| S.No | PRACTICAL QUESTIONS | PAGE NO. |
| --- | --- | --- |
| **1** | Write program to read and display digital image using MATLAB or SCILAB  a. Become familiar with SCILAB/MATLAB Basic commands  b. Read and display image in SCILAB/MATLAB  c. Resize given image  d. Convert given color image into gray-scale image  e. Convert given color/gray-scale image into black & white image  f. Draw image profile  g. Separate color image in three R G & B planes  h. Create color image using R, G and B three separate planes  i. Flow control and LOOP in SCILAB  j. Write given 2-D data in image file | 4 |
| **2** | To write and execute image processing programs using point processing method  a. Obtain Negative image  b. Obtain Flip image  c. Thresholding  d. Contrast stretching | **10** |
| **3** | To write and execute programs for image arithmetic operations  a. Addition of two images  b. Subtract one image from other image  c. Calculate mean value of image | **13** |
| **4** | To write and execute programs for image logical operations  a. AND operation between two images  b. OR operation between two images  c. Calculate intersection of two images  d. NOT operation (Negative image) | 16 |
| **5** | To write a program for histogram calculation and equalization using  a. Standard MATLAB function  b. Program without using standard MATLAB functions | 22 |
| **6** | To write and execute program for geometric transformation of image  a. Translation  b. Scaling  c. Rotation  d. Shrinking  e. Zooming | 27 |
| **7** | To understand various image noise models and to write programs for  a. image restoration  b. Remove Salt and Pepper Noise  c. Minimize Gaussian noise  d. Median filter | 31 |
| **8** | Write and execute programs to use spatial low pass and high pass filters | 34 |
| **9** | Write and execute programs for image frequency domain filtering  a. Apply FFT on given image  b. Perform low pass and high pass filtering in frequency domain  c. Apply IFFT to reconstruct image | 37 |
| **10** | Write a program in C and MATLAB/SCILAB for edge detection using different edge  detection mask | 40 |
| **11** | Write and execute program for image morphological operations erosion and dilation. | 46 |

**1. Write program to read and display digital image using MATLAB or SCILAB**

a. Become familiar with SCILAB/MATLAB Basic commands

b. Read and display image in SCILAB/MATLAB

c. Resize given image

d. Convert given color image into gray-scale image

e. Convert given color/gray-scale image into black & white image

f. Draw image profile

g. Separate color image in three R G & B planes

h. Create color image using R, G and B three separate planes

i. Flow control and LOOP in SCILAB

j. Write given 2-D data in image file

**ANS.**

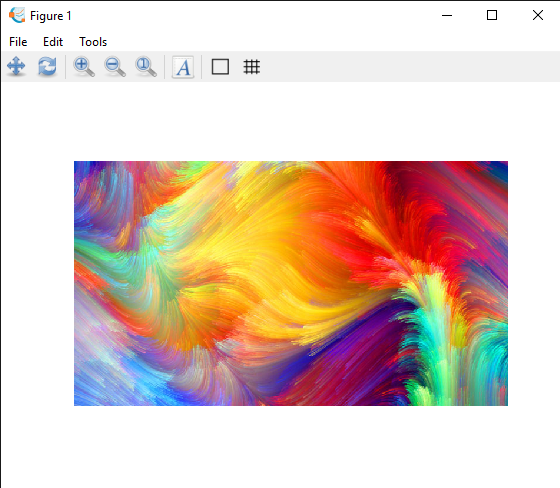
**b.**

clear all;

close all;

clc;

img = imshow('colour.jpg')



**c.**

clear all;

close all;

clc;

pkg load image

img = imread('colour.jpg');

subplot(1,2,1)

imshow(img);

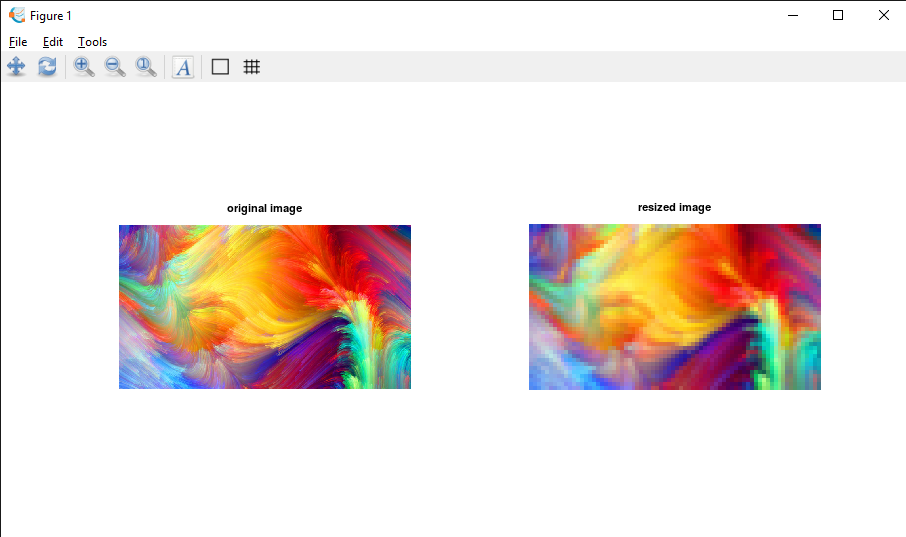
title('original image')

resized\_img = imresize(img,0.1);

subplot(1,2,2)

imshow(resized\_img);

title('resized image')



**d.**

clear all;

close all;

clc;

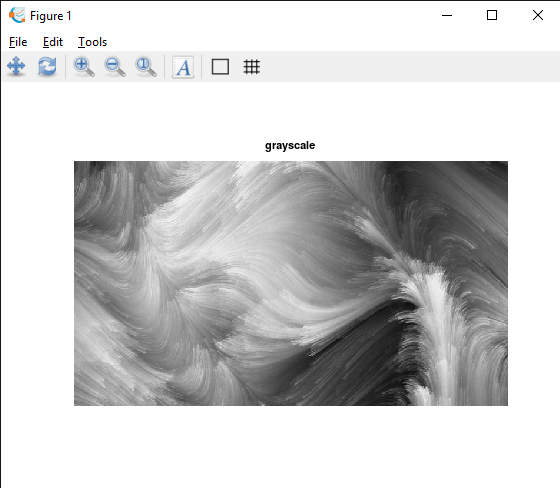
pkg load image

img = imread('colour.jpg');

grayscale\_img = rgb2gray(img);

imshow(grayscale\_img)

title('grayscale')



**e.**

clear all;

close all;

clc;

pkg load image

img = imread('colour.jpg');

bw\_img = im2bw(img);

imshow(bw\_img)

title('black and white')



**g.**

clear all;

close all;

clc;

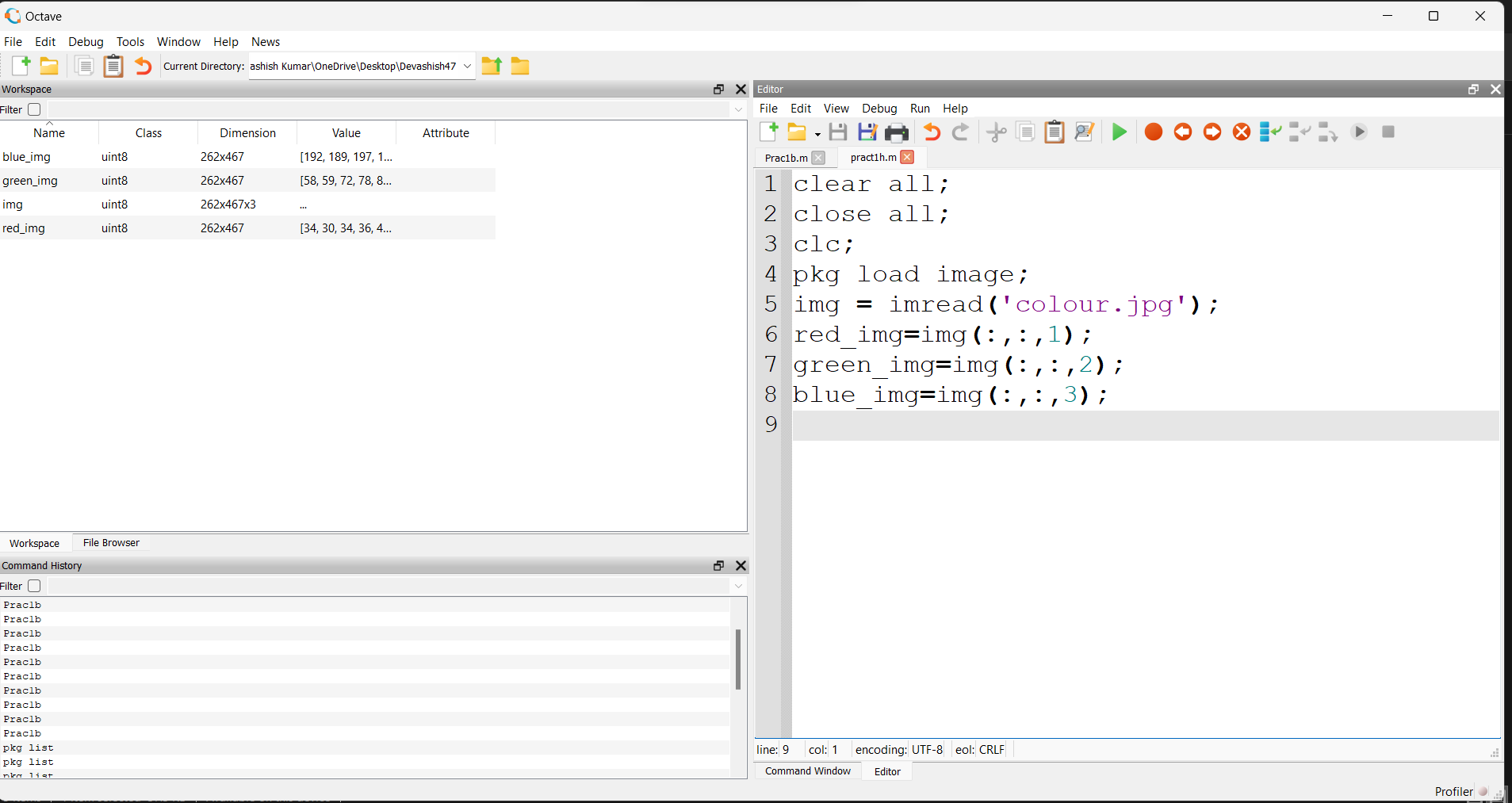
pkg load image;

img = imread('colour.jpg');

red\_img=img(:,:,1);

green\_img=img(:,:,2);

blue\_img=img(:,:,3);



