****

**RAMNIRANJAN JHUNJHUNWALA COLLEGE**

**GHATKOPAR (W), MUMBAI - 400 086**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**2020 - 2021**

**M.Sc. (I.T.) SEM I**

**Image Vision and Processing**

**Name: Manjunath V Kareru**

**Roll No.: 22**



**CERTIFICATE**

This is to certify that Miss. **Manjunath V Kareru** with Roll No. **22** has successfully completed the necessary course of experiments in the subject of **Image vision and processing** during the academic year **2020 – 2021** complying with the requirements of **RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE AND COMMERCE**, for the course of **M.Sc. (IT) semester -I**.

Date: -

Internal Examiner

External

College Seal

Head of Department

**Index**

|  |  |  |
| --- | --- | --- |
| **NO.** | PRACTICAL | PAGE |
| 1. | Implement Basic Intensity transformation functions   1. A) Image Inverse 2. B) Log Transformation 3. C) Power-law Transformation |  |
| 2. | Piecewise Transformation   1. A) Contrast Stretching 2. B) Thresholding 3. C) Bit-Plane Slicing |  |
| 3. | Implement Histogram Equalization |  |
| 4. | Image filtering in Spatial Domain   1. A) Low-pass Filter/Smoothing Filters   (Average, Weighted Average, Median and Gaussian)   1. B) High-pass Filter / Sharpening Filter   (Laplacian Filter, Sobel, Robert and Prewitt Filter to detect edge) |  |
| 5. | Analyze image in Frequency Domain   1. A) Low Pass/Smoothing filter 2. B) High Pass/Sharpening filter |  |
| 6. | Color Image Processing   1. A) Pseudocoloring 2. B) Separating the RGB Channels 3. C) Color Slicing |  |
| 7. | Image Compression Techniques and watermarking  A) Implement Huffman Coding  B) Add a watermark to the image |  |
| 8. | Basic Morphological Transformations  A) Boundary Extraction  B) Thinning and Thickening  C) Hole filling and Skeletons |  |

**Practical 1: Implement Basic Intensity transformation functions**

1. **Image Inverse:** The negative or inverse of an image with intensity levels in the range [0, L-1] is obtained by using the negative transformation, which is given by the expression,

S = L – 1 – r

Where L – 1 (Maximum pixel value)

r (Pixel of an image)

**Code: -**

#Practical 1: Implement Basic Intensity transformation functions

pkg load image;

clear all;

close all;

image= imread('baby.jpg');

image1 = rgb2gray(image);

image1 = im2double(image1);

[row col] = size(image1);

for i = 1:row

for j = 1:col

N(i,j)=1-image1(i,j);

endfor

endfor

figure

imshow(image1);

title("Original Inage");

figure

imshow(N);

title("Negative Transformation Inage");

****

**Output: -**

** **

**Practical 1(b) -**

**Log Transformation:**

Practical 1(b) - Log Transformation: The log transformation maps a narrow range of low intensity values in the input into a wider range of output levels. We use the transformation if this type to extend the values of dark pixel in an image while compress the higher-level values.

**Code: -**

pkg load image;

clear all;

close all;

img = imread('airplane.png');

img = rgb2gray(img);

img = im2double(img);

[row col] = size(img);

c=3;

for i = 1:row

for j = 1:col

N(i,j) = c\*log(1+img(i,j));

endfor;

endfor;

figure

imshow(img);

title('Original Image');

figure

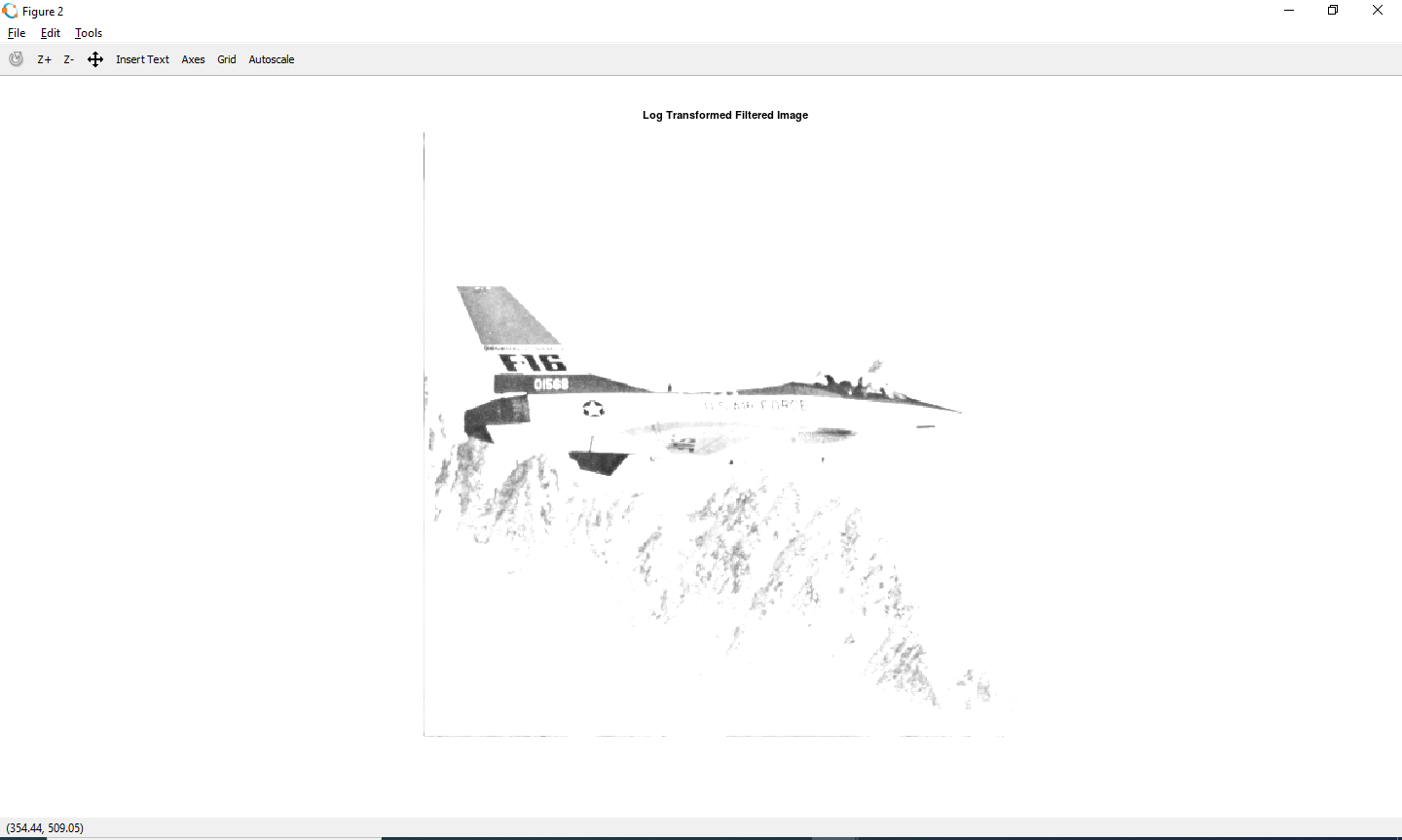
imshow(N);

