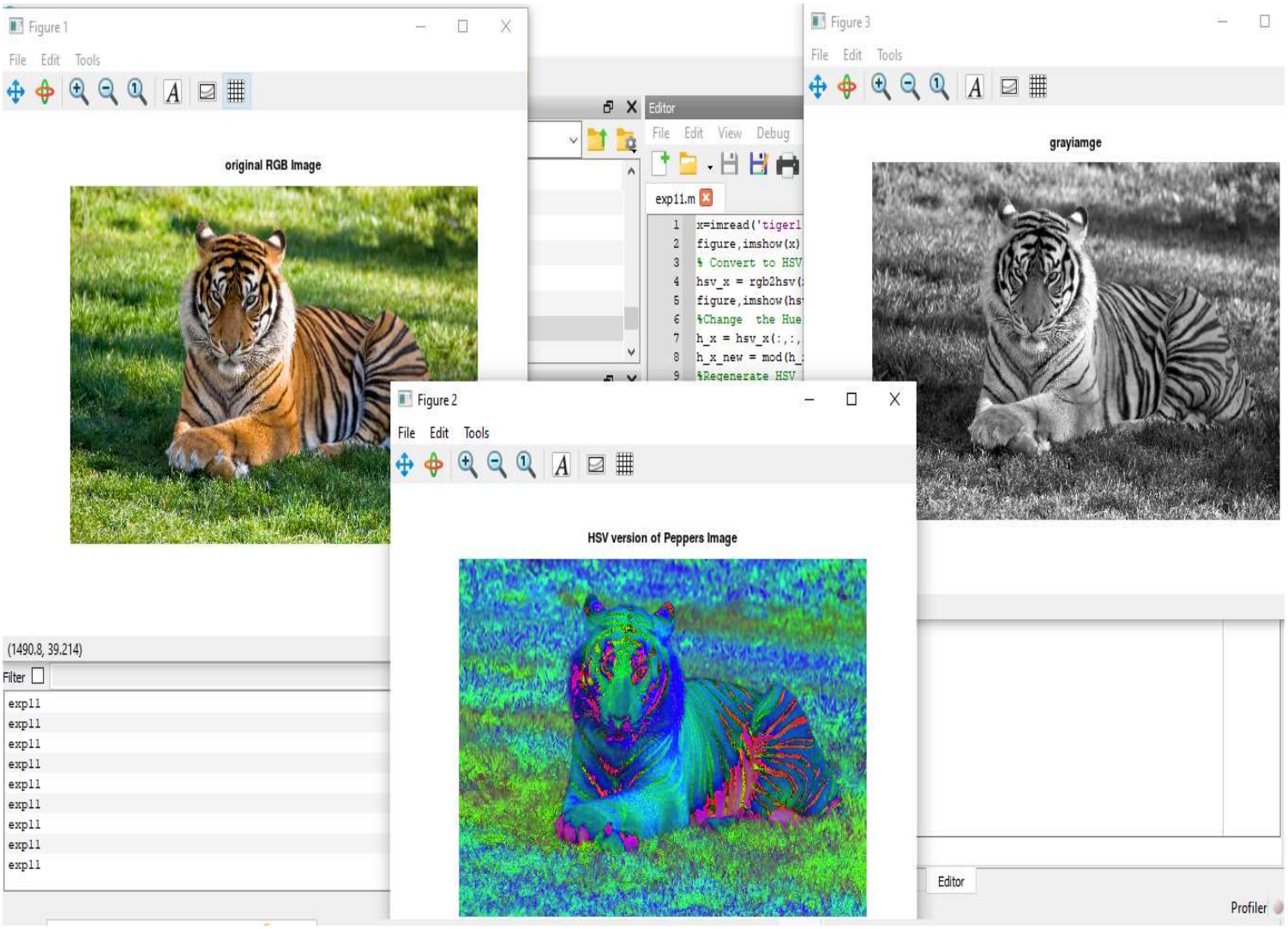
ii. HSV to Gray image

Theory-

HSV (hue, saturation, value) and HSL (hue, saturation, lightness or luminance) are transformations of a Cartesian RGB color space. To convert HSV to grayscale, we first need to convert HSV to RGB and then convert the RGB triple to a grayscale value.

Code and output-

```
x=imread('tiger1.jpeg');
figure,imshow(x),title('original RGB Image');
% Convert to HSV
hsv_x = rgb2hsv(x);
figure,imshow(hsv_x),title('HSV version of Peppers Image');
%Change  the Hue
h_x = hsv_x(:,:,1);
h_x_new = mod(h_x*1.5,1);  %to make sure range
%Regenerate HSV Image
new_hsv_x = cat(3,h_x_new,hsv_x(:,:,2),hsv_x(:,:,3));
% Convert HSV back to RGB
new_x = hsv2rgb(new_hsv_x);
%finally display modified hue image
figure,imshow(new_x);title('modified Image')
p=rgb2gray(new_x);
imshow(p);title('grayimage')
```



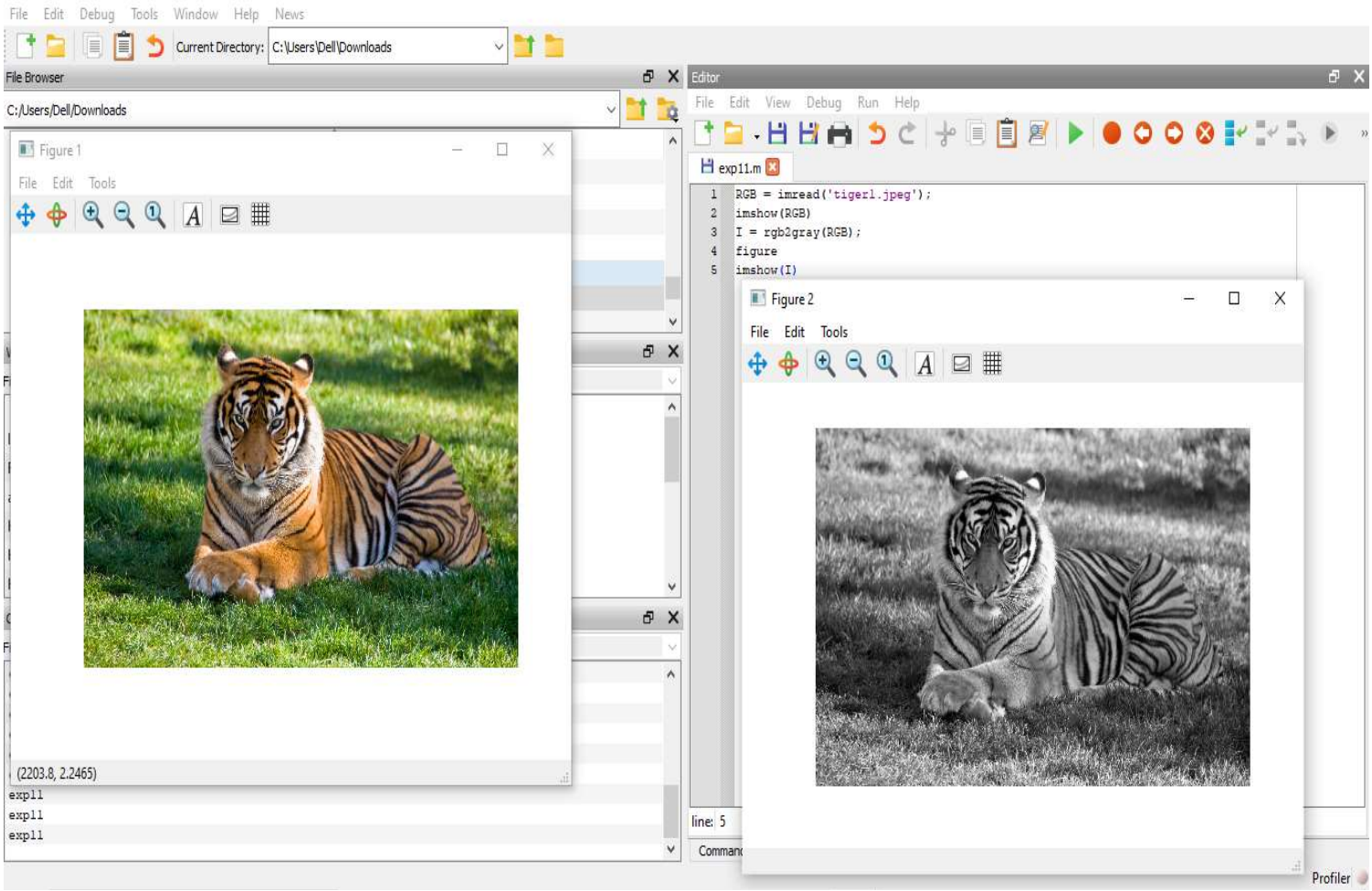
iii. RGB to Gray image

Theory-

I = rgb2gray(RGB) converts the TrueColor image RGB to the grayscale image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

Code and output-

```
RGB = imread('tiger1.jpeg');

imshow(RGB)

I = rgb2gray(RGB);

figure

imshow(I)
```



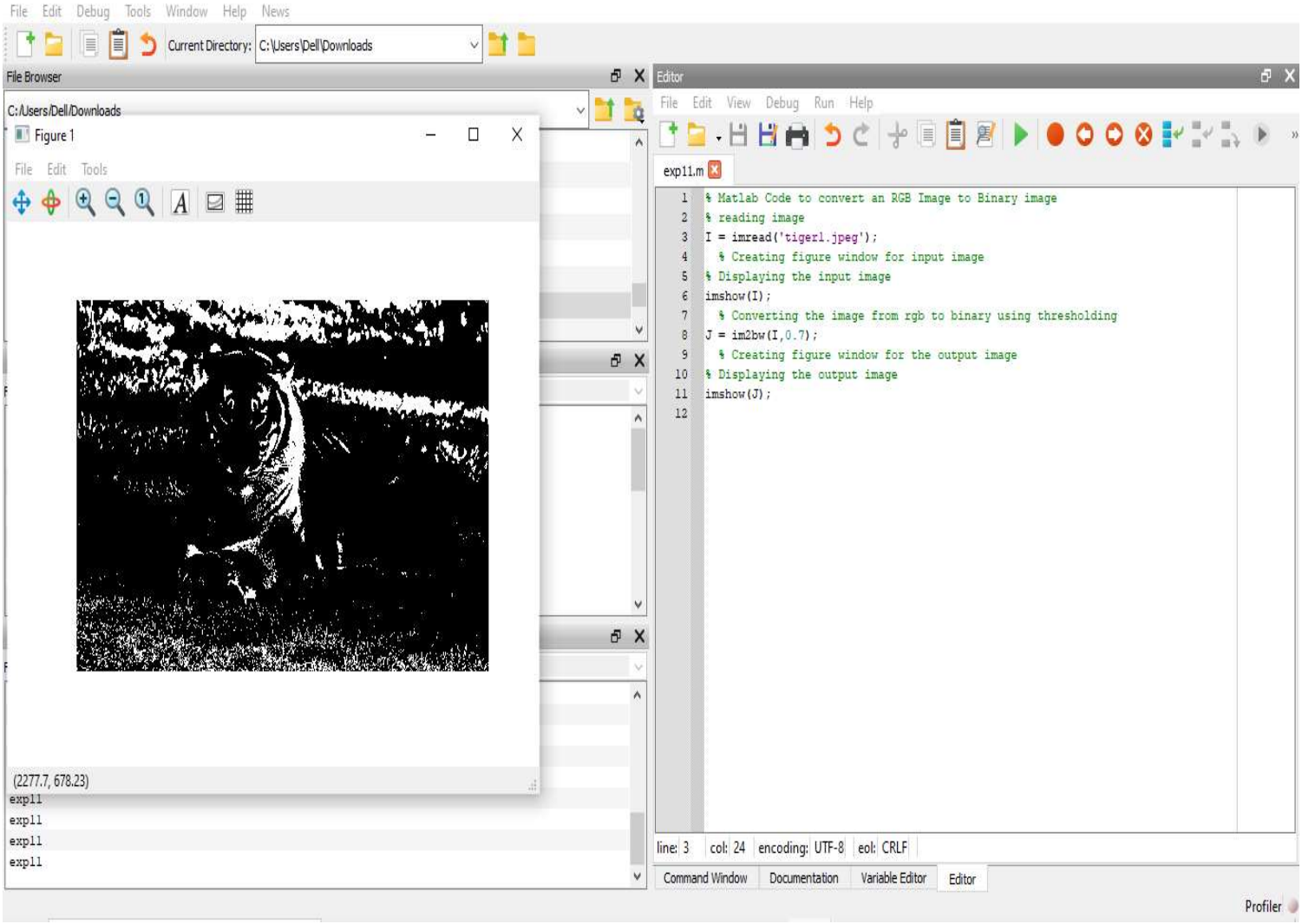
vi. RGB to Binary image

Theory-

Read the RGB Image.
Using the im2bw() function in MATLAB, applying thresholding and classifying the pixel values as 0 or 1 by comparing with the threshold.
Show both the Images together for comparison purposes.