**h.**

clear all;

close all;

clc;

pkg load image;

img = imread('colour.jpg');

temp\_img=img;

subplot(2,2,1);

imshow(img);

title('original img');

#red image

img(:,:,2)=0;

img(:,:,3)=0;

subplot(2,2,2);

imshow(img);

title('red img');

#green image

img=temp\_img;

img(:,:,1)=0;

img(:,:,3)=0;

subplot(2,2,3);

imshow(img);

title('green img');

#blue image

img=temp\_img;

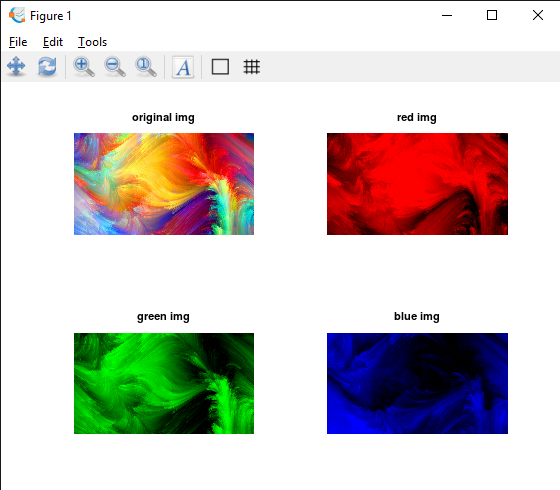
img(:,:,1)=0;

img(:,:,2)=0;

subplot(2,2,4);

imshow(img);

title('blue img');



**j.**

clear all;

close all;

clc;

pkg load image;

m=[0:9];

n=reshape(m,2,5);

imshow(n,[]);



# Q2. To write and execute image processing programs using point processing method

1. **Obtain Negative image**

# Obtain Flip image

1. **Thresholding**

# Contrast stretching CODE:

clc;

pkg load image;

A=imread('colour.jpg'); subplot(3,2,1);

imshow(A); title('ORIGINAL IMAGE');

L = 2 ^ 8;

neg = (L - 1) - A;

subplot(3, 2, 2), imshow(neg); title("Negative Image")

mirror\_img = flip(A, 2); subplot(3,2,3); imshow(mirror\_img);

title('Mirror image');

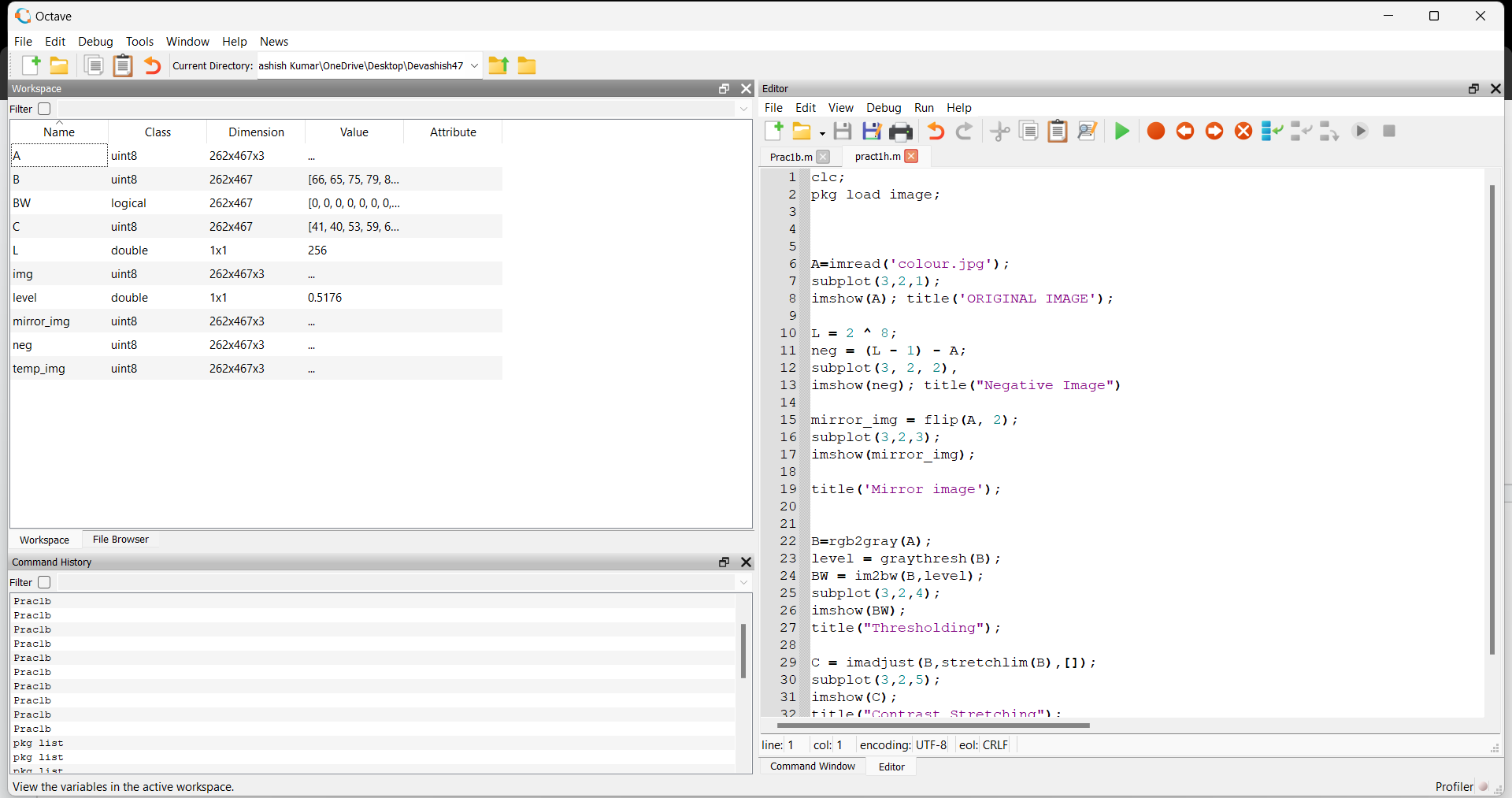
B=rgb2gray(A);

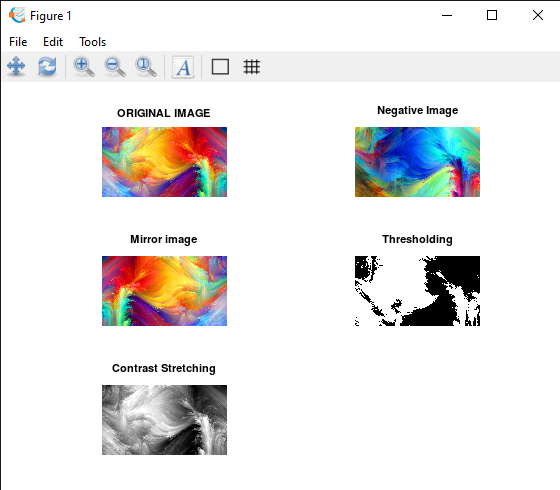
level = graythresh(B); BW = im2bw(B,level); subplot(3,2,4); imshow(BW); title("Thresholding");

C = imadjust(B,stretchlim(B),[]); subplot(3,2,5);

imshow(C);

title("Contrast Stretching");





**Q3. To write and execute programs for image arithmetic operations**

1. **Addition of two images**
2. **Subtract one image from other image**
3. **Calculate mean value of image**

CODE:

clc;

pkg load image;

i1 = imread("beach.jpg");

subplot(2,2,1);

imshow(i1); title("IMAGE 1");

g=size(i1);

i2 = imread("birds.jpg");

i2 = imresize(i2,[g(1),g(2)]);

subplot(2,2,2);

imshow(i2); title("IMAGE 2");

i3 = i1 + i2;

subplot(2,2,3);

imshow(i3);

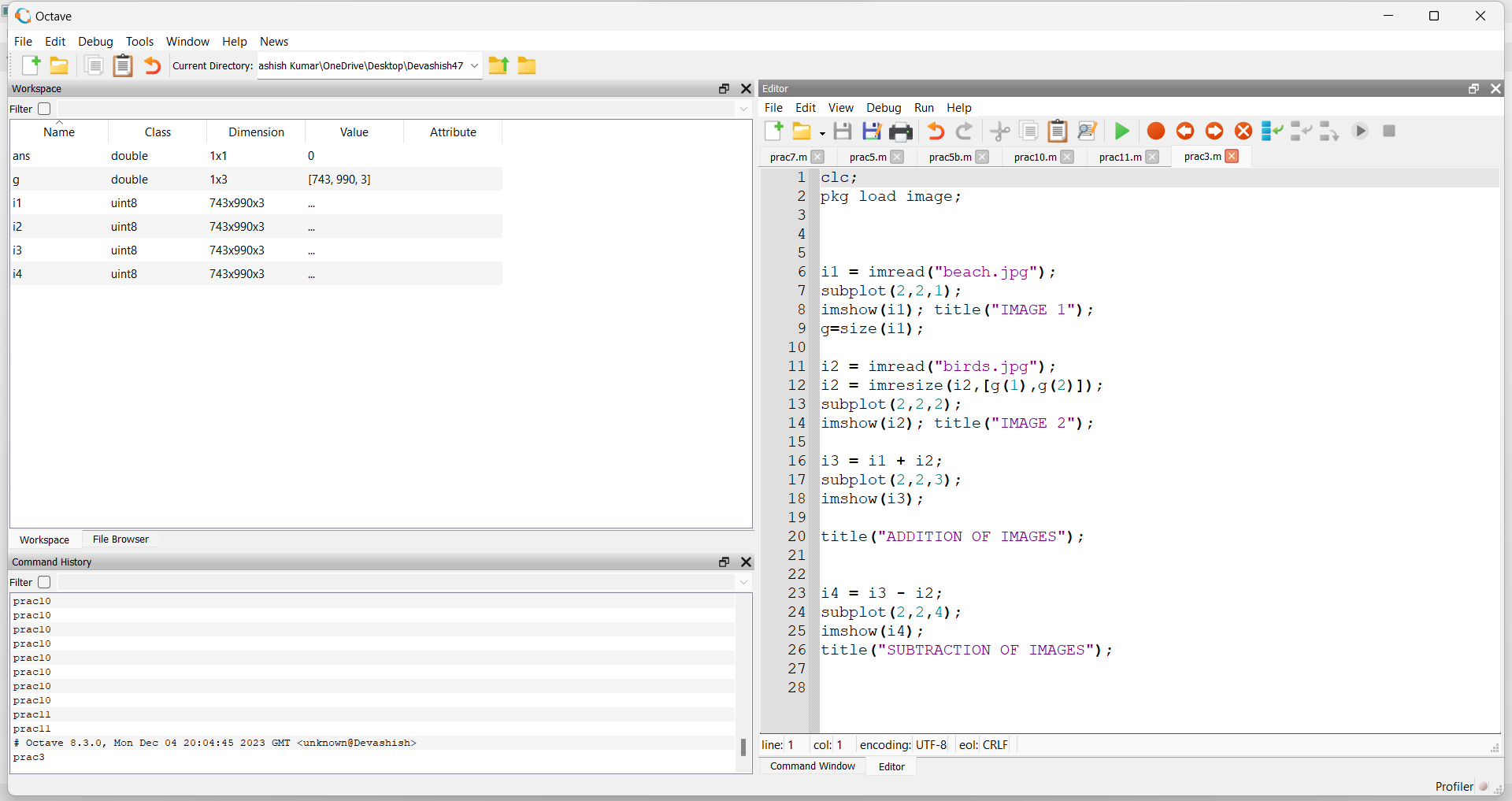
title("ADDITION OF IMAGES");