title('Log Transformed Filtered Image');



Output:-





**Practical No. 1 ( c ) Power-Law Transformation:**

#Practical No. 1 ( c ) Power-Law Transformation:

**Code: -**

pkg load image;

clear all;

close all;

img = imread('cameraman.png');

#img = rgb2gray(img);

img = im2double(img);

[row col] = size(img);

gamma = 2;

c = 2;

for i = 1:row

for j = 1: col

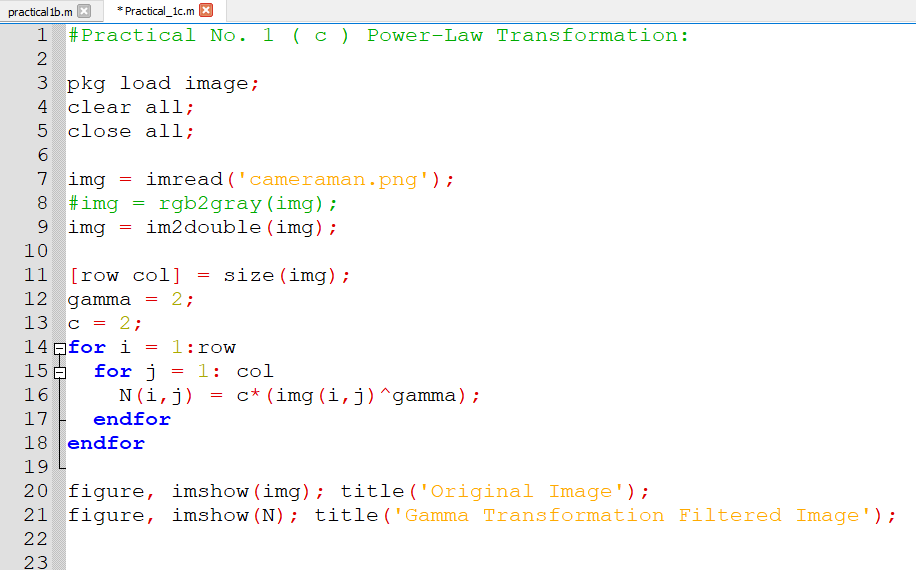
N(i,j) = c\*(img(i,j)^gamma);

endfor

endfor

figure, imshow(img); title('Original Image');

figure, imshow(N); title('Gamma Transformation Filtered Image');



Output:





**Practical 2: Piecewise Transformation**

**Practical 2(a)-** Contrast Stretching

**Code: -**

pkg load image;

clear all;

close all;

ogimg=imread('fields.jpg');

#oggray = rgb2gray(ogimg)

ogd = im2double(ogimg);

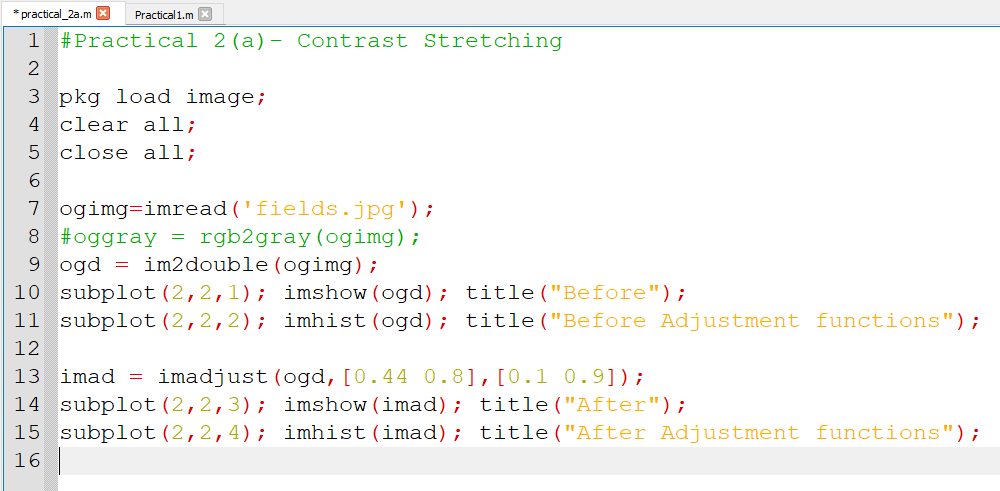
subplot(2,2,1); Imshow(ogd); title("Before");

subplot(2,2,2); imhist(ogd); title("Before Adjustment functions");