Code and output-

```
% Matlab Code to convert an RGB Image to Binary image

% reading image

I = imread('tiger1.jpeg');

  % Creating figure window for input image

% Displaying the input image

imshow(I);

  % Converting the image from rgb to binary using thresholding

J = im2bw(I,0.7);

  % Creating figure window for the output image

% Displaying the output image

imshow(J);
```



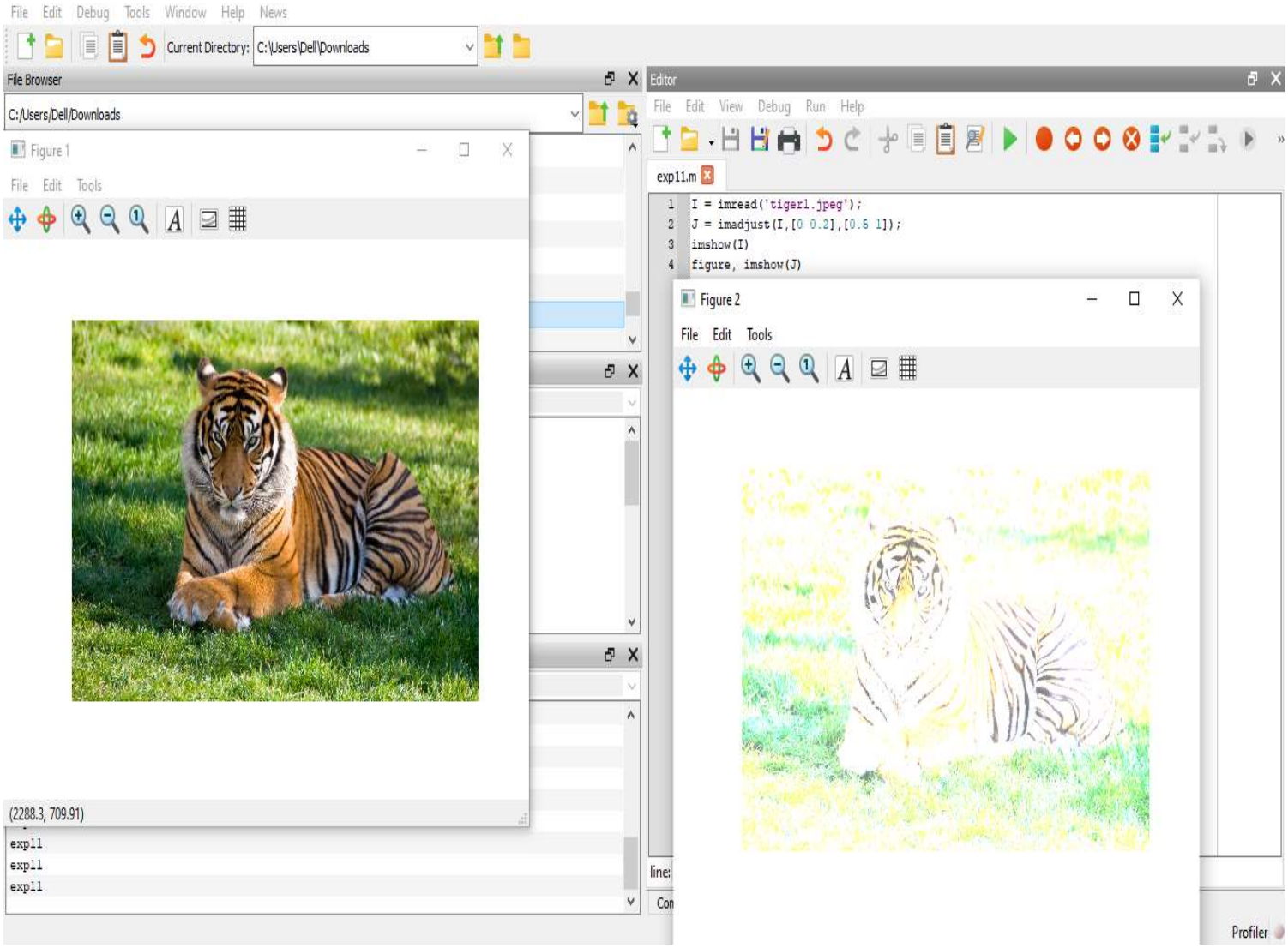
V. Change Contrast of original image (Very high & Very Low).

Theory-

Contrast is the distinction between lighter and darker areas of an image, and it rerefers to making more obvious the objects or details within an image. Increasing contrast on an image will increase the difference between light and dark areas so light areas will become lighter and dark areas will become darker.

Code and output-

```
I = imread('tiger1.jpeg');

J = imadjust(I,[0 0.2],[0.5 1]);

imshow(I)

figure, imshow(J)
```



EXPERIMENT – 03

Write an Octave program for Sampling & Quantization of an image.

Theory-

The sampling rate determines the spatial resolution of the digitized image, while the quantization level determines the number of grey levels in the digitized image.The transition between continuous values of the image function and its digital equivalent is called quantization.The number of quantization levels should be high enough for human perception of fine shading details in the image. The occurrence of false contours is the main problem in image which has been quantized with insufficient brightness levels.

Code and output-

Sampling-

```
im=imread('tiger1.jpeg');

im=rgb2gray(imread('tiger1.jpeg'));

im1=imresize(im, [1024 1024]);

im2=imresize(im1, [1024 1024]/2);

im3=imresize(im1, [1024 1024]/4);

im4=imresize(im1, [1024 1024]/8);

im5=imresize(im1, [1024 1024]/16);

im6=imresize(im1, [1024 1024]/32);
```

figure;imshow(im1)

figure;imshow(im2)

figure;imshow(im3)

figure;imshow(im4)

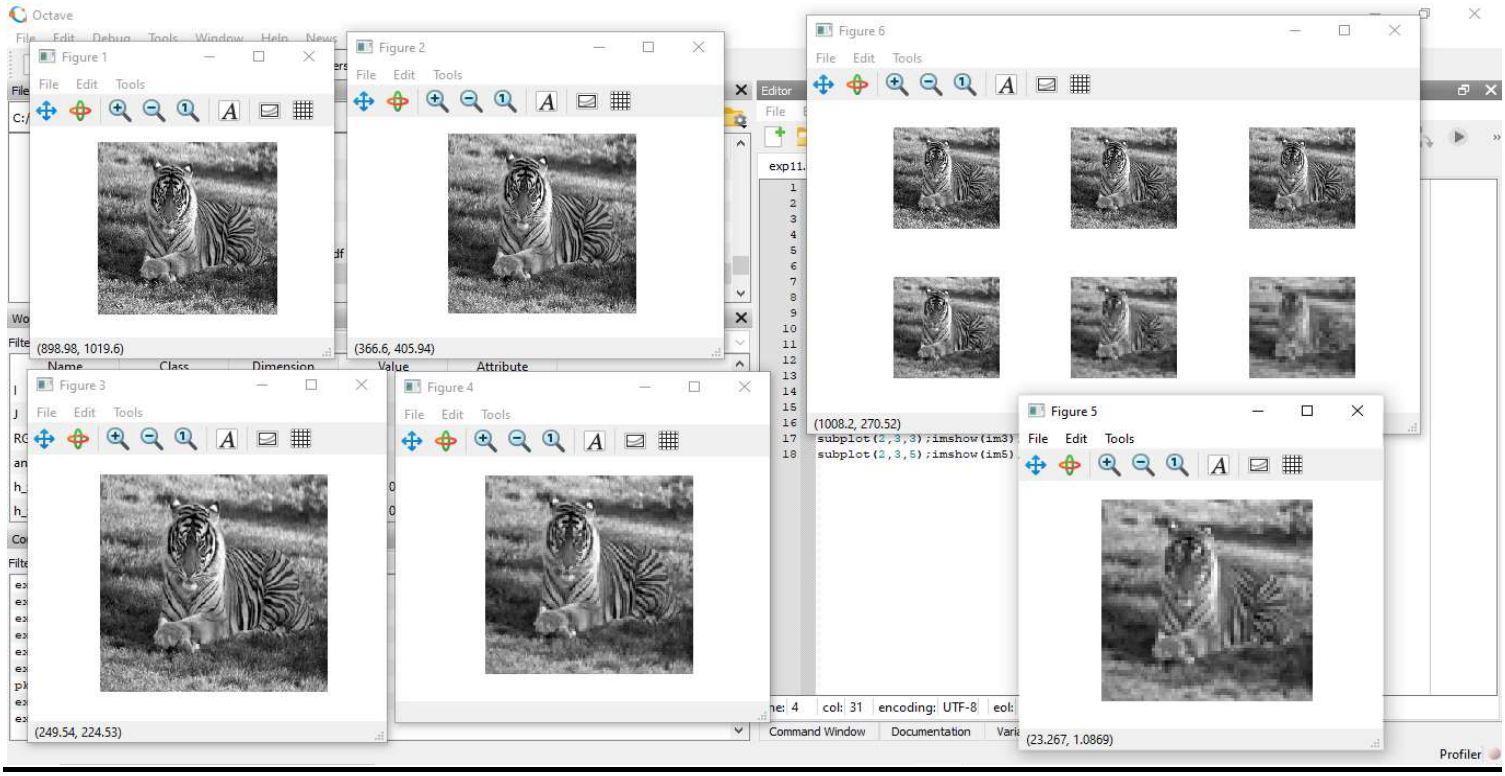
figure;imshow(im5)

figure;

subplot(2,3,1);imshow(im1);subplot(2,3,2);imshow(im2)

subplot(2,3,3);imshow(im3);subplot(2,3,4);imshow(im4)