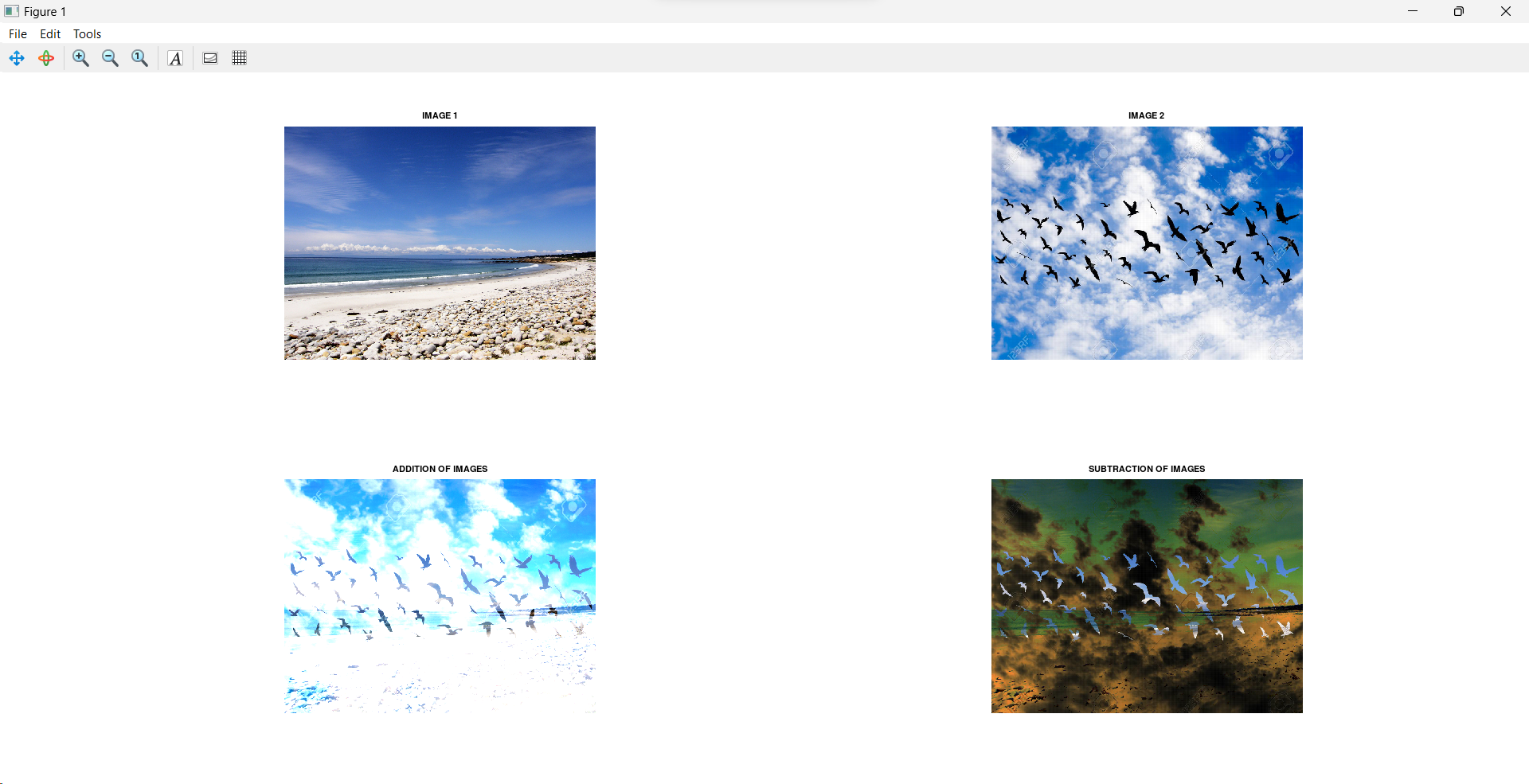
i4 = i3 - i2;

subplot(2,2,4);

imshow(i4);

title("SUBTRACTION OF IMAGES");





**Q4. To write and execute programs for image logical operations**

1. **AND operation between two images**
2. **OR operation between two images**
3. **Calculate intersection of two images**
4. **NOT operation (Negative image)**

clc;

pkg load image;

%adding image 1 A=imread(‘insta.jfif'); subplot(3,2,1);

imshow(A); title('A');

%adding image 2 B=imread('blue2.png'); subplot(3,2,2);

imshow(B); title('B');

%A AND B

C=bitand(A,B);

subplot(3,2,3); imshow(C); title('A AND B');

%A OR B

D=bitor(A,B);

subplot(3,2,4); imshow(D); title('A OR B');

%A Intersection B r1=im2bw(A); r2=im2bw(B); E=bitand(r1,r2); subplot(3,2,5); imshow(r1); title('A(BW Image)');

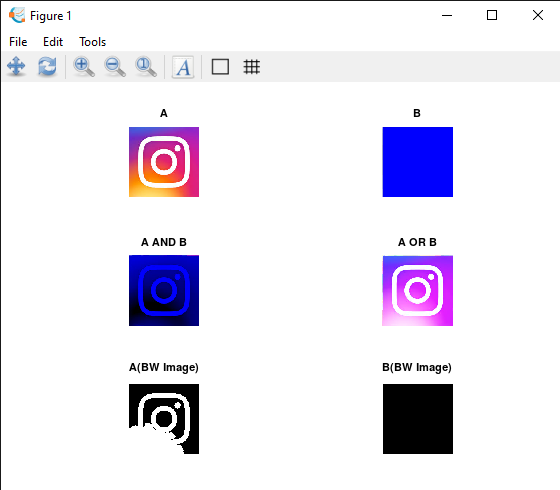
subplot(3,2,6); imshow(r2); title('B(BW Image)');

subplot(3,2,7); imshow(E); title('Intersection');

%NOT A

F=bitcmp(A); subplot(3,2,8);

imshow(F); title('NOT A');



# Q5. To write a program for histogram calculations and equalisations using:

1. **Standard MATLAB function**

# Program without using standard MATLAB function

1. code -

clc

clear all

close all

a=imread('loki.png');

b=rgb2gray(a);

subplot(2,2,1);

imshow(b);

title('Original Grayscale Image');

subplot(2,2,3);

imhist(b);

title('Histogram of Original Grayscale Image');

j=histeq(b);

subplot(2,2,2);

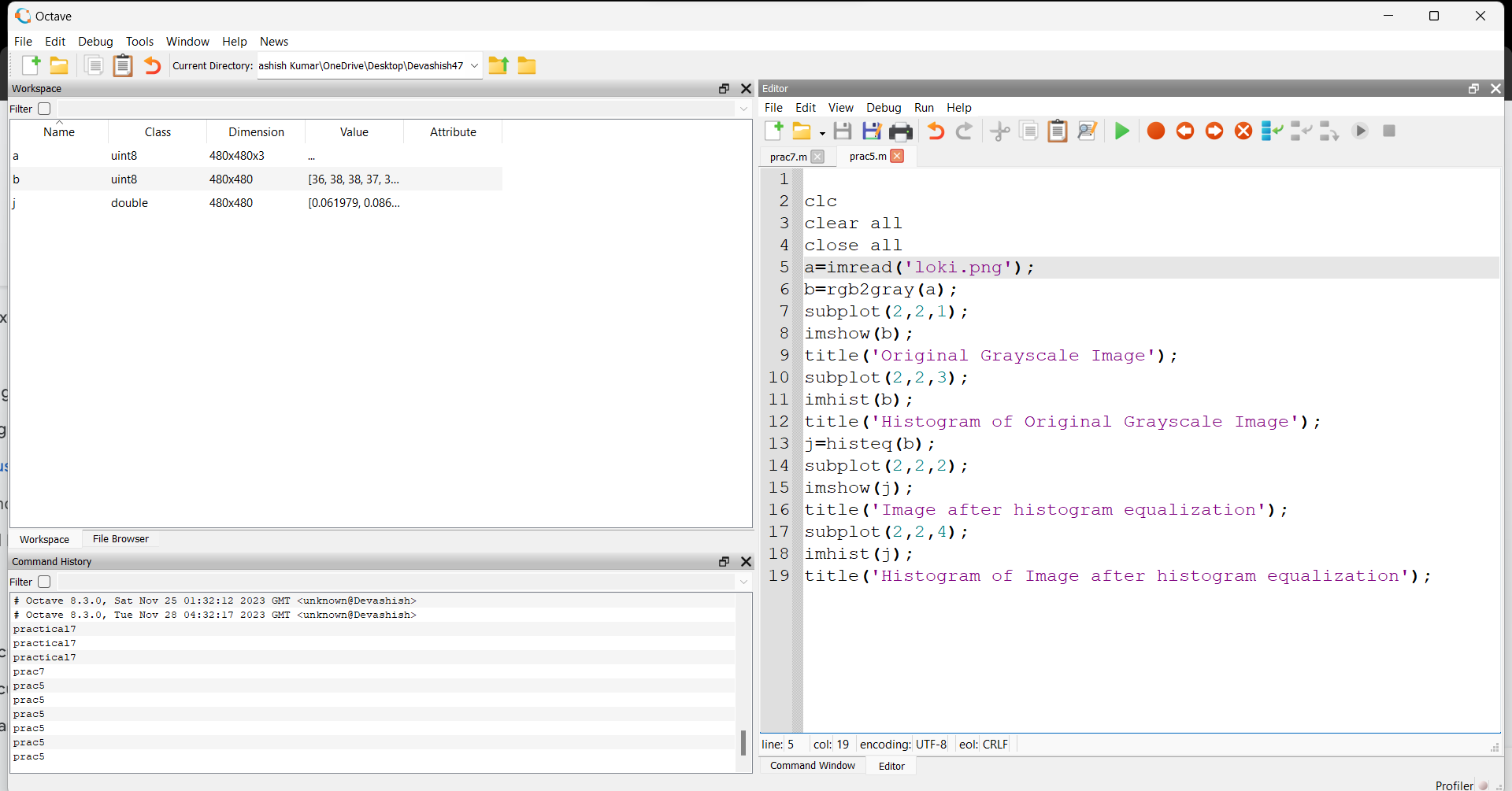
imshow(j);