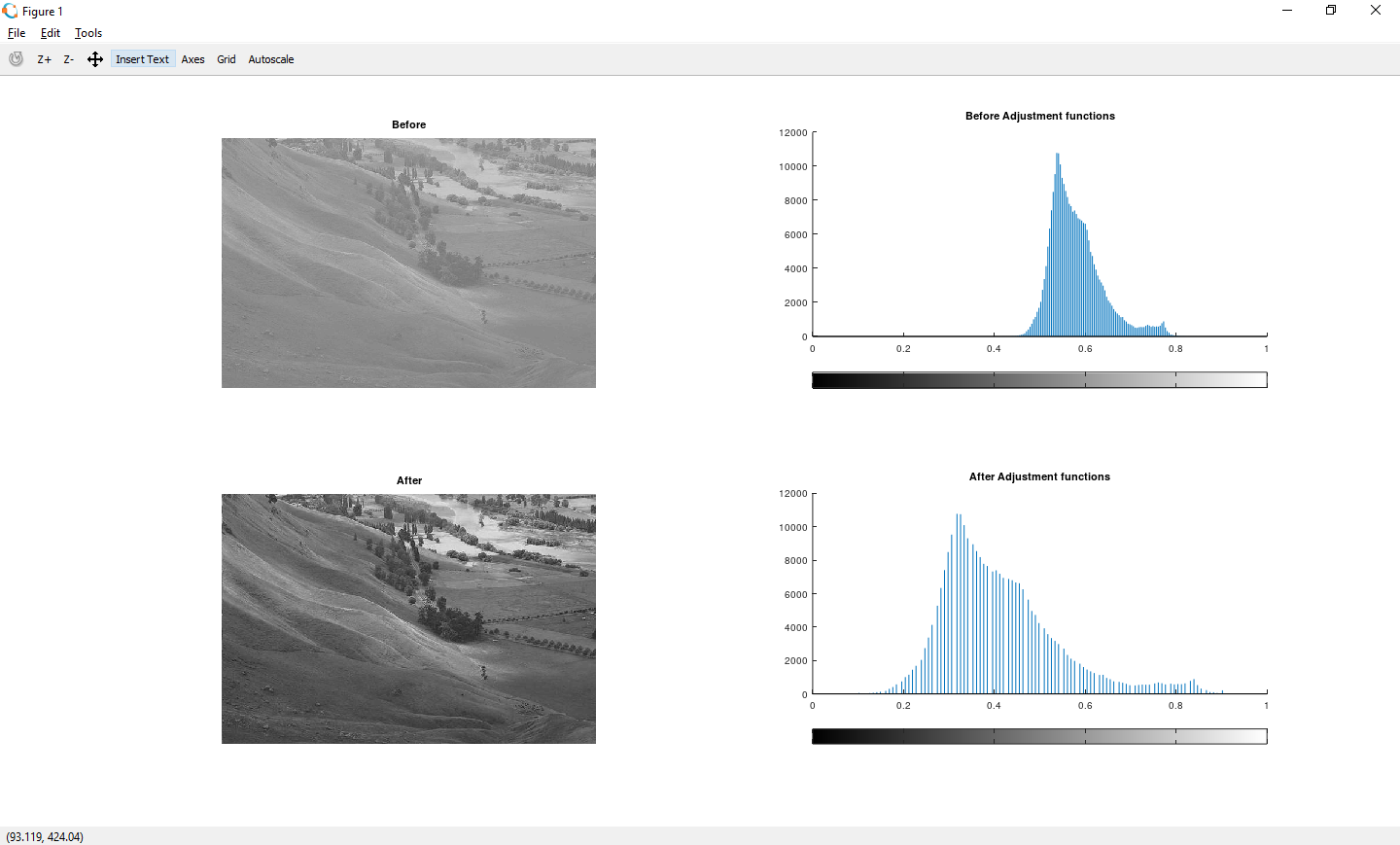
imad = imadjust(ogd,[0.44 0.8],[0.1 0.9]);

subplot(2,2,3); imshow(imad); title("After");

subplot(2,2,4); imhist(imad); title("After Adjustment functions");



**Output:**



**Practical 2(a)- Contrast Stretching using user input**

**Code: -**

#Practical 2(a)- Contrast Stretching Using inputs from user r1,r2,s1,s2

pkg load image;

clear all;

close all;

img = imread('fruits.png');

imgray = rgb2gray(img);

imdouble = im2double(imgray);

[m n] = size(imdouble); # getting dimension of image-imgdouble

# getting inputs from user (r1,r2,s1,s2)

r1 = input ("Enter R1:")

r2 = input ("Enter R2:")

s1 = input ("Enter S1:")

s2 = input ("Enter S2:")