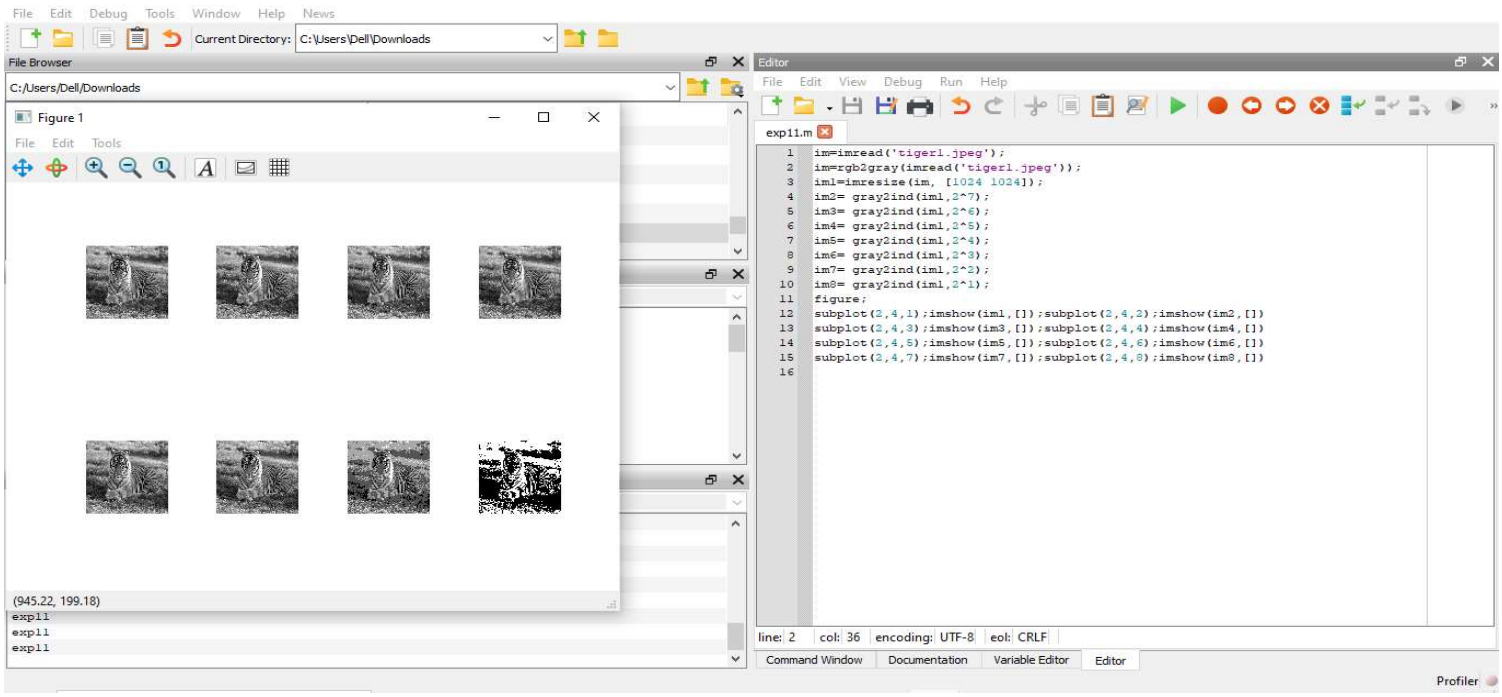
subplot(2,3,5);imshow(im5);subplot(2,3,6);imshow(im6)



Quantization-

im=imread('tiger1.jpeg');

```
im=rgb2gray(imread('tiger1.jpeg'));
im1=imresize(im, [1024 1024]);
im2= gray2ind(im1,2^7);
im3= gray2ind(im1,2^6);
im4= gray2ind(im1,2^5);
im5= gray2ind(im1,2^4);
im6= gray2ind(im1,2^3);
im7= gray2ind(im1,2^2);
im8= gray2ind(im1,2^1);
figure;
subplot(2,4,1);imshow(im1,[]);subplot(2,4,2);imshow(im2,[])
subplot(2,4,3);imshow(im3,[]);subplot(2,4,4);imshow(im4,[])
subplot(2,4,5);imshow(im5,[]);subplot(2,4,6);imshow(im6,[])
subplot(2,4,7);imshow(im7,[]);subplot(2,4,8);imshow(im8,[])
```



EXPERIMENT – 04

Write a program in Octave to implement the following modification of an image:

- Smoothing or Average filter in spatial domain.
- To fill the region of interest for the image.
- To drawing a line and a rectangle on image.

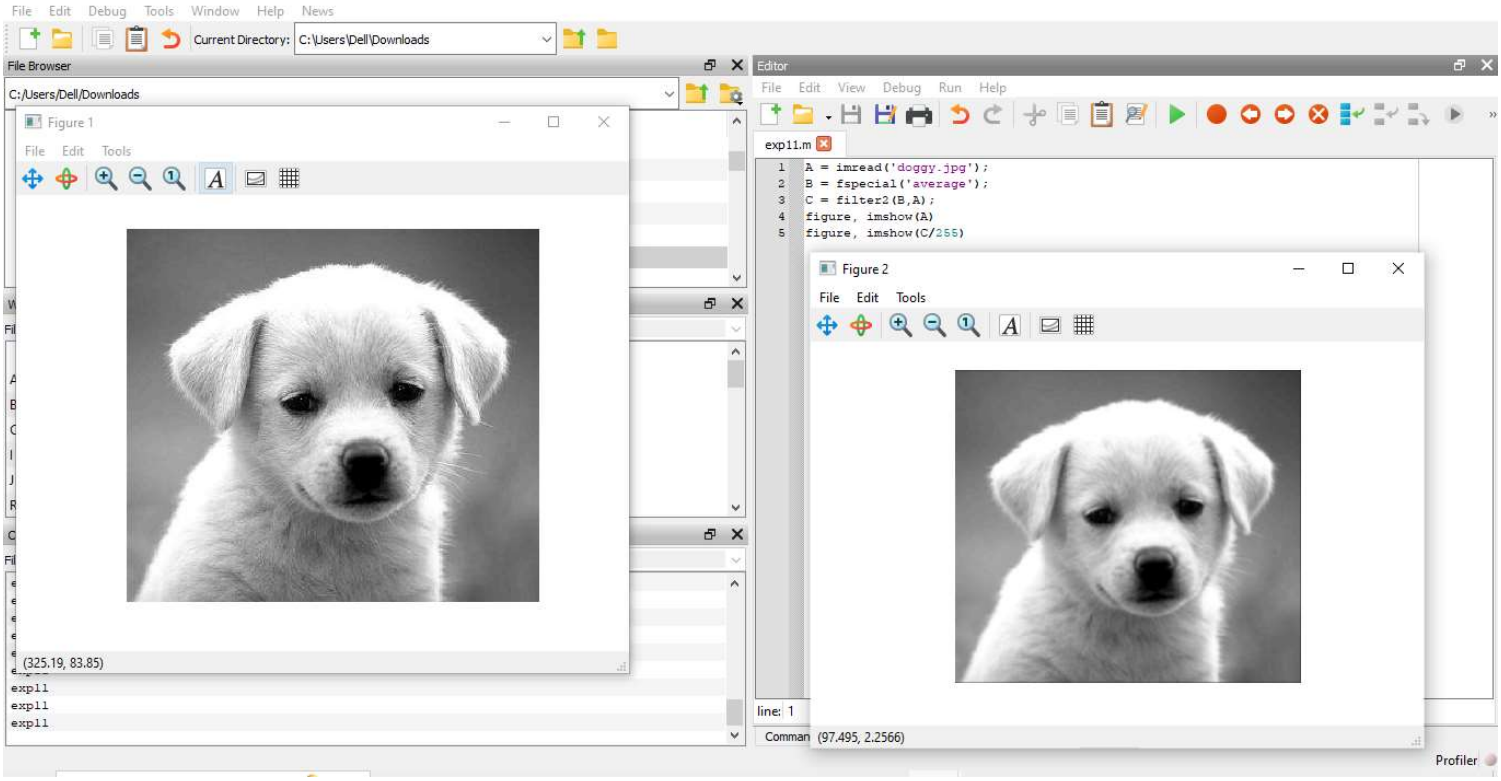
i. Smoothing or Average filter in spatial domain.

Theory-

Image smoothing is a digital image processing technique that reduces and suppresses image noises. In the spatial domain, neighborhood averaging can generally be used to achieve the purpose of smoothing. Smoothing Spatial Filter: Smoothing filter is used for blurring and noise reduction in the image. Blurring is pre-processing steps for removal of small details and Noise Reduction is accomplished by blurring.

Code and output-

```
A = imread('doggy.jpg');
B = fspecial('average');
C = filter2(B,A);
figure, imshow(A),figure, imshow(C/255)
```



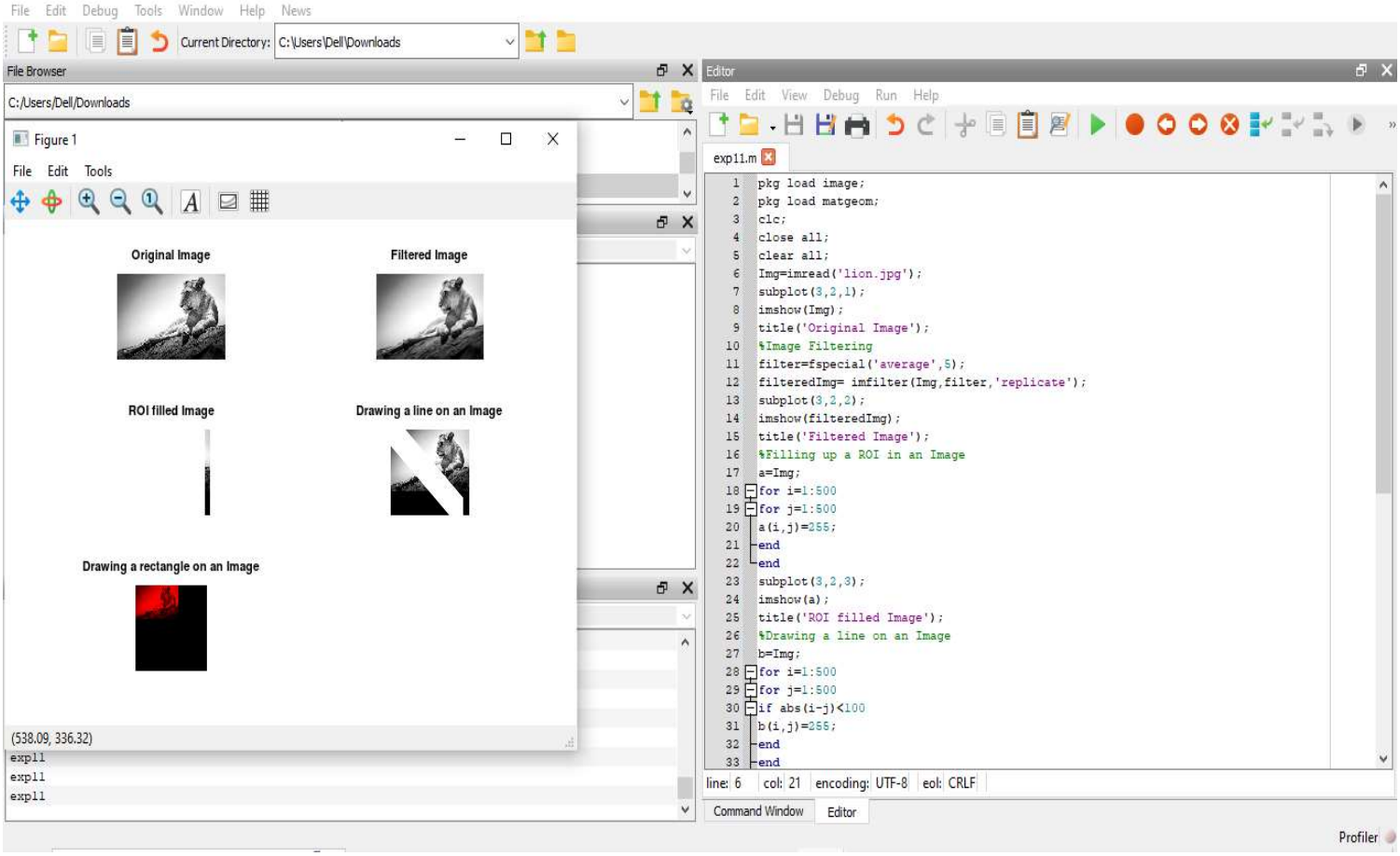
ii. To fill the region of interest for the image.