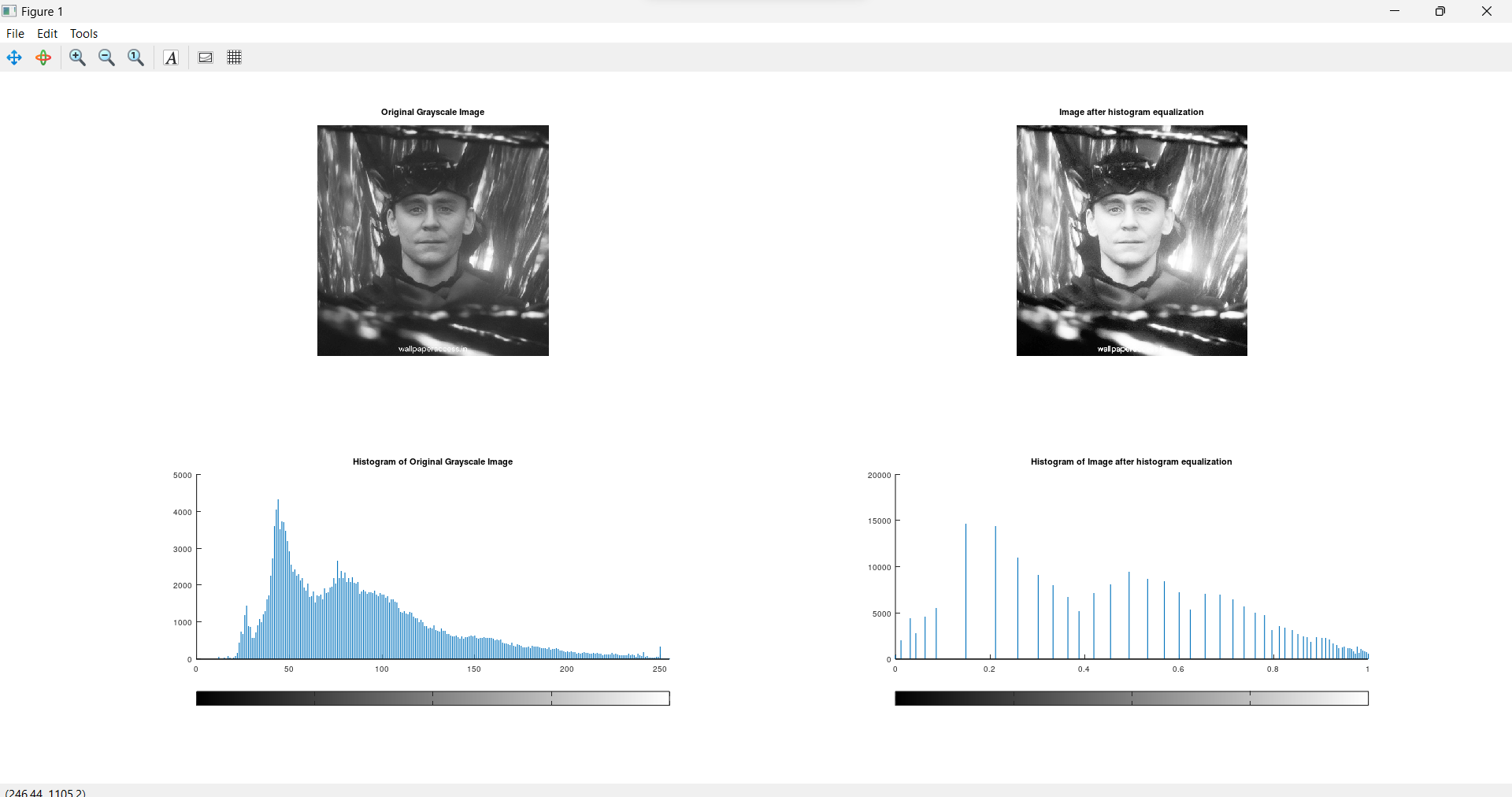
title('Image after histogram equalization');

subplot(2,2,4);

imhist(j);

title('Histogram of Image after histogram equalization');





1. **Code -**

**Q6. To write and execute program for geometric transformation of image**

1. **Translation**
2. **Scaling**
3. **Rotation**
4. **Shrinking**
5. **Zooming**

clc;

pkg load image;

A = imread("randomimage1.jpg"); figure;

subplot(231); imshow(A);

title("ORIGINAL IMAGE");

axis on

%image translation

x\_shift = 200; %shift column 200px y\_shift = 100; %shift row 100px

%assigning empty matrix for the resultant image

%100px in rows and 200px in columns

B = uint8(zeros(size(A,1)+y\_shift-1, size(A,2)+x\_shift-1, size(A,3)));

%note : imshow displays the image with the upper right corner at (0,0)

%translating the image

B(y\_shift:end, x\_shift:end, :) = A; subplot(232);

imshow(B);

title("IMAGE TRANSLATION");

axis on

%image scaling

%set magnification factor mfactor = 0.75;

C = imresize(A, mfactor, "Bilinear"); %scaling subplot(233);

imshow(C);

title("IMAGE SCALING");

axis on

%image rotation

%set rotation angle rAngle = 45;

D = imrotate(A, rAngle); %rotating by 45 degrees subplot(234);

imshow(D);

title("IMAGE ROTATION");

axis on

%image shrinking

%set magnification factor < 1 mfactor = 0.5;

E = imresize(A, mfactor, "Bilinear"); %shrinking subplot(235);

imshow(E);

title("IMAGE SHRINKING");

axis on

%image zooming

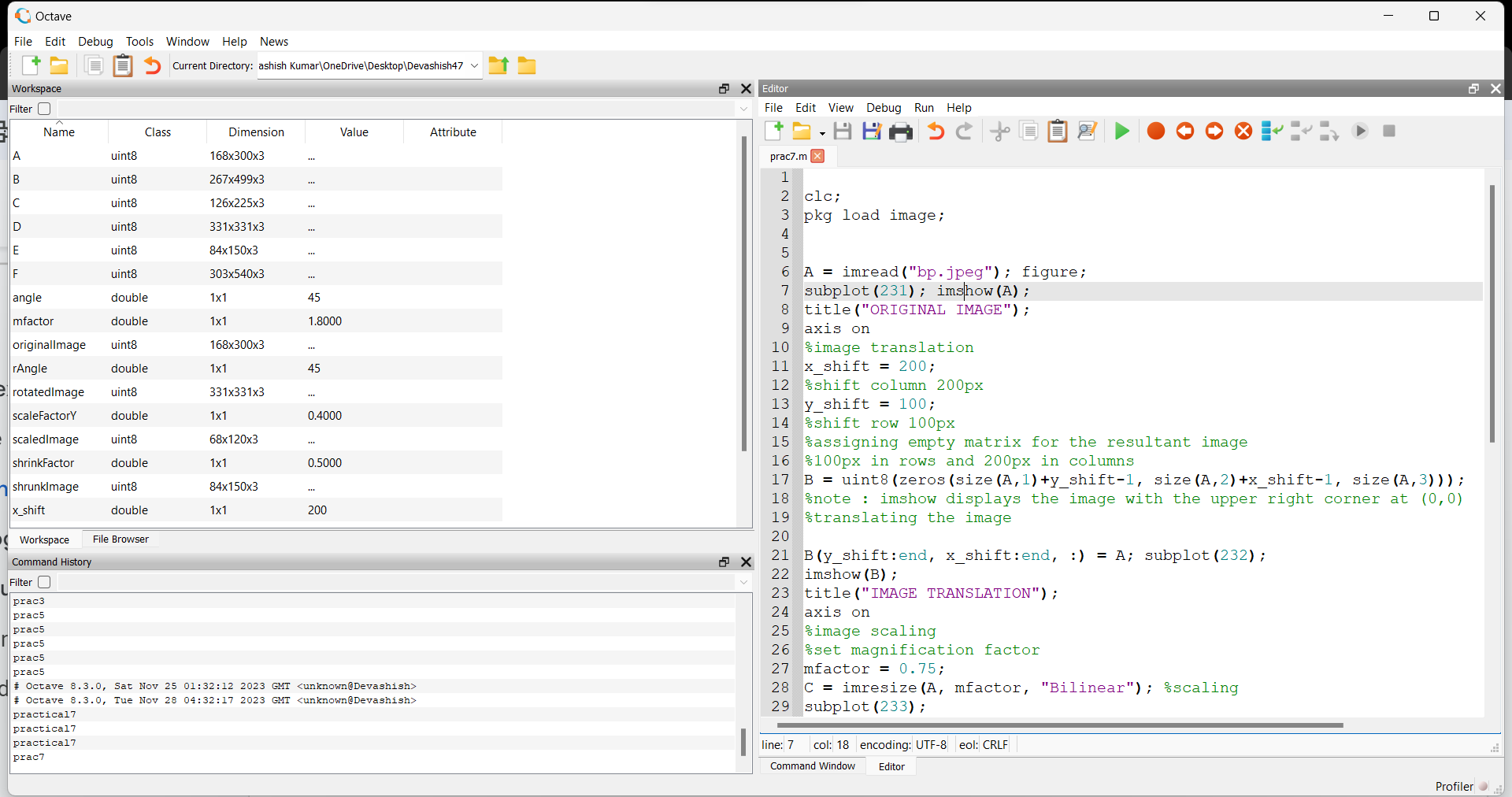
%set magnification factor > 1 mfactor = 1.8;