#Calculation of contrast stretching

a = s1/s2;

b = (s2-s1)/(r2-r1);

c = (255-s2)/(255-r2);

for i=1:m

for j=1:n

if imgray(i,j) < r1

s(i,j) = a\*imgray(i,j);

elseif imgray(i,j) < r2

s(i,j) = b\*(imgray(i,j)-r1)+s1;

else

s(i,j) = c\*(imgray(i,j)-r2)+s2;

endif

endfor

endfor

#Displaying the Original and Contrast Images

figure(3)

subplot(1,2,1)

imshow(img);

title('Original Image');

subplot(1,2,2)

imhist(imgray);

title('Histogtram Of Original Image');

figure(4);

subplot(1,2,1)

imshow(s);

title("contrast Streached Image");

subplot(1,2,2)

imhist(s);

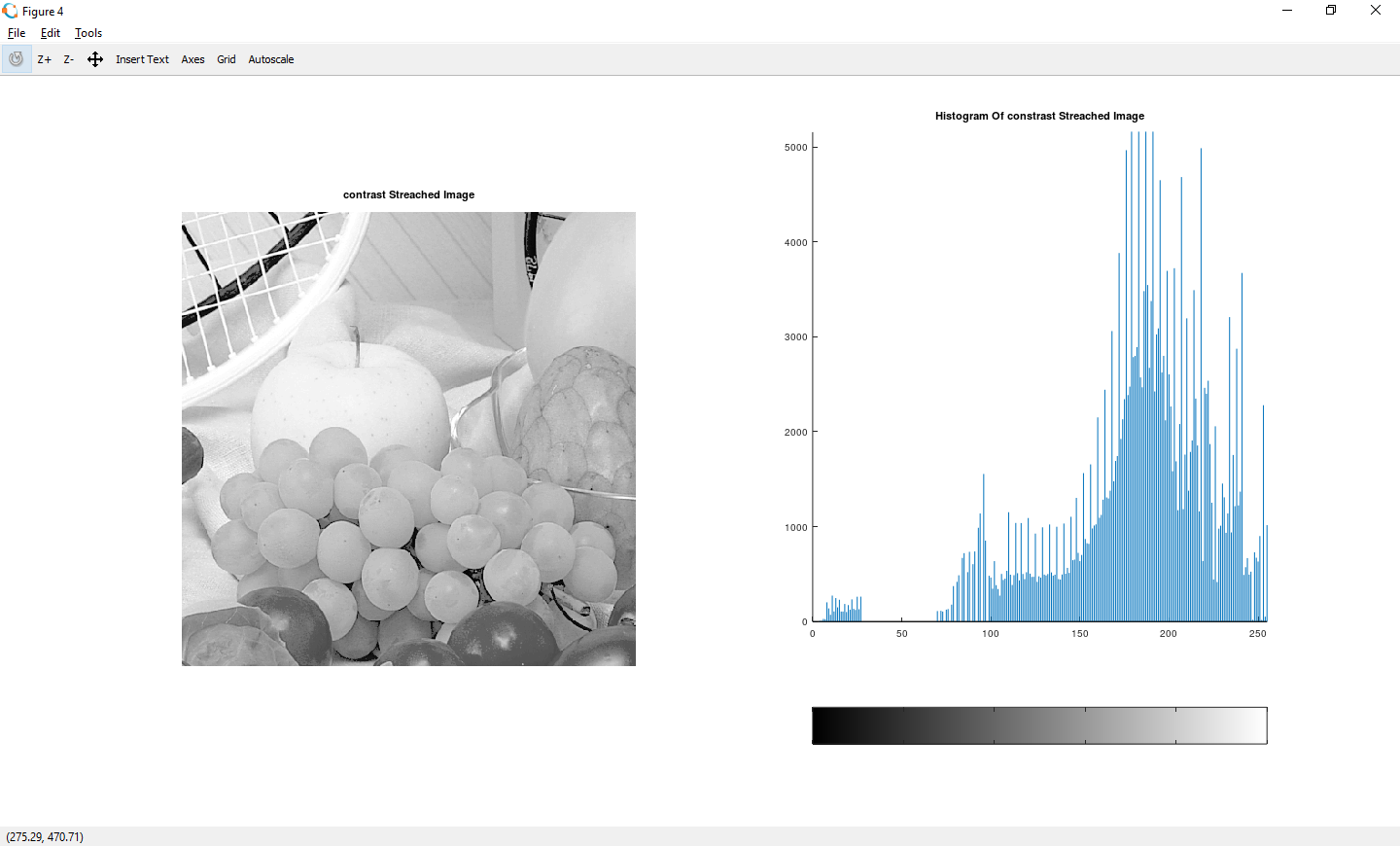
title("Histogram Of constrast Streached Image");





**Output:-**





**Practical 2(b)- Thresholding**

**Code:**

#Practical 2(a)- Contrast Stretching

pkg load image;

clear all;

close all;

img = imread('fruits.png');

imgray = rgb2gray(img);

imgdouble = im2double(imgray);

imhist(imgray);

thr=150;