Theory-

Use the roipoly function to specify the region interactively. Call roipoly and move the pointer over the image. The pointer shape changes to cross hairs . Define the ROI by clicking the mouse to specify the vertices of a polygon.

Code and output-

```
I = imread(lion.jpg');
imshow(I)
x = [222 272 300 270 221 194];
y = [21 21 75 121 121 75];
J = regionfill(I,x,y);
imshow(J)
title('Filled Image with One Fewer Coin')
```



iii. To drawing a line and a rectangle on image.

Theory-

Line draw code-

%# read and display image

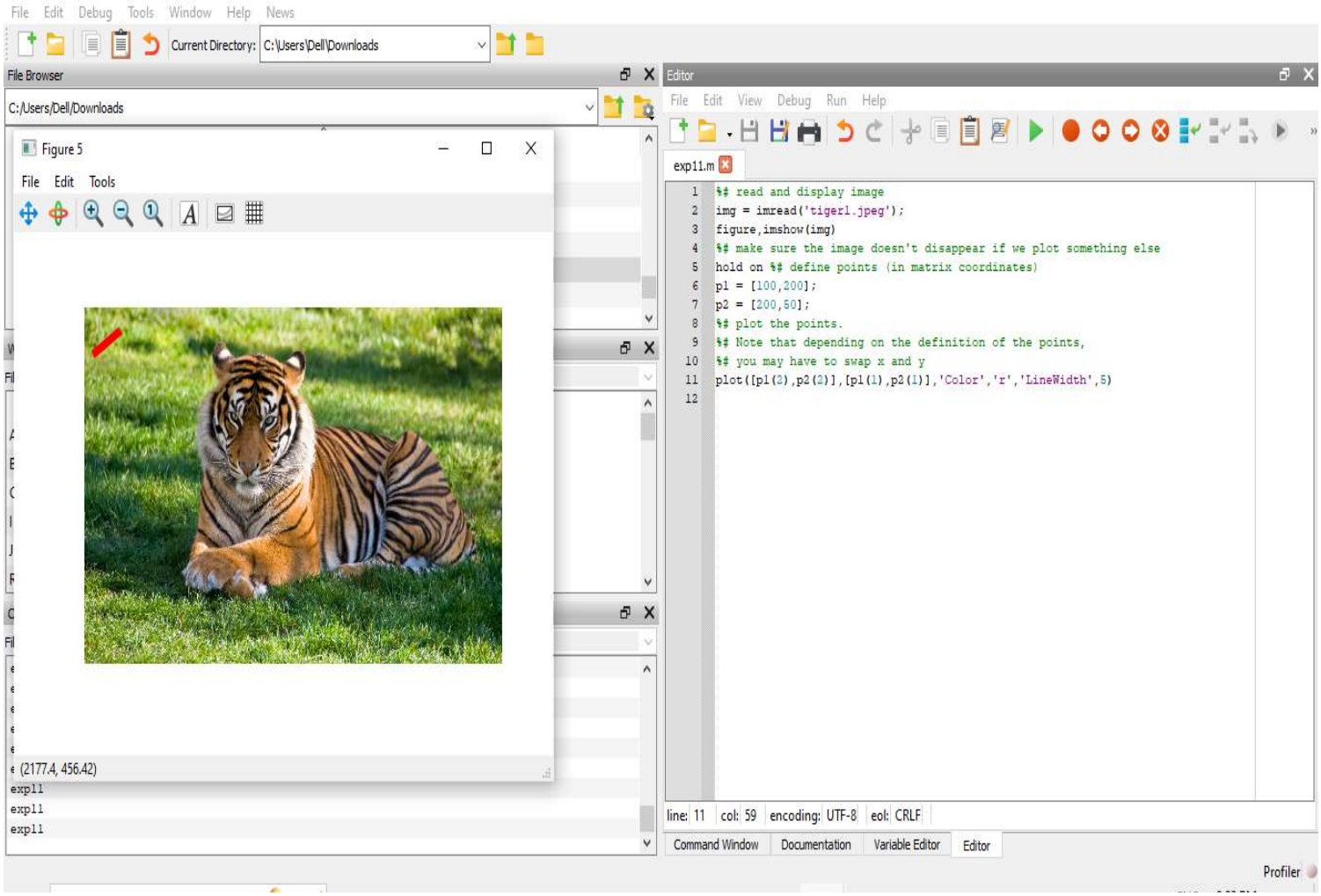
img = imread(‘tiger1.jpeg');

figure,imshow(img)

%# make sure the image doesn't disappear if we plot something else

hold on %# define points (in matrix coordinates)

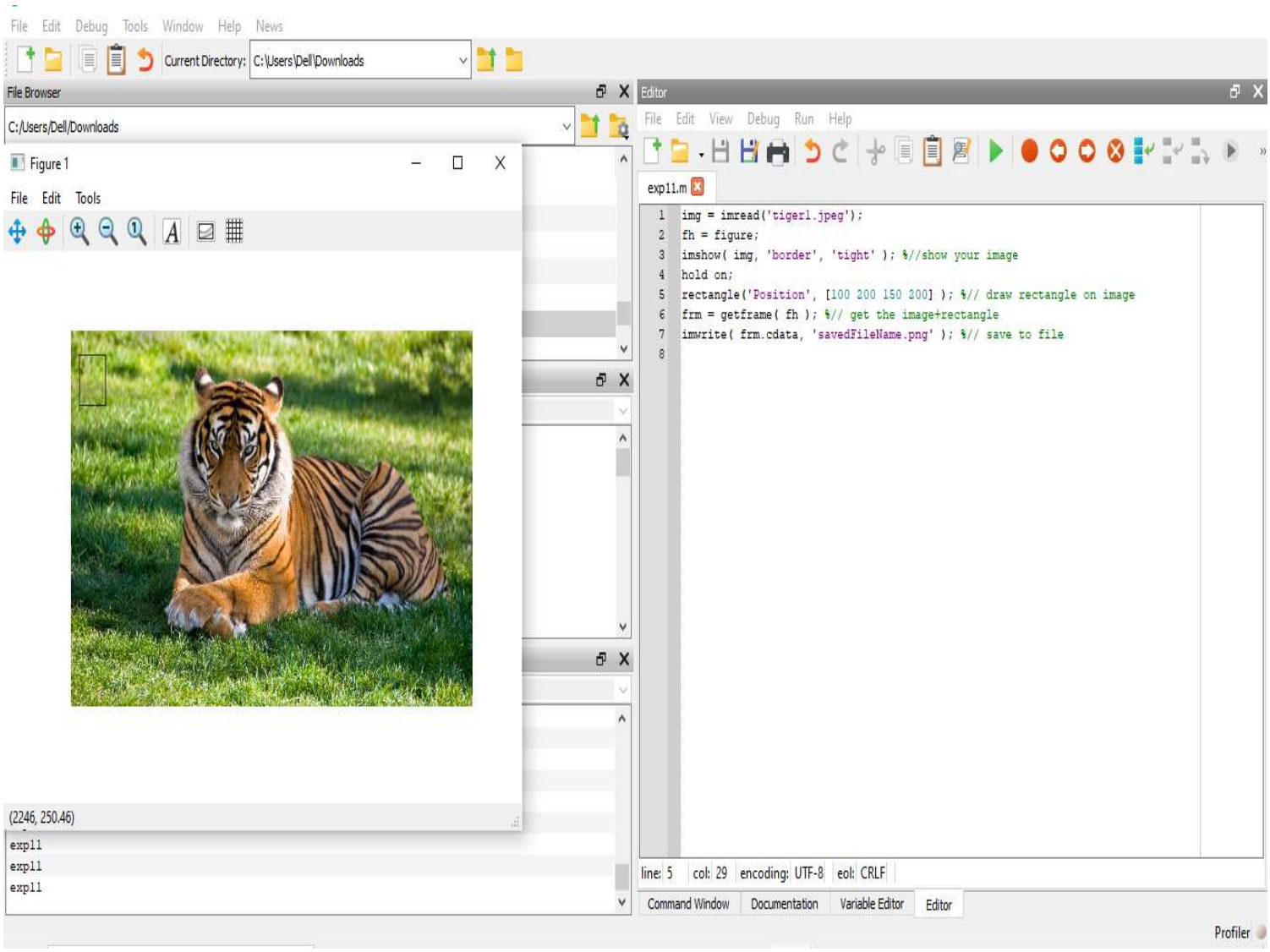
```
p1 = [10,100];

p2 = [100,20];

%# plot the points.

%# Note that depending on the definition of the points,

%# you may have to swap x and y

plot([p1(2),p2(2)],[p1(1),p2(1)],'Color','r','LineWidth',2)
```



Draw rectangle-

Code-

```
img = imread('tiger1.jpeg');

fh = figure;

imshow( img, 'border', 'tight' ); %//show your image

hold on;

rectangle('Position', [50 100 150 200] ); %// draw rectangle on image

frm = getframe( fh ); %// get the image+rectangle

imwrite( frm.cdata, 'savedFileName.png' ); %// save to file
```



EXPERIMENT – 05

Write a program in Octave to perform following morphological operations onanimage

I. opening
II. closing
III. erosion
IV. dilation

Theory-

Types of Morphological operations:

Dilation: Dilation adds pixels on the object boundaries.

Erosion: Erosion removes pixels on object boundaries.