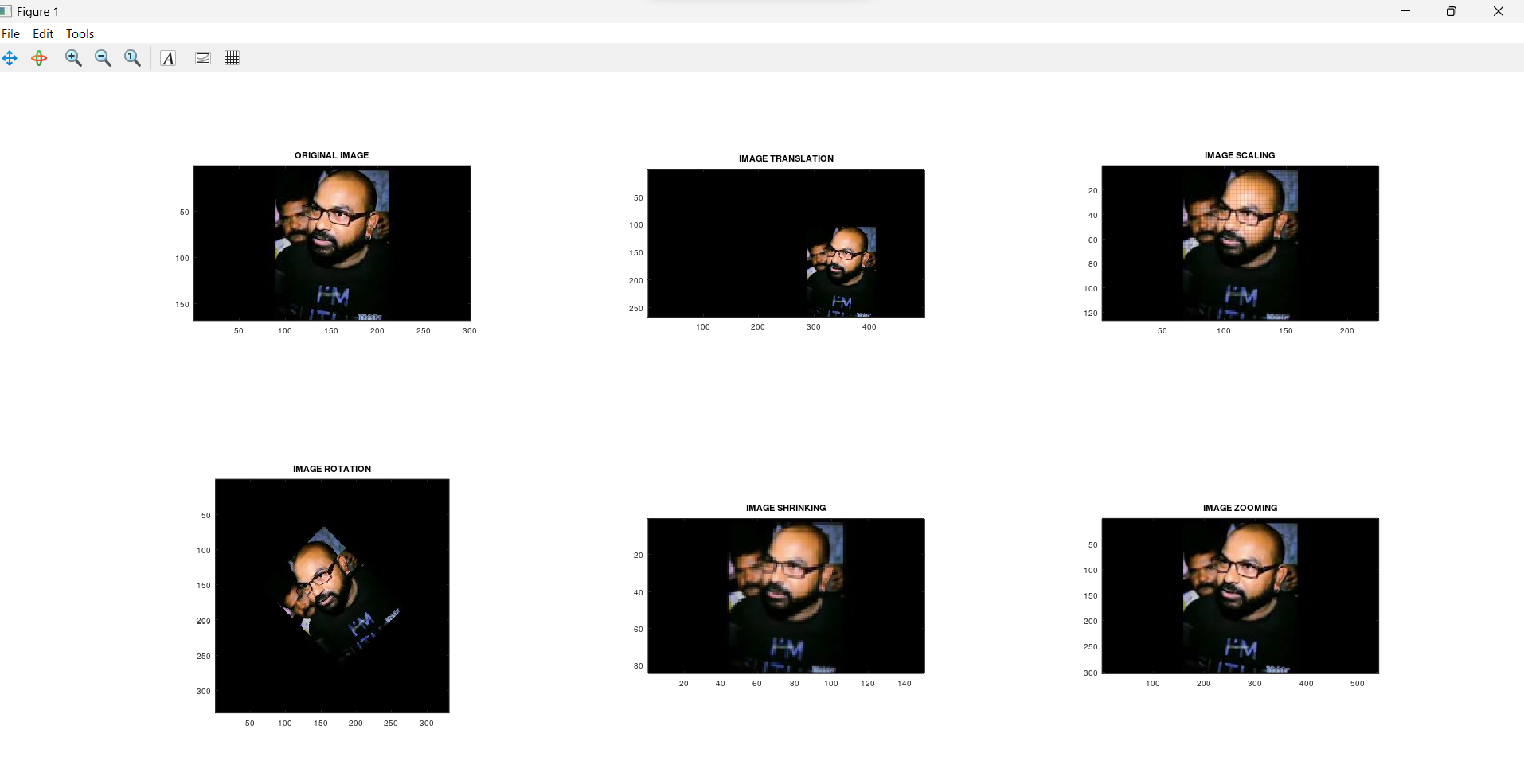
F = imresize(A, mfactor, "Bilinear"); %zooming subplot(236);

imshow(F);

title("IMAGE ZOOMING");

axis on





## Q7. To understand various image noise models and to write programs for

1. image restoration

## Remove Salt and Pepper Noise

1. Minimize Gaussian noise

## Median

## filter

clc;

pkg load image;

%IMAGE RESTORATION

%MINIMIZING GAUSSIAN NOISE

figure;

A=rgb2gray(imread("baigan.jpg"));

subplot(121);

imshow(A);

title("Original Image - With Gaussian Noise")

%ARITHMETIC MEAN FILTER

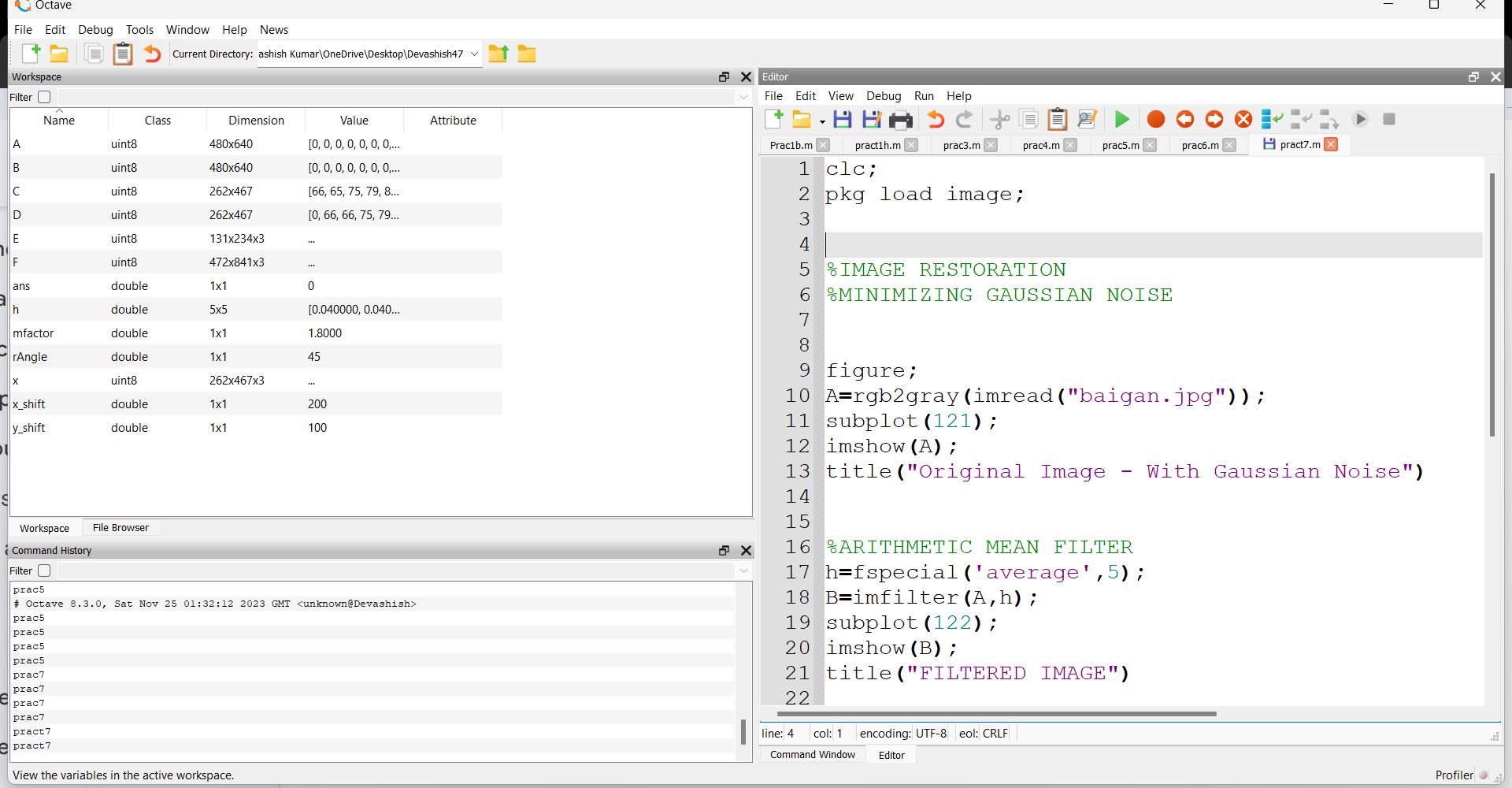
h=fspecial('average',5);

B=imfilter(A,h);

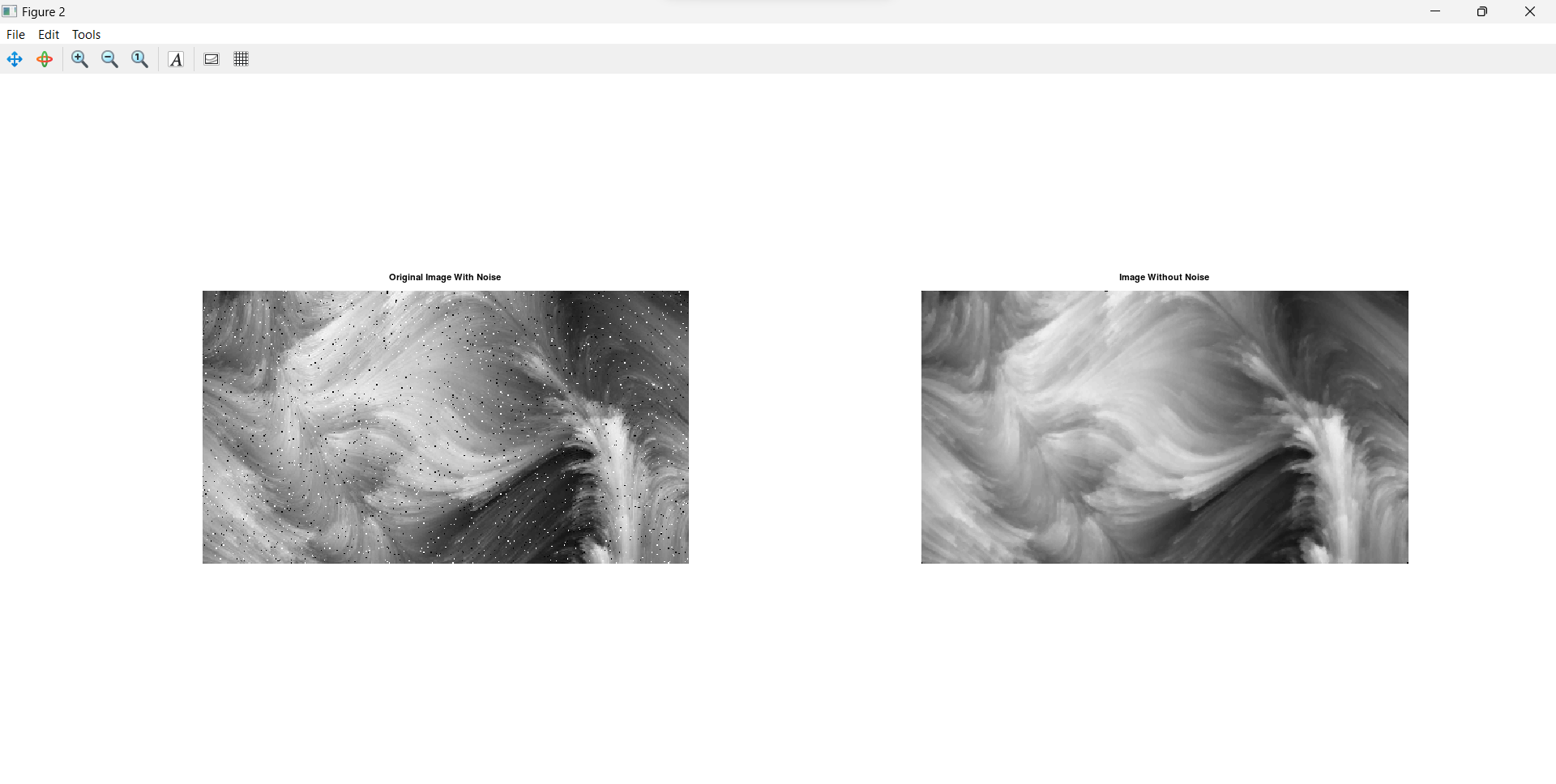
subplot(122);

imshow(B);

title("FILTERED IMAGE")