[m n] = size(imgray);

s = zeros(m,n);

for i = 1:m

for j = 1:n

if(imgray(i,j))> thr

s(i,j) = 1;

else

s(i,j) = 0;

endif

endfor

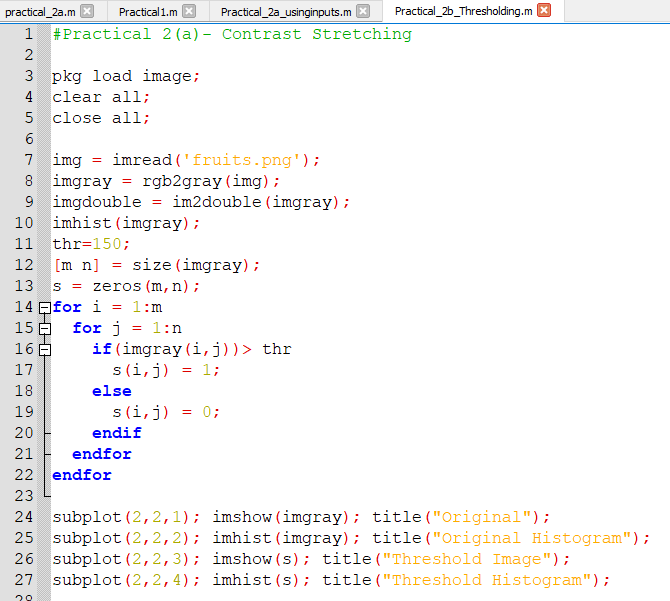
endfor

subplot(2,2,1); imshow(imgray); title("Original");

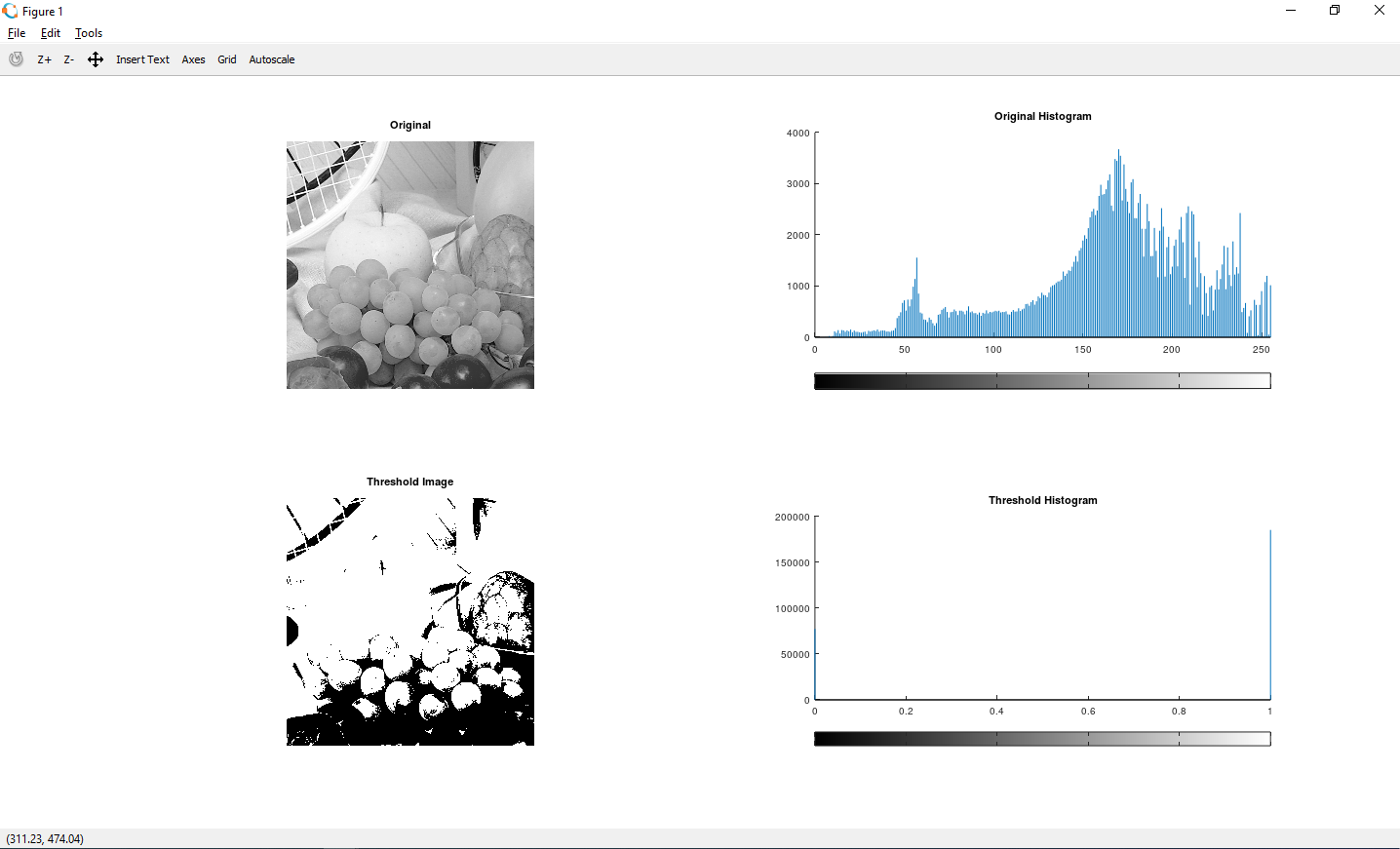
subplot(2,2,2); imhist(imgray); title("Original Histogram");

subplot(2,2,3); imshow(s); title("Threshold Image");

subplot(2,2,4); imhist(s); title("Threshold Histogram");



**Output:-**



**Practical -2 ( C )** **Image reconstruction using n bit planes**

**Code: -**

#Practical -2 ( C )- Image reconstruction using n bit planes.

pkg load image;

clear all;

close all;

img = imread('doller.png');

g = rgb2gray(img);