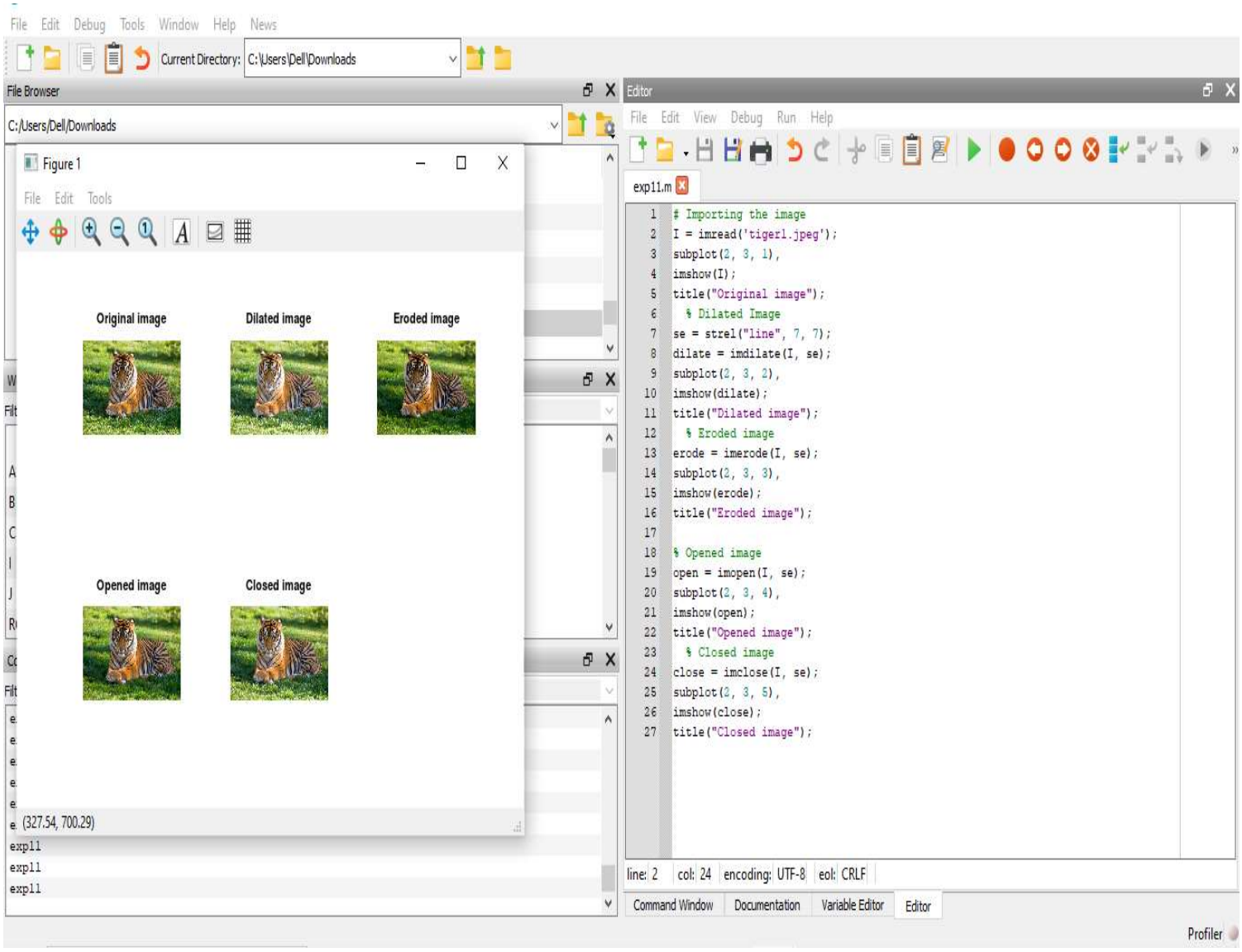
Open: The opening operation erodes an image and then dilates the eroded image, using the same structuring element for both operations.

Close: The closing operation dilates an image and thenerodes the dilated image, using the same structuring element for both operations.

Code and output-

```
# Importing the image

I = imread('tiger1.jpeg');

subplot(2, 3, 1),

imshow(I);

title("Original image");

 % Dilated Image
```

```
se = strel("line", 7, 7);
dilate = imdilate(I, se);
subplot(2, 3, 2),
imshow(dilate);
title("Dilated image");

  % Eroded image
erode = imerode(I, se);
subplot(2, 3, 3),
imshow(erode);
title("Eroded image");

% Opened image
open = imopen(I, se);
subplot(2, 3, 4),
imshow(open);
title("Opened image");
  % Closed image
close = imclose(I, se);
subplot(2, 3, 5),
imshow(close);
title("Closed image");
```


File Edit Debug Tools Window Help News
Current Directory: C:\Users\Dell\Downloads
File Browser
C:/Users/Dell/Downloads
Figure 1
File Edit Tools
Original image
Dilated image
Eroded image
Opened image
Closed image
(327.54, 700.29)
expl1
expl1
expl1
Editor
File Edit View Debug Run Help
exp11.m
1 # Importing the image
2 I = imread('tigerl.jpeg');
3 subplot(2, 3, 1),
4 imshow(I);
5 title("Original image");
6 % Dilated Image
7 se = strel("line", 7, 7);
8 dilate = imdilate(I, se);
9 subplot(2, 3, 2),
10 imshow(dilate);
11 title("Dilated image");
12 % Eroded image
13 erode = imerode(I, se);
14 subplot(2, 3, 3),
15 imshow(erode);
16 title("Eroded image");
17
18 % Opened image
19 open = imopen(I, se);
20 subplot(2, 3, 4),
21 imshow(open);
22 title("Opened image");
23 % Closed image
24 close = imclose(I, se);
25 subplot(2, 3, 5),
26 imshow(close);
27 title("Closed image");
line: 2 col: 24 encoding: UTF-8 eol: CRLF
Command Window Documentation Variable Editor Editor
Profiler

EXPERIMENT – 06

Write a program in Octave to obtain negative of an image andperform histogram equalization of bothoriginal and negative image.

Answer-

Negative of an image

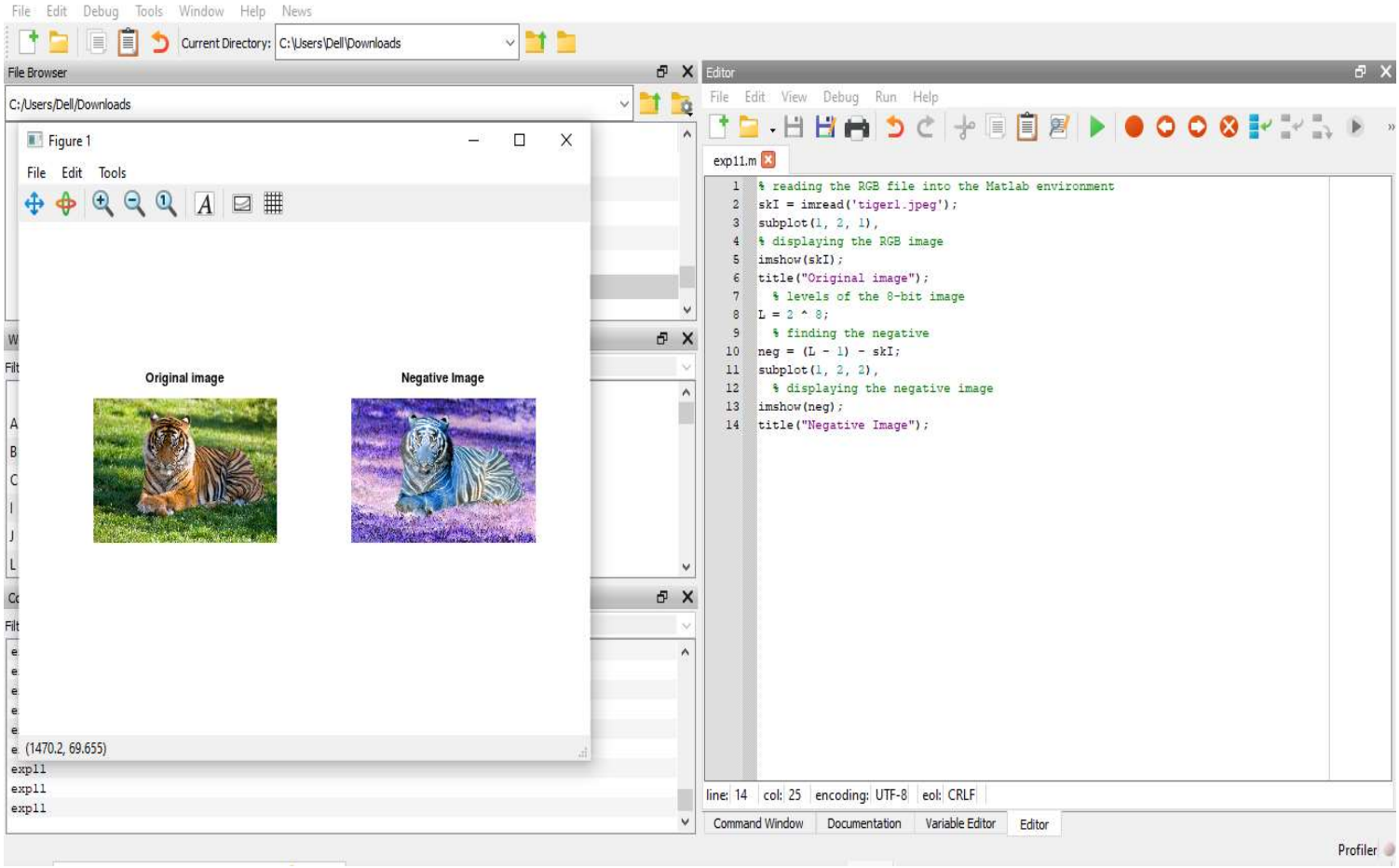
Theory-

The negative of an image is achieved by replacing the intensity ‘i’ in the original image by ‘i-1’, i.e. the darkest pixels will become the brightest and the brightest pixels will become the darkest. Image negative is produced by subtracting each pixel from the maximum intensity value.

Code and output-

```
% reading the RGB file into the Matlab environment

skI = imread(‘tiger1.jpeg');

subplot(1, 2, 1),

  % displaying the RGB image

imshow(skI);

title("Original image");
```

```
  % levels of the 8-bit image
L = 2 ^ 8;
  % finding the negative
neg = (L - 1) - skI;
subplot(1, 2, 2),
  % displaying the negative image
imshow(neg);
title("Negative Image")
```



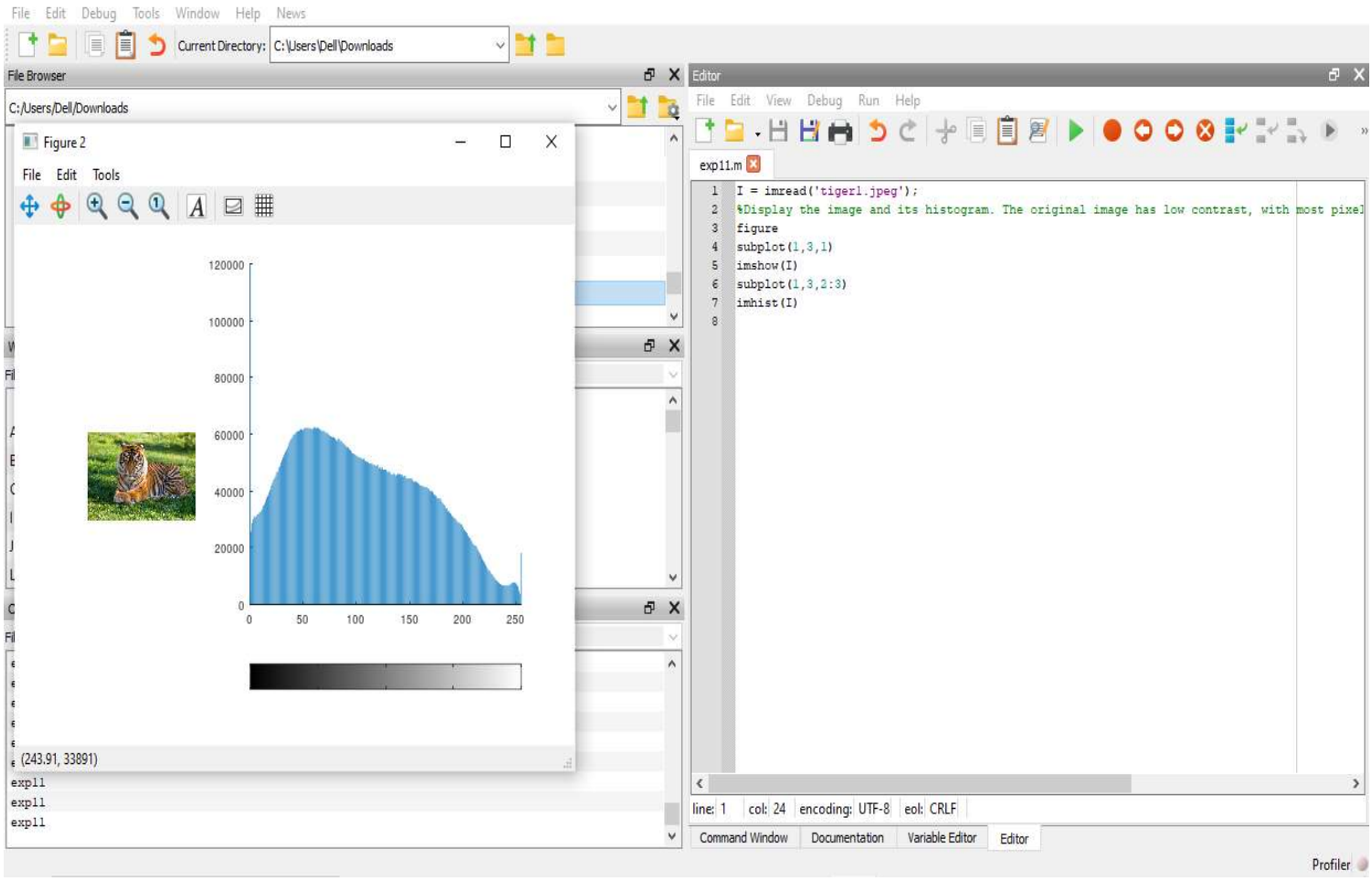
(B) perform histogram equalization of bothoriginal and negative image.