# Q8. Write and execute programs to use spatial low and high pass filters.

clc;

pkg load image;

%spatial low pass filter

A = imread("imag4.jpeg"); B = im2double(A);

% imgd in [0,1]

B = imnoise(B,'salt & pepper',0.02);

kernel = ones(3,3)/9;

C = imfilter(single(B), kernel);

figure

subplot(221);

imshow(B);

title("With Noise")

subplot(222);

imshow(C);

title("Low-Pass Filter")

%spatial high pass filter

A = imread("img4.jpg");

B = rgb2gray(A);

kernel = -1 \* ones(3)/9;

kernel(2,2) = 8/9;

C = imfilter(single(B), kernel);

figure

subplot(121);

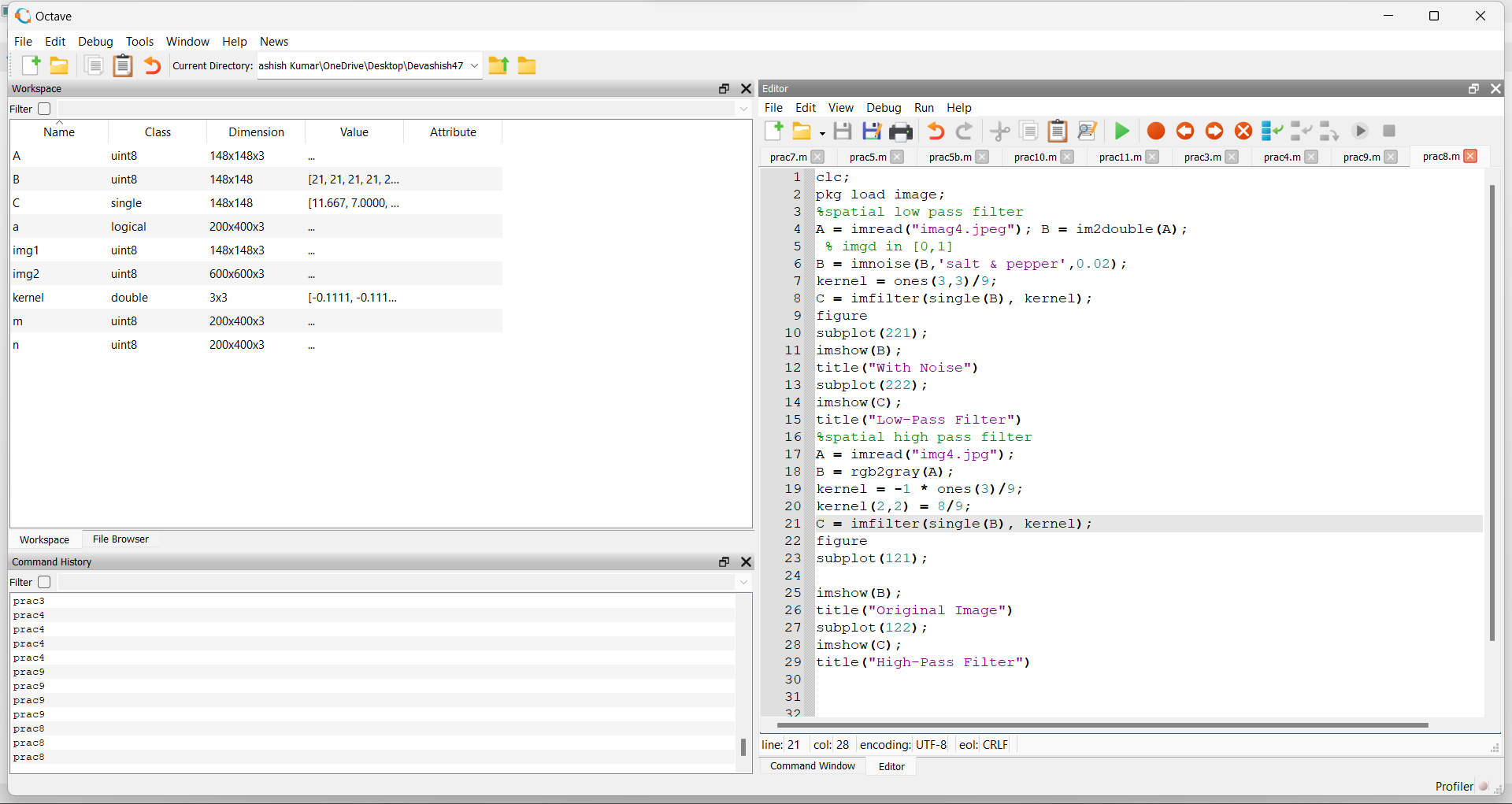
imshow(B);

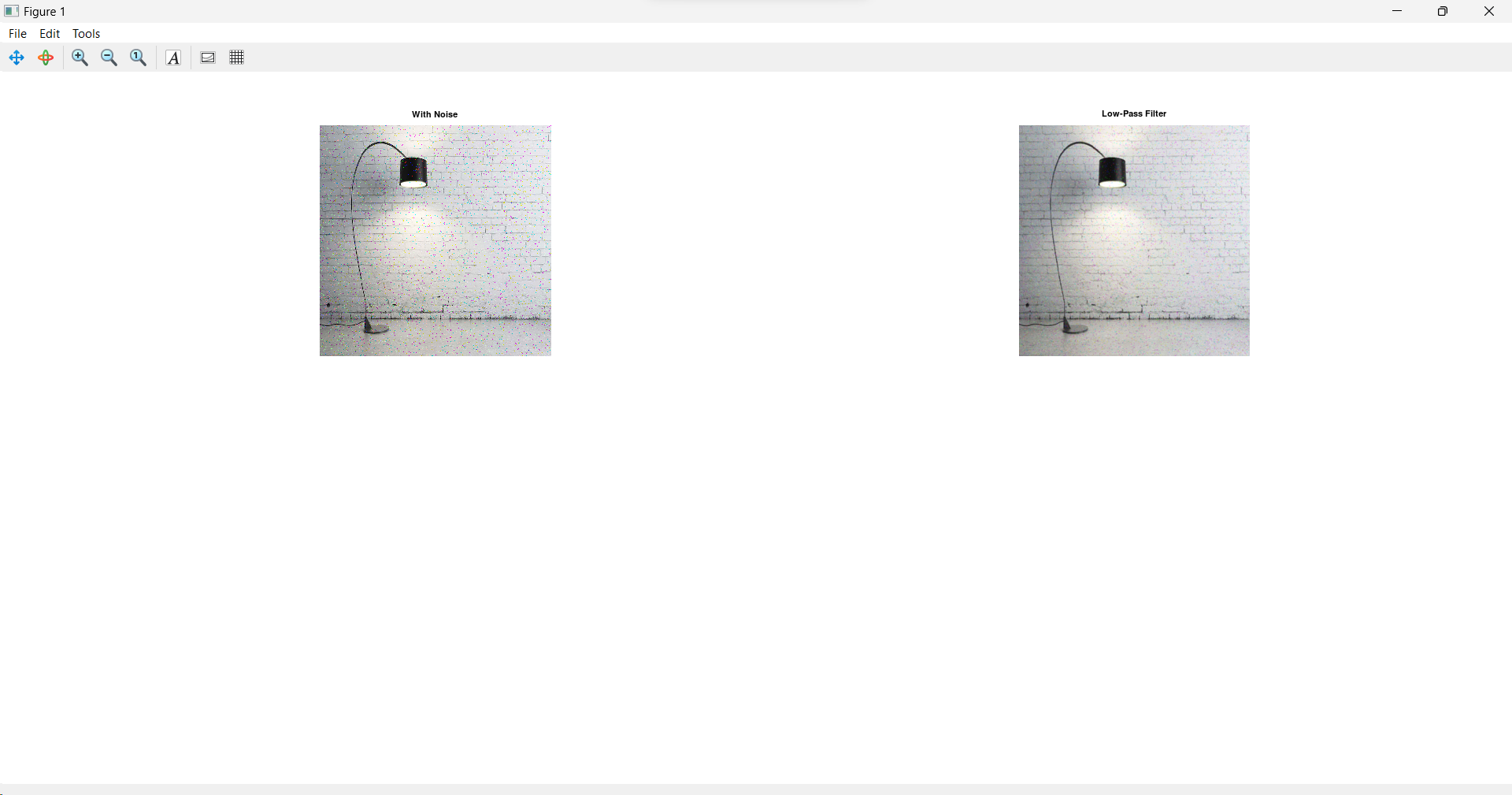
title("Original Image")

subplot(122);

imshow(C);

title("High-Pass Filter")







# Q9. Write and execute programs for image frequency domain filtering

1. **Apply FFT on given image**

# Perform low pass and high pass filtering in frequency domain

1. **Apply IFFT to reconstruct image**

clc;

pkg load image;

Im = imread('abc.jfif');

I = rgb2gray(Im);

[M, N] = size(I);

% Getting Fourier Transform of the input\_image

% using MATLAB library function fft2 (2D fast fourier transform) FT\_img = fft2(double(I));