B = zeros(size(g));

g1 = bitget(g,1);

g2 = bitget(g,2);

g3 = bitget(g,3);

g4 = bitget(g,4);

g5 = bitget(g,5);

g6 = bitget(g,6);

g7 = bitget(g,7);

g8 = bitget(g,8);

figure,

subplot(2,2,1), imshow(logical(g1)); title('Bit 1');

subplot(2,2,2), imshow(logical(g2)); title("Bit 2");

subplot(2,2,3), imshow(logical(g3)); title('Bit 3');

subplot(2,2,4), imshow(logical(g4)); title('Bit 4');

figure,

subplot(2,2,1), imshow(logical(g5)); title('Bit 5');

subplot(2,2,2), imshow(logical(g6)); title("Bit 6");

subplot(2,2,3), imshow(logical(g7)); title('Bit 7');

subplot(2,2,4), imshow(logical(g8)); title('Bit 8');

B=bitset(B,5,g5);

B=bitset(B,6,g6);

B=bitset(B,7,g7);

B=bitset(B,8,g8);

B=uint8(B);

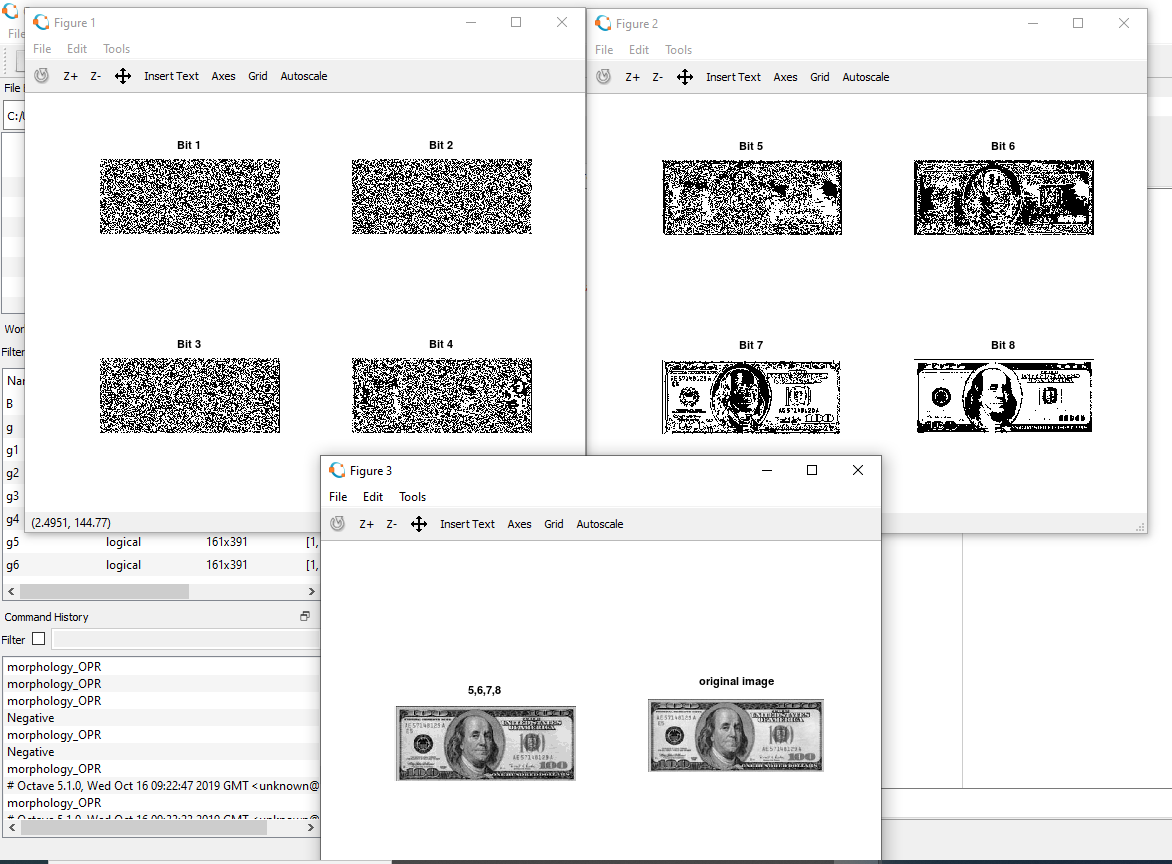
figure,

subplot(1,2,1),imshow(B); title("5,6,7,8")

subplot(1,2,2),imshow(g); title("original image");



Output:-



**Practical-3 Histogram equalization without using histeq() function**

**Code: -**

#Practical 3:- Histogram equalization without using histeq() function.

pkg load image;

clear all;

close all;

img=imread('fields.jpg');

#img=rgb2gray(a);

#img=img(1:10,1:10)

#r=size(img,1);

#c=size(img,2);

[r c] = size(img);

ah=uint8(zeros(r,c));

n=r\*c;

f=zeros(256,1);

pdf=zeros(256,1);

cdf=zeros(256,1);

cumm=zeros(256,1);

out=zeros(256,1);

for i=1:r

for j=1:c

values=img(i,j);

f(values+1)=f(values+1)+1;

pdf(values+1)=f(values+1)/n;

endfor

endfor

sum=0; L=255; size(pdf);

for i=1:size(pdf)

sum=sum+f(i);

cum(i)=sum;

cdf(i)=cum(i)/n;

out(i)=round(cdf(i)\*L);

endfor

for i=1:r

for j=1:c

ah(i,j)=out(img(i,j)+1);

endfor

endfor

figure,

subplot(2,2,1), imshow(img); title('original image');

subplot(2,2,2), imhist(img); title('original hist');