Theory-

histogram equalization involves transforming the intensity values so that the histogram of the output image approximately matches a specified histogram. By default, the histogram equalization function, histeq, tries to match a flat histogram with 64 bins such that the output image has pixel values evenly distributed throughout the range. You can also specify a different target histogram to match a custom contrast

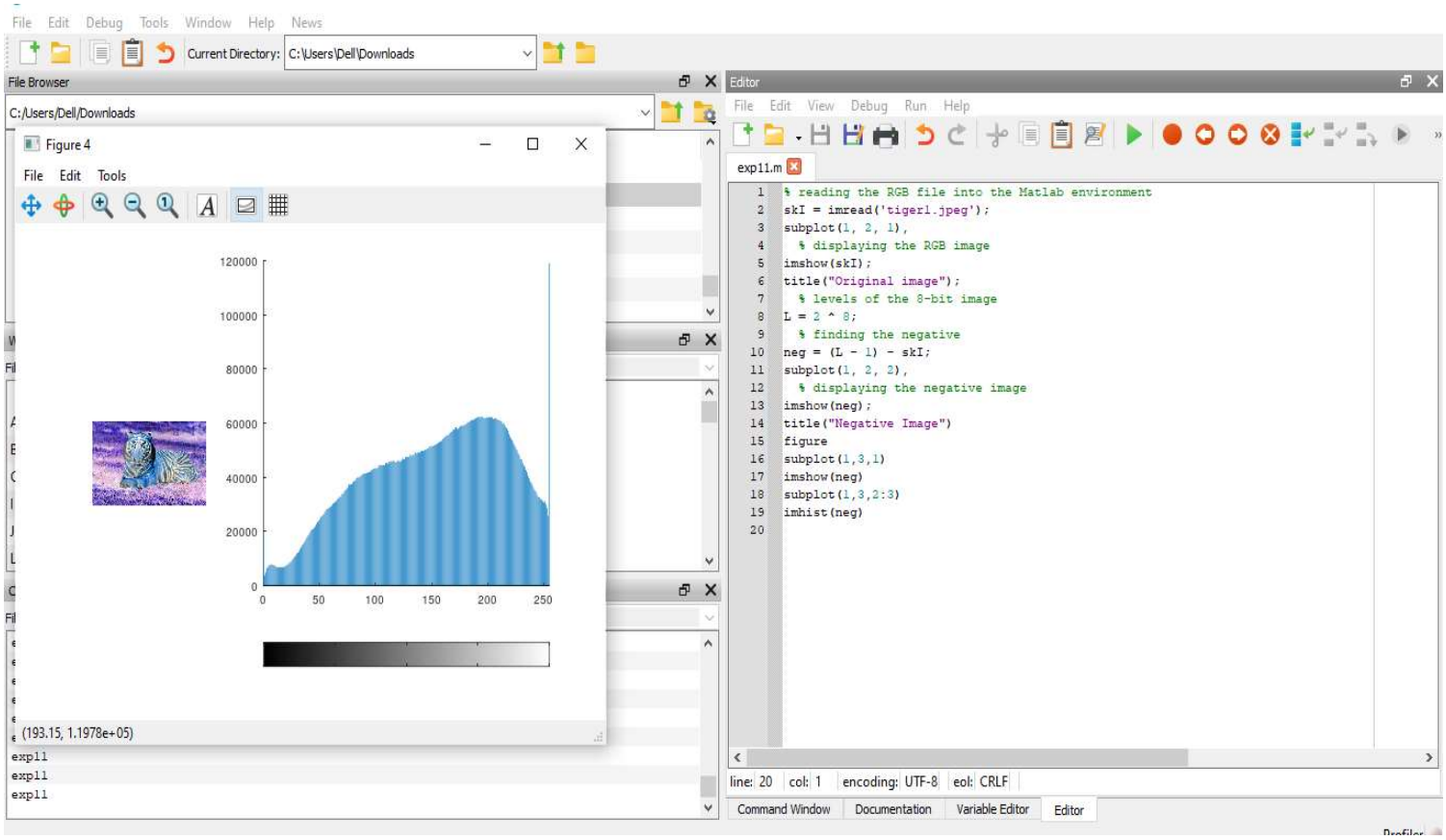
Code and output-

```
I = imread(‘tiger1.jpeg');

%Display the image and its histogram. The original image has low contrast, with most pixel values in the middle of the intensity range.

figure

subplot(1,3,1)

imshow(I)

subplot(1,3,2:3)

imhist(I)
```




```
% reading the RGB file into the Matlab environment
skI = imread('tiger1.jpeg');
subplot(1, 2, 1),
  % displaying the RGB image
imshow(skI);
title("Original image");
  % levels of the 8-bit image
L = 2 ^ 8;
  % finding the negative
```

```
neg = (L - 1) - skI;
subplot(1, 2, 2),
  % displaying the negative image
imshow(neg);
title("Negative Image")
figure
subplot(1,3,1)
imshow(neg)
subplot(1,3,2:3)
imhist(neg)
```



EXPERIMENT – 07

Perform various edge detection method for a given image .