% Assign Cut-off Frequency

D0 = 10; % one can change this value accordingly

% Designing filter u = 0:(M-1);

idx = find(u>M/2);

u(idx) = u(idx)-M;

v = 0:(N-1);

idy = find(v>N/2); v(idy) = v(idy)-N;

[V, U] = meshgrid(v, u);

% Calculating Euclidean Distance D = sqrt(U.^2+V.^2);

LP = double(D <= D0); HP = double(D > D0);

GLP = LP.\*FT\_img;

% Getting the resultant image by Inverse Fourier Transform

% of the convoluted image using MATLAB library function

% ifft2 (2D inverse fast fourier transfor LPI = real(ifft2(double(GLP)));

GHP = HP.\*FT\_img;

% Getting the resultant image by Inverse Fourier Transform

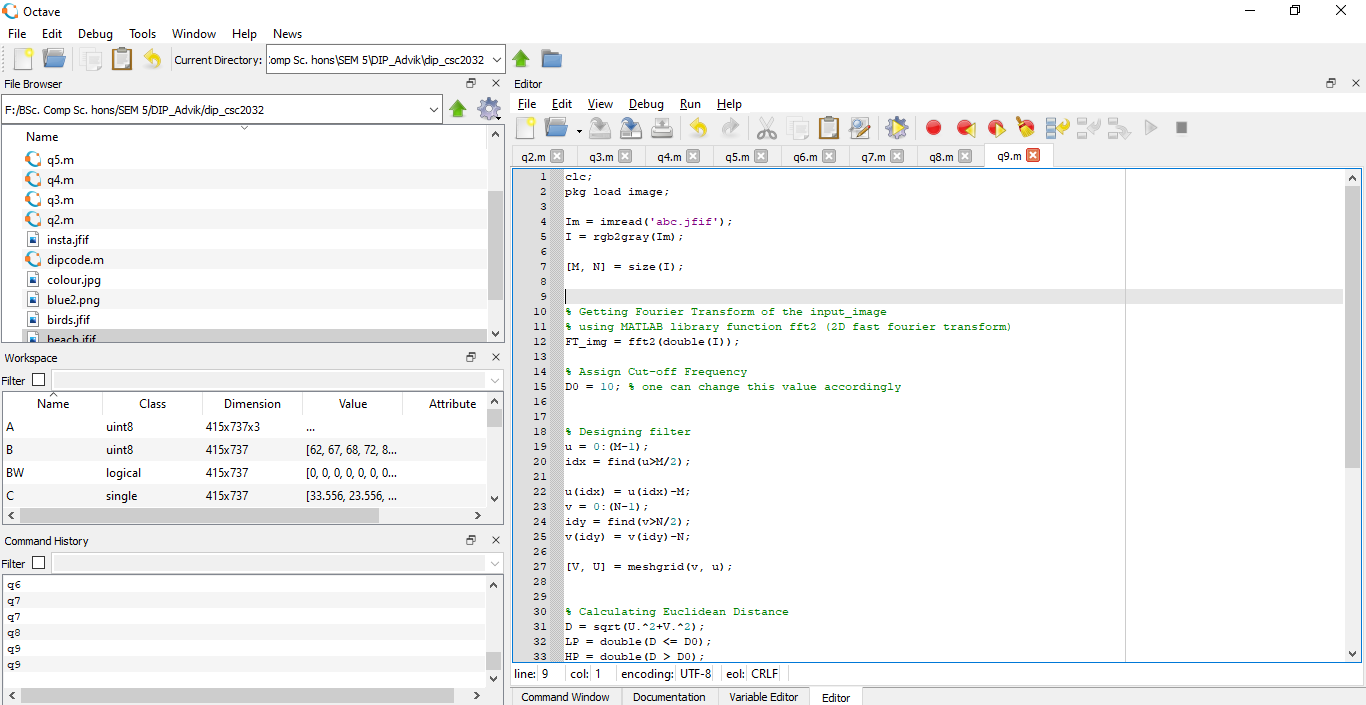
% of the convoluted image using MATLAB library function

% ifft2 (2D inverse fast fourier transform) HPI = real(ifft2(double(GHP)));

subplot(1, 3, 1), imshow(I),title('Original image');

subplot(1, 3, 2), imshow(LPI, [ ]);title('Low Pass Filter in Frequency domain');

subplot(1, 3, 3), imshow(HPI, [ ]);title('High Pass Filter in Frequency Domain');





**Q10. Write a program in C and MATLAB/SCILAB for edge detection using different edge detection mask**

Code -

clear all;

clc;

close all;

pkg load image

% Read the input image

inputImage = imread("loki.png");

% Convert the image to grayscale

grayImage = rgb2gray(inputImage);

subplot(2, 2, 1);

imshow(grayImage);

title('Original Image');

% Apply different edge detection masks

sobelEdge = edge(grayImage, 'sobel');

subplot(2, 2, 2);

imshow(sobelEdge);

title('Sobel Edge Detection');

prewittEdge = edge(grayImage, 'prewitt');

subplot(2, 2, 3);

imshow(prewittEdge);

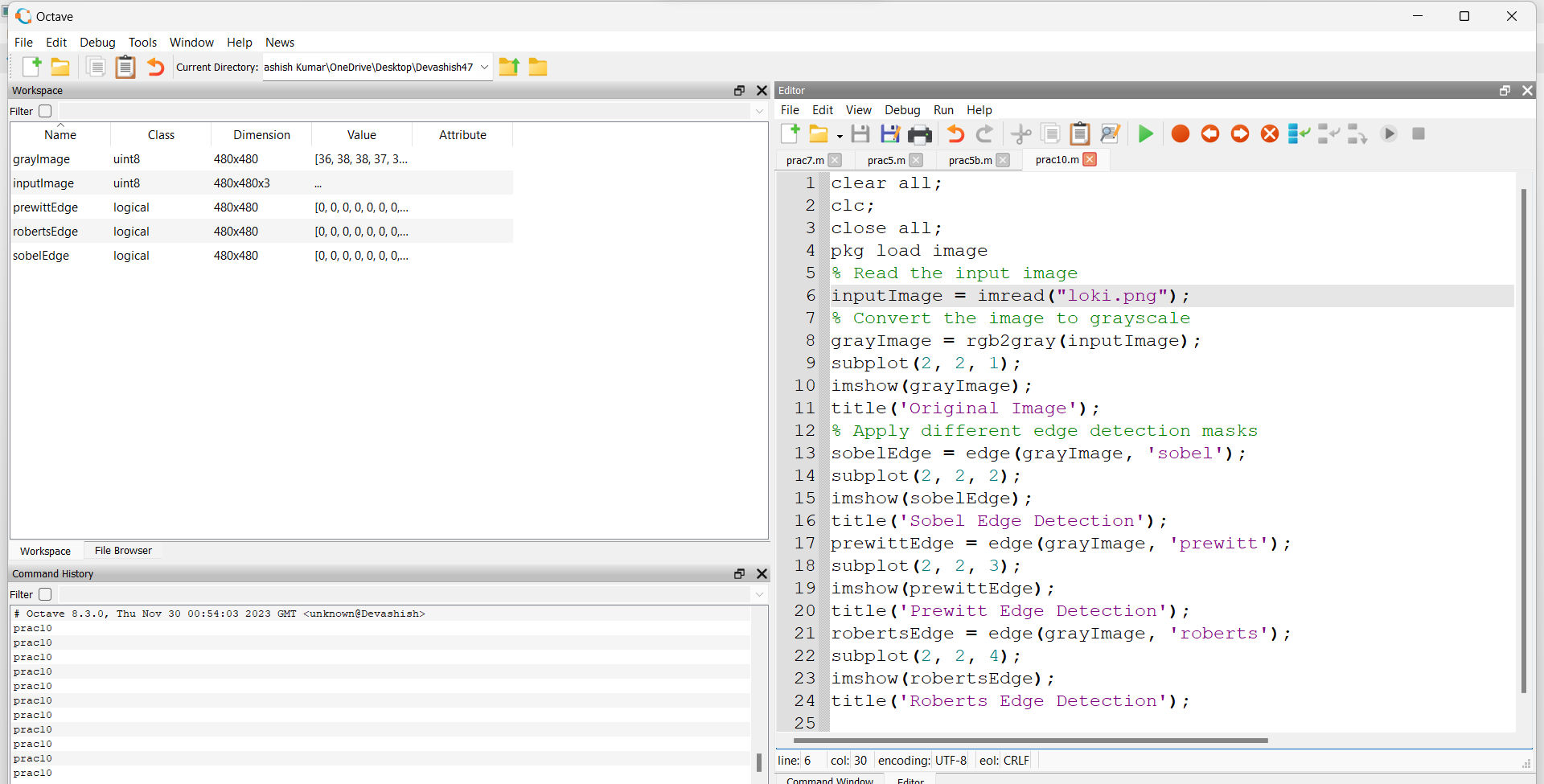
title('Prewitt Edge Detection');