#he=histeq(a);

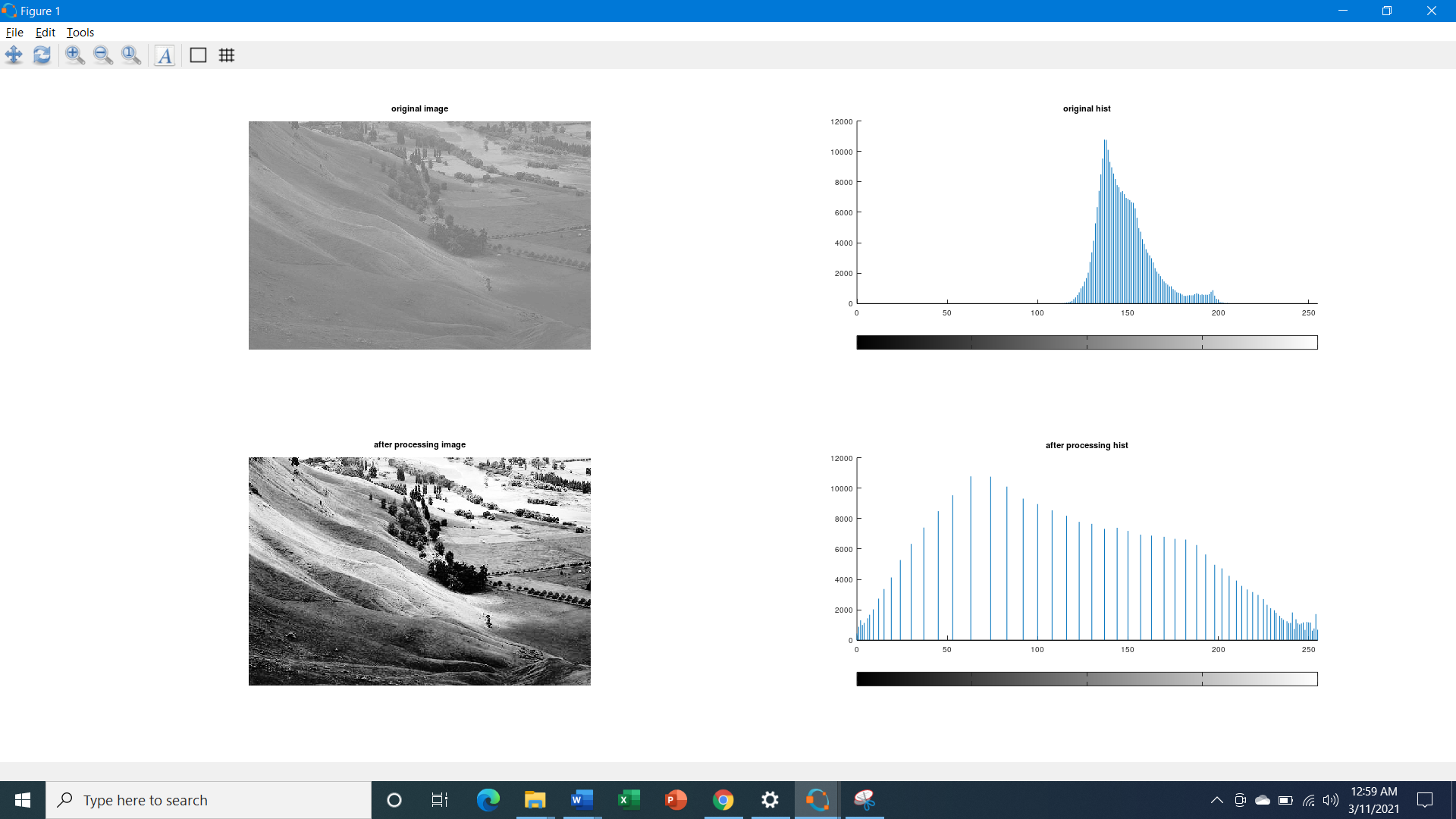
subplot(2,2,3), imshow(ah); title('after processing image');

subplot(2,2,4), imhist(ah); title('after processing hist');

#imhist(he);

|  |  |
| --- | --- |
|  |  |

**Output:-**



**Practical 4: Image filtering in Spatial Domain**

1. **Low pass filter/ Smoothing filters.**
2. **Low pass – Average Filter**

**Code: -**

#Practical4\_A\_Lowpass-Average Filter

pkg load image;

clear all;

close all;

img=imread('hawk1.png');

#img=rgb2gray(img);

#imwrite(img,'hawk1.png');

img=im2double(img);

r=imnoise(img,'salt & pepper' );

f=ones(3,3)/9;

af=filter2(f,r);