Answer-

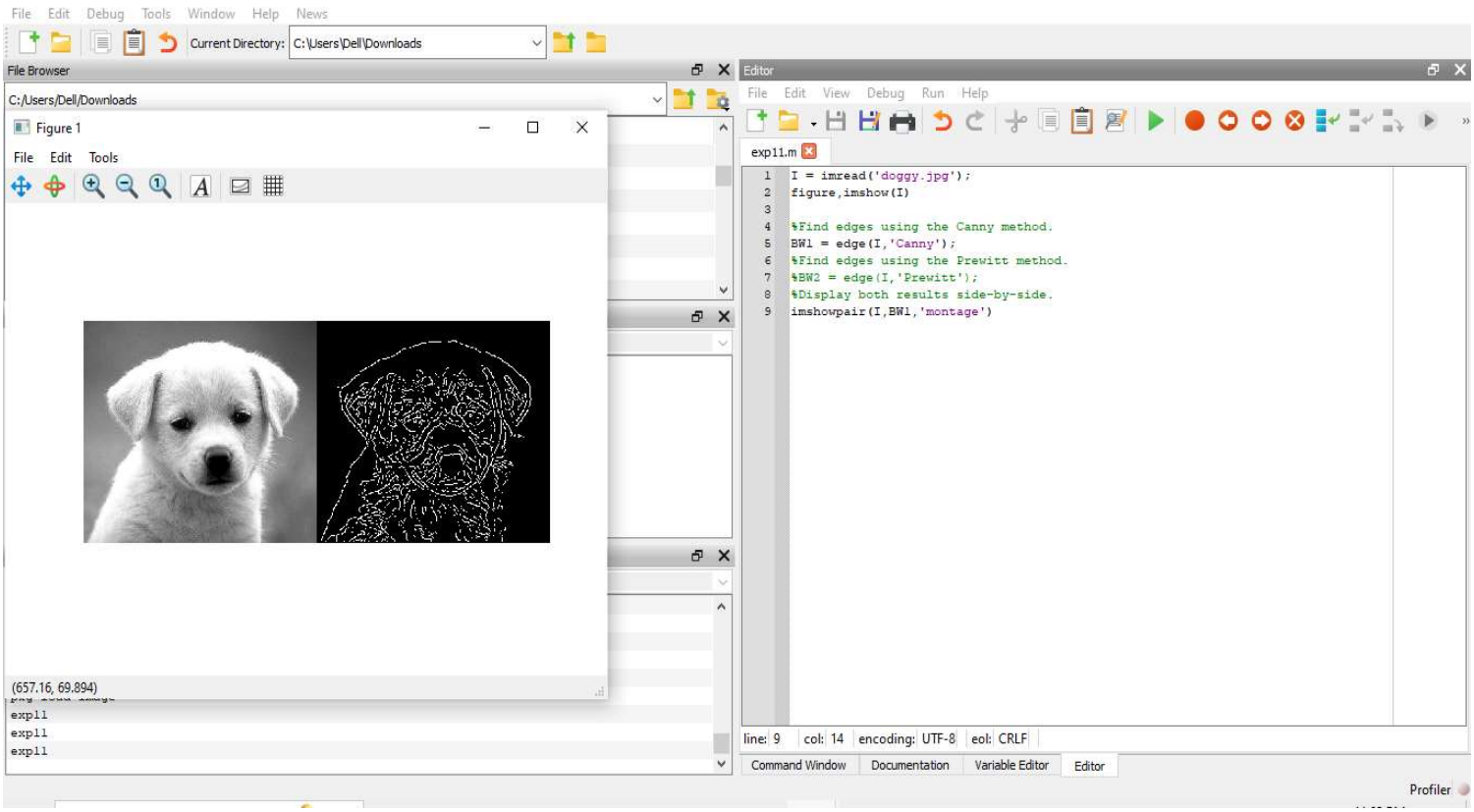
Theory-

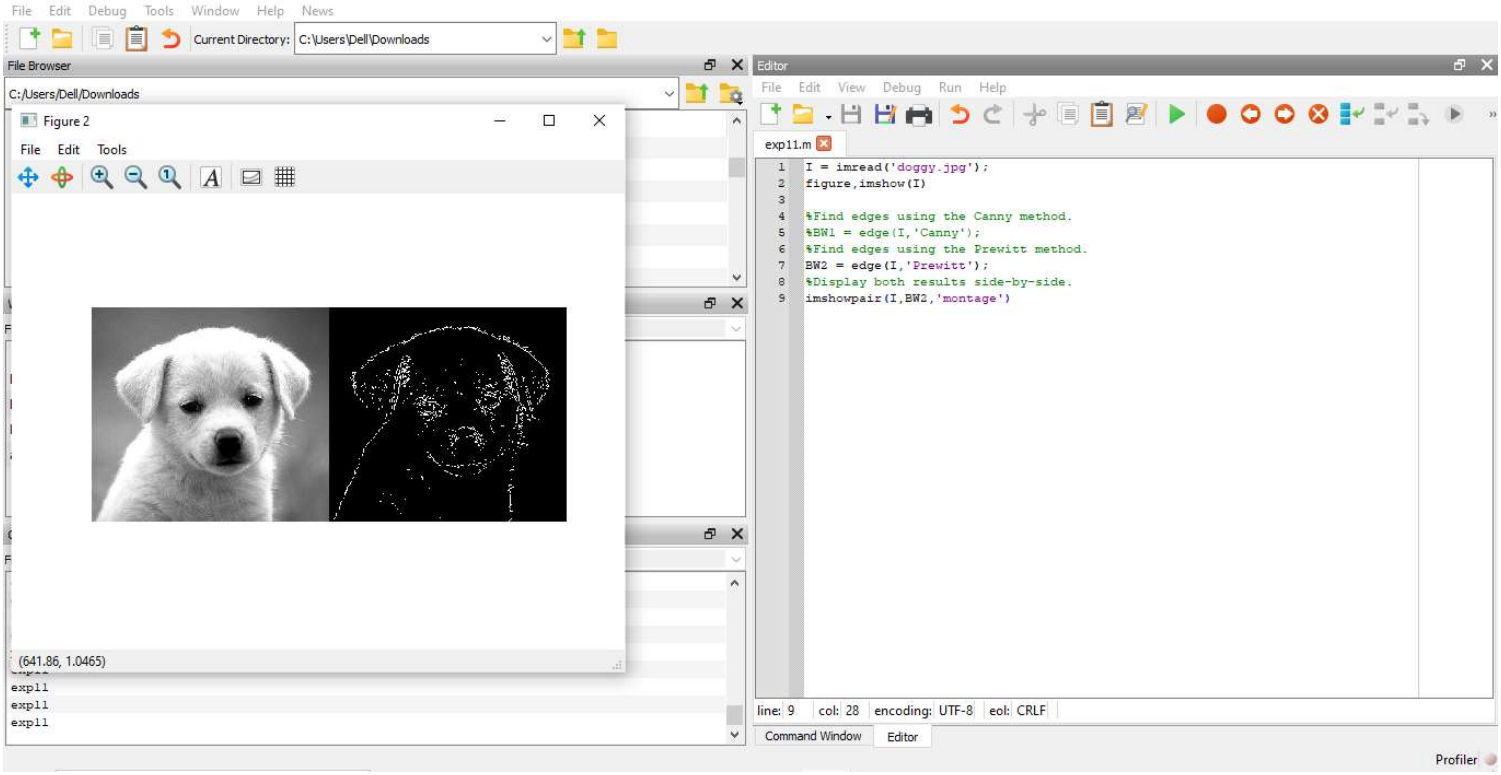
Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image.

Code and output-

```
I = imread('doggy.jpg');
imshow(I)
```

```
%Find edges using the Canny method.
BW1 = edge(I,'Canny');
%Find edges using the Prewitt method.
BW2 = edge(I,'Prewitt');
%Display both results side-by-side.
imshowpair(BW1,BW2,'montage')
```


File Edit Debug Tools Window Help News
Current Directory: C:\Users\Dell\Downloads
File Browser
C:/Users/Dell/Downloads
Figure 1
File Edit Tools
(657.16, 69.894)
exp11
exp11
exp11
Editor
File Edit View Debug Run Help
exp11.m
1 I = imread('doggy.jpg');
2 figure,imshow(I)
3
4 %Find edges using the Canny method.
5 BW1 = edge(I,'Canny');
6 %Find edges using the Prewitt method.
7 %BW2 = edge(I,'Prewitt');
8 %Display both results side-by-side.
9 imshowpair(I,BW1,'montage')
line: 9 col: 14 encoding: UTF-8 eol: CRLF
Command Window Documentation Variable Editor Editor
Profiler


File Edit Debug Tools Window Help News
Current Directory: C:\Users\Dell\Downloads
File Browser
C:/Users/Dell/Downloads
Figure 2
File Edit Tools
(641.86, 1.0465)
exp11
exp11
exp11
Editor
File Edit View Debug Run Help
exp11.m
1 I = imread('doggy.jpg');
2 figure,imshow(I)
3
4 %Find edges using the Canny method.
5 %BW1 = edge(I,'Canny');
6 %Find edges using the Prewitt method.
7 BW2 = edge(I,'Prewitt');
8 %Display both results side-by-side.
9 imshowpair(I,BW2,'montage')
line: 9 col: 28 encoding: UTF-8 eol: CRLF
Command Window Editor
Profiler

EXPERIMENT – 08

Perform logical and arithmetic operations with two different images.

Answer-

Theory-

logical operators are often used to combine two (mostly binary) images. In the case of integer images, the logical operator is normally applied in a bitwise way. In this lecture we will talk about arithmetic operations such as subtraction and averaging as well as logic operations such as Not, And, and OR.

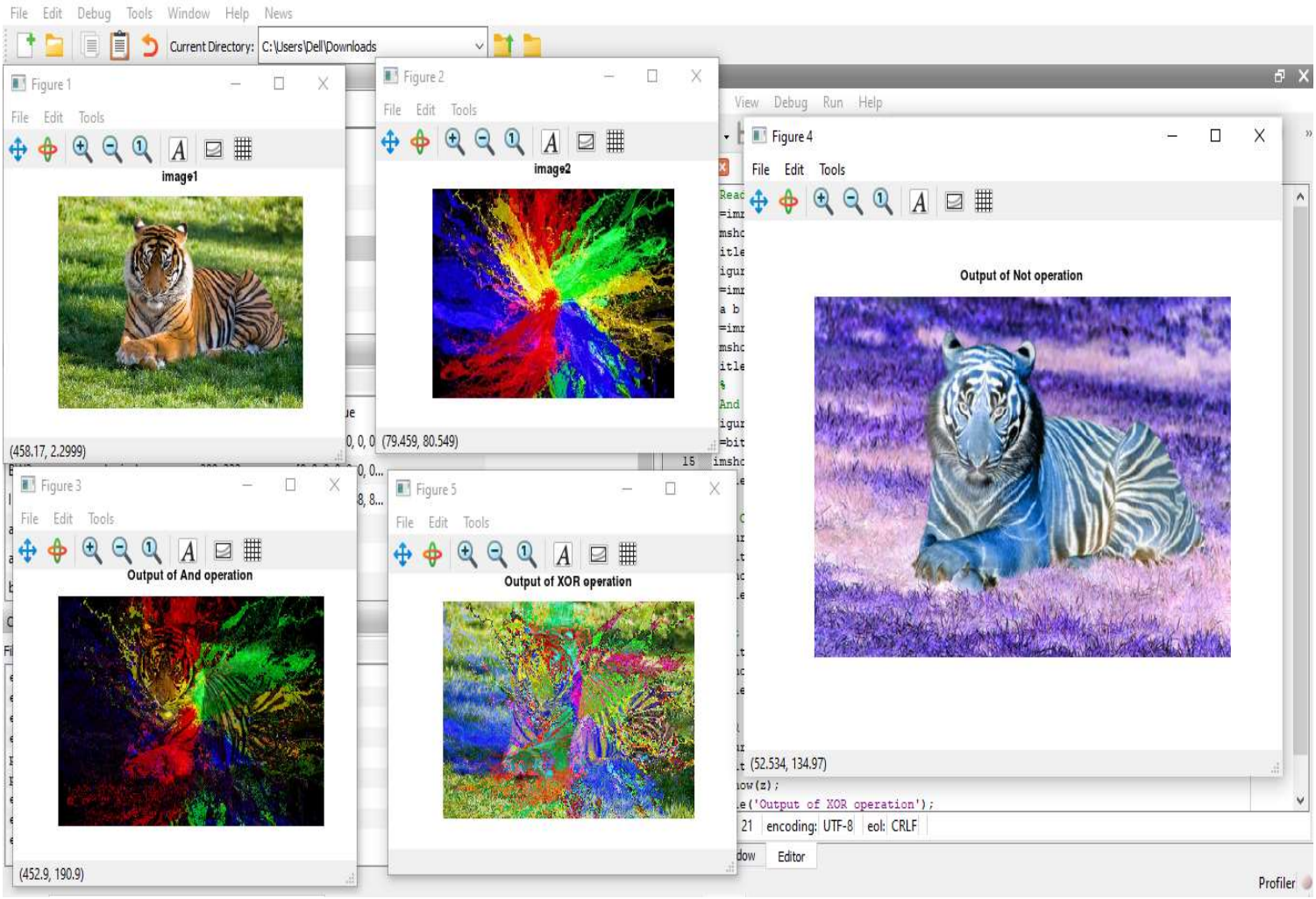
Logical operation-

Code and output-

```
%%

%Reading Required Images & required Preprocessing

x=imread(‘tiger1.jpeg');

imshow(x);

title('image1');

figure;

y=imread('color.jpeg');

[a b c]=size(x);
```

```
y=imresize(y,[a,b]);
imshow(y);
title('image2');
%%
%And Operation
figure;
z=bitand(x,y);
imshow(z);
title('Output of And operation');
%%
%Or Operation
figure;
z=bitor(x,y);
imshow(z);
title('Output of Or operation');
%%
%Not Operation
z=bitcmp(x);
imshow(z);
```

```
title('Output of Not operation');
%%
%XOR Operation
figure;
z=bitxor(x,y);
imshow(z);
title('Output of XOR operation');
```



Arithmetic operation-

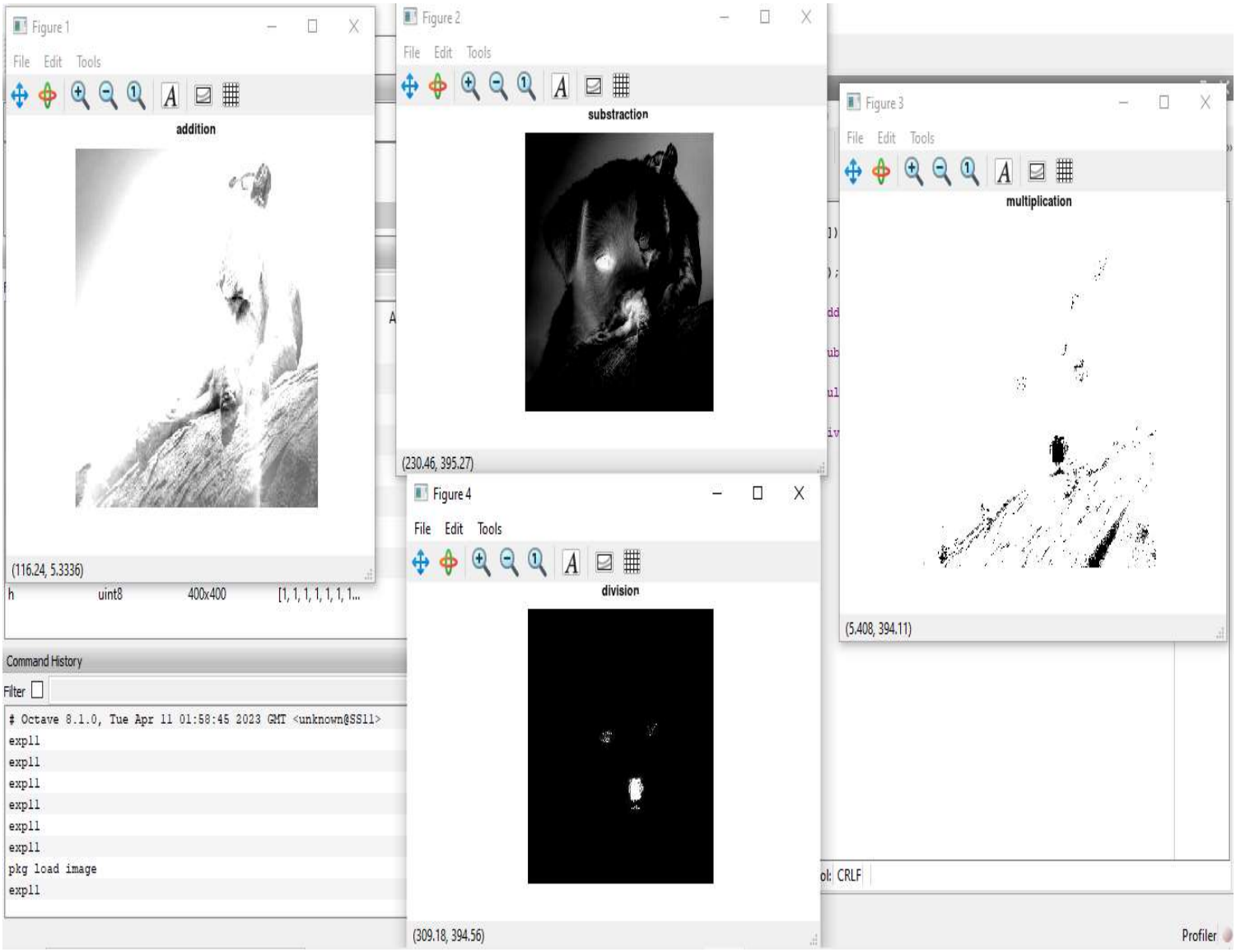
Theory-

Image arithmetic is **the application of one or more images to one of the standard arithmetic operations or a logical operator**. The operators are applied pixel by pixel, so the value of a pixel in the output image is determined solely by the values of the corresponding pixels in the input images.