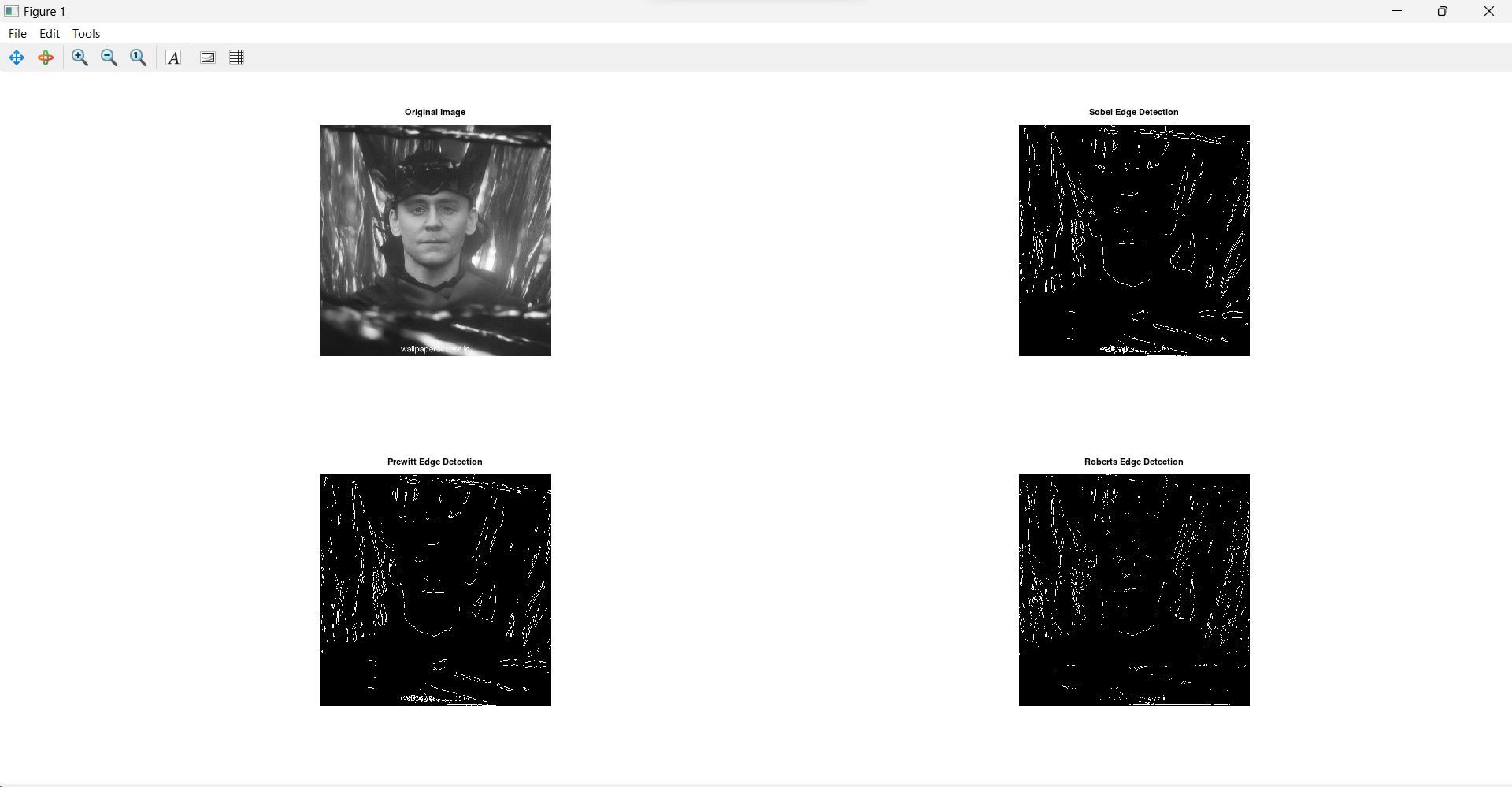
robertsEdge = edge(grayImage, 'roberts');

subplot(2, 2, 4);

imshow(robertsEdge);

title('Roberts Edge Detection');





**Q11. Write and execute program for image morphological operations erosion and dilation.**

Code -

clear all;

clc;

close all;

pkg load image

%11

x=imread('Dark.jpg');

subplot(2,2,1);

imshow(x);

title("Original image");

se = strel('square',3);

eroded = imerode(x,se);

subplot(2,2,2);

imshow(eroded);

title("Erosion");

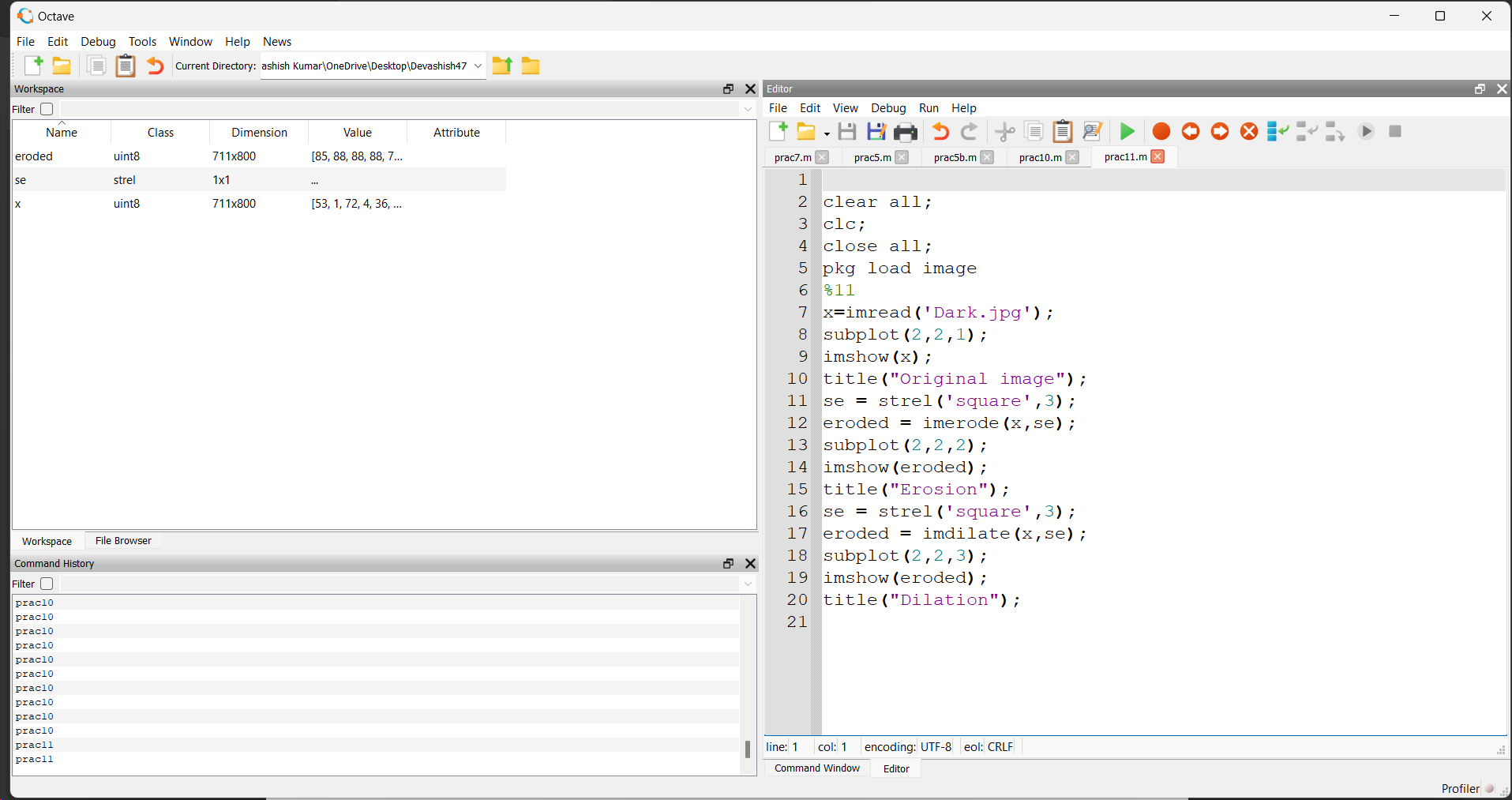
se = strel('square',3);

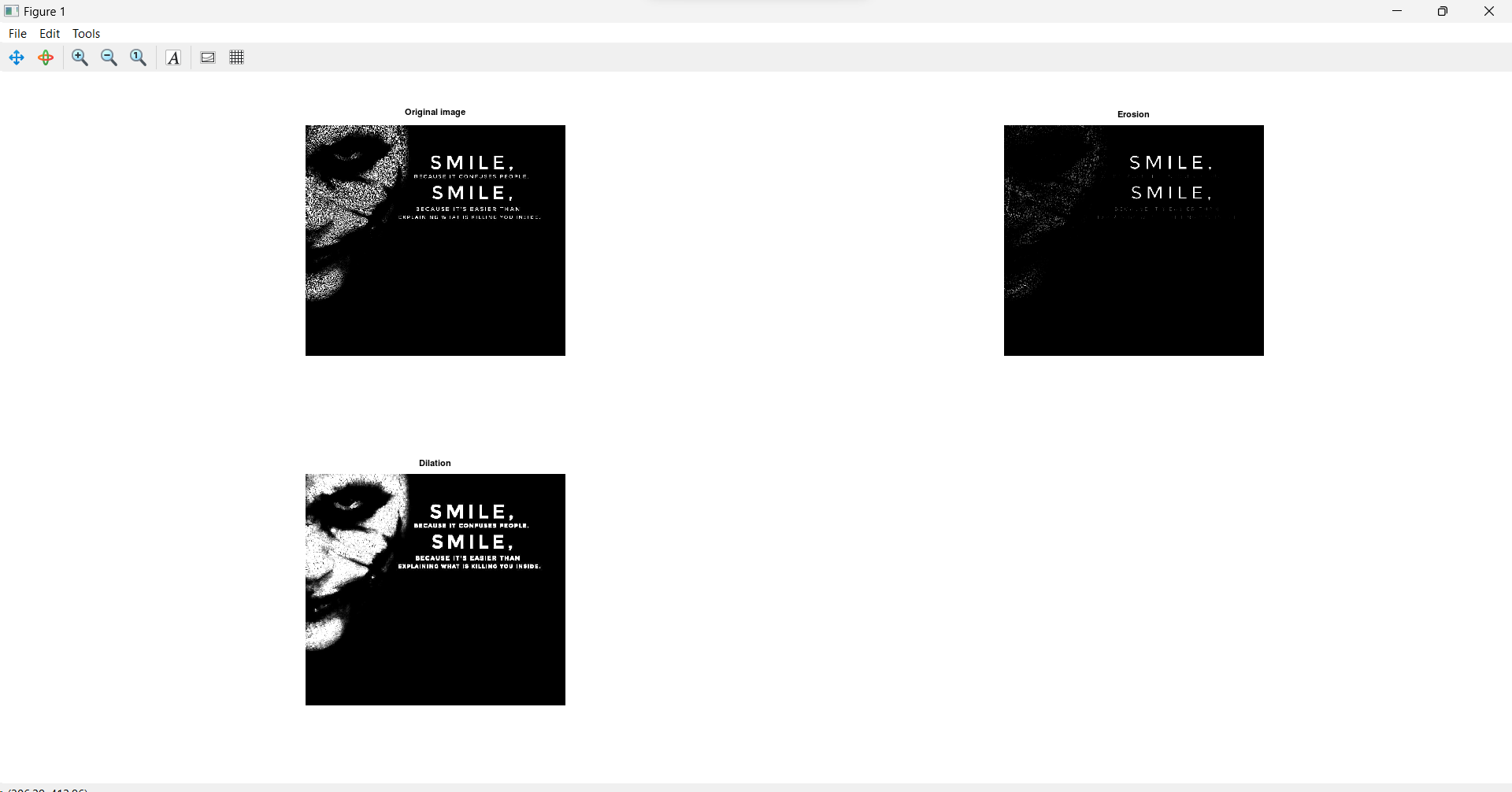
eroded = imdilate(x,se);

subplot(2,2,3);

imshow(eroded);

title("Dilation");





**THE END**