Code and output-

```
a=imread ('lion.jpeg');

b = imresize (a, [400,400]);

c=imread ('doggy.jpg');

d= imresize (c, [400,400]);

e=imadd(b,d);

figure,imshow(e),title('addition');

f=imsubtract(b,d);

figure,imshow(f),title('substraction');

g=immultiply(b,d);
```

```
figure,imshow(g),title('multiplication');

h=imdivide(b,d);

figure,imshow(h),title('division');
```



EXPERIMENT – 09

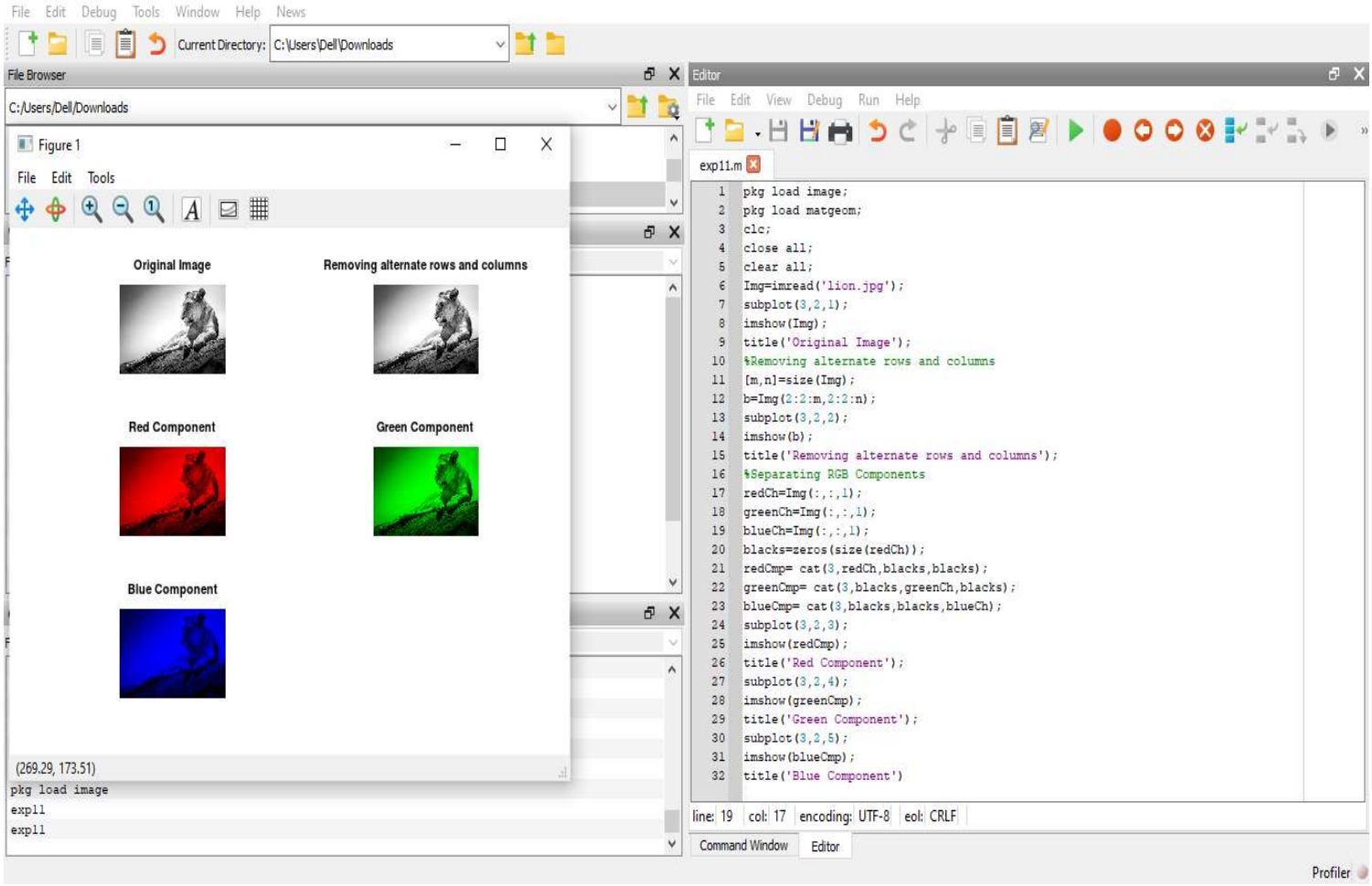
AIM:- Remove alternate rows and columns from an Image and plot RGB components of an Image separately in Octave

Code and output-

```
pkg load image;

pkg load matgeom;

clc;

close all;

clear all;

Img=imread('lion.jpg');

subplot(3,2,1);

imshow(Img);

title('Original Image');

%Removing alternate rows and columns

[m,n]=size(Img);
```

```
b=Img(2:2:m,2:2:n);
 subplot(3,2,2);
imshow(b);
title('Removing alternate rows and columns');
 %Separating RGB Components
 redCh=Img(:,:,1);
 greenCh=Img(:,:,2);
blueCh=Img(:,:,3);
blacks=zeros(size(redCh));
redCmp= cat(3,redCh,blacks,blacks);
 greenCmp= cat(3,blacks,greenCh,blacks);
blueCmp= cat(3,blacks,blacks,blueCh);
subplot(3,2,3);
imshow(redCmp);
 title('Red Component');
subplot(3,2,4);
 imshow(greenCmp);
```

```
 title('Green Component');
subplot(3,2,5);
imshow(blueCmp);
 title('Blue Component');
```



EXPERIMENT – 10

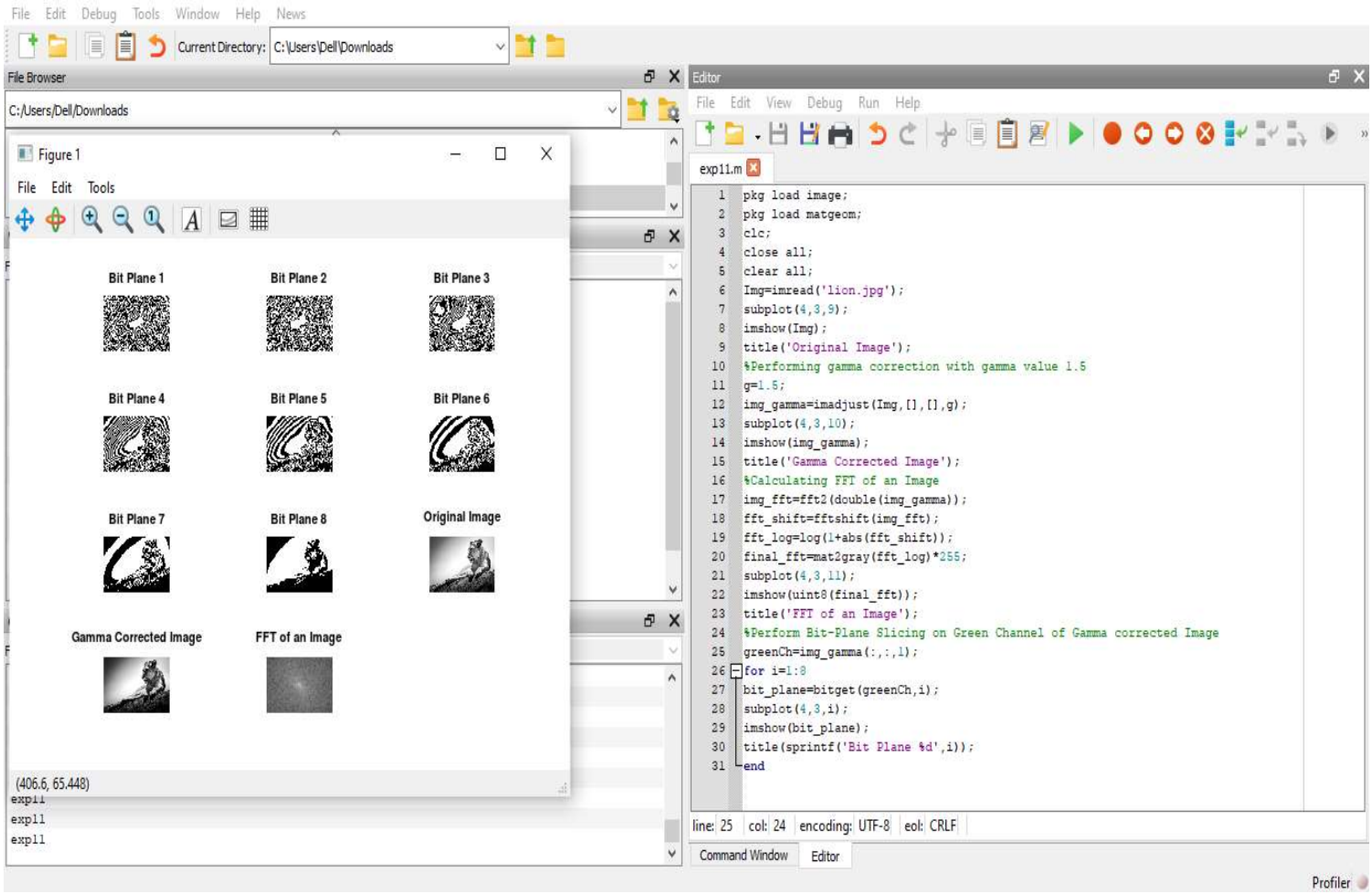
AIM: - Apply Gamma Correction on an Image, show the FFT of an Image also use the BIT Plane Slicing for the Green Channel of an Image.

Code and output-

```
pkg load image;
pkg load matgeom;
 clc;
close all;
 clear all;
 Img=imread('lion.jpg');
subplot(4,3,9);
 imshow(Img);
 title('Original Image');
 %Performing gamma correction with gamma value 1.5
 g=1.5;
```

```
 img_gamma=imadjust(Img,[],[],g);
 subplot(4,3,10);
 imshow(img_gamma);
 title('Gamma Corrected Image');
 %Calculating FFT of an Image
img_fft=fft2(double(img_gamma));
 fft_shift=fftshift(img_fft);
 fft_log=log(1+abs(fft_shift));
 final_fft=mat2gray(fft_log)*255;
subplot(4,3,11);
imshow(uint8(final_fft));
 title('FFT of an Image');
%Perform Bit-Plane Slicing on Green Channel of Gamma
corrected Image
 greenCh=img_gamma(:,:,2);
 for i=1:8
   bit_plane=bitget(greenCh,i);
```

```
  subplot(4,3,i);
  imshow(bit_plane);
  title(sprintf('Bit Plane %d',i));
end
```



THANK YOU !