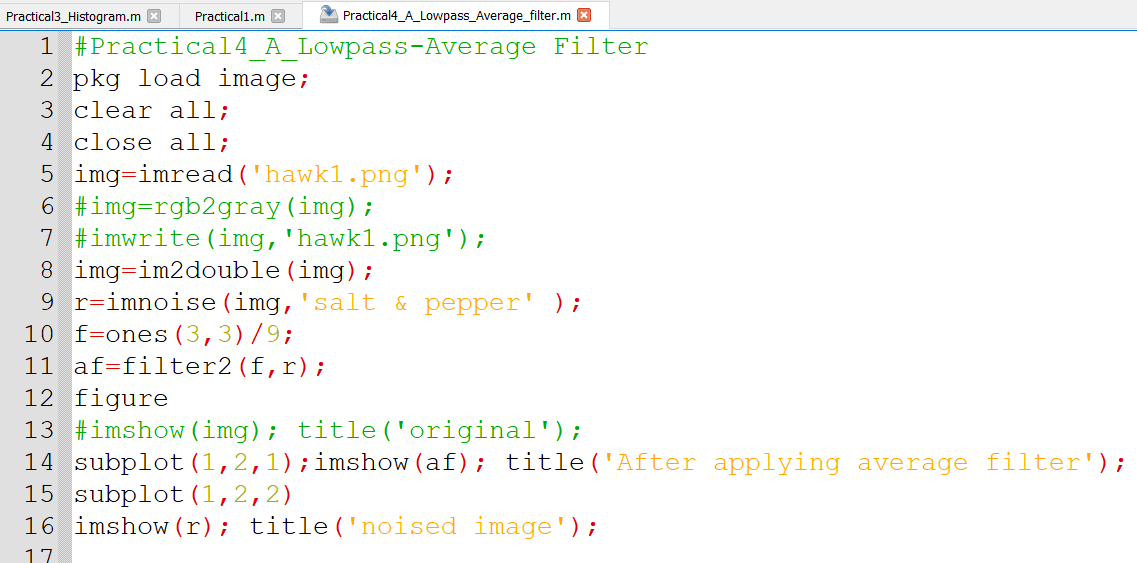
figure

#imshow(img); title('original');

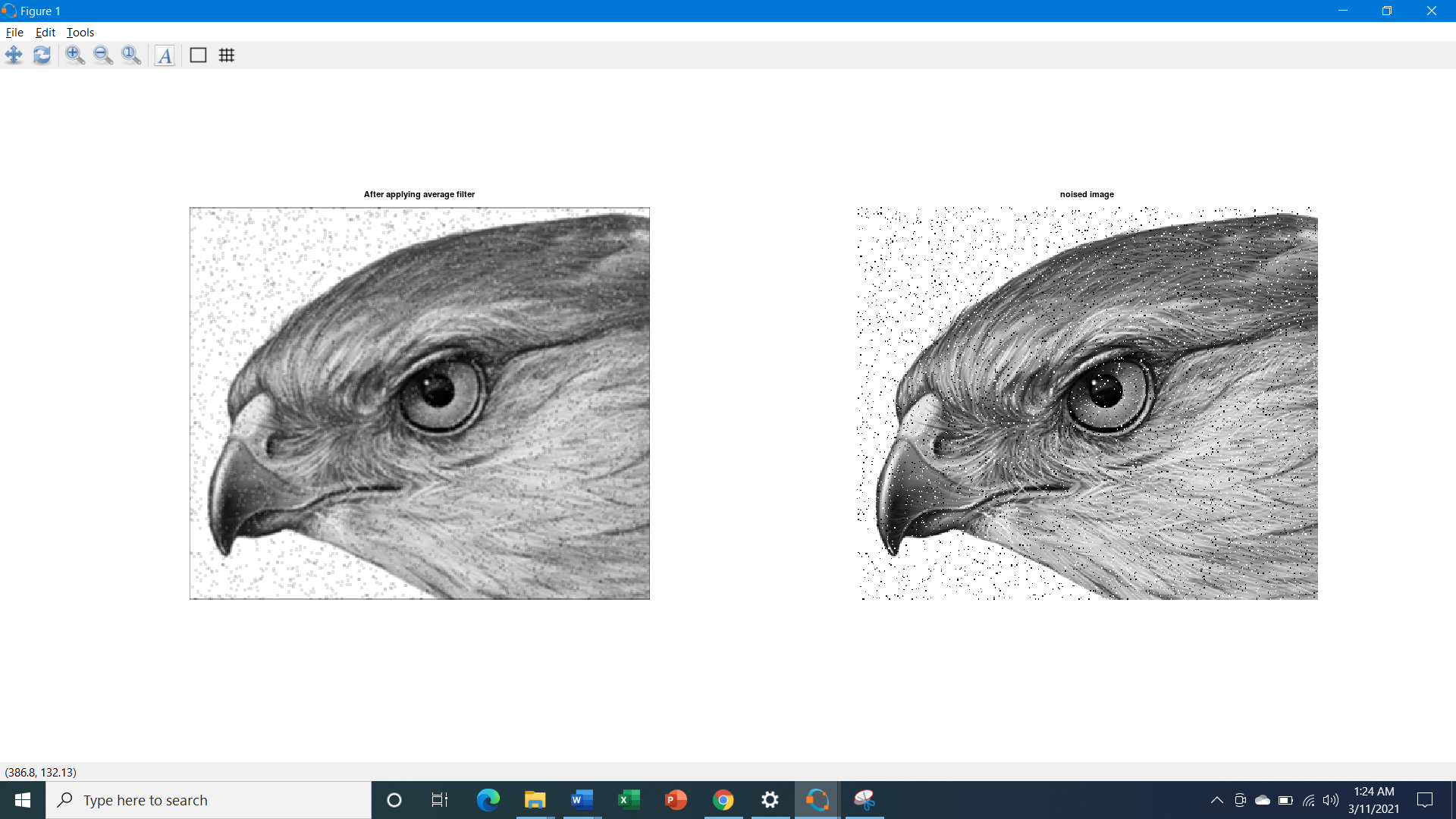
subplot(1,2,1);imshow(af); title('After applying average filter');

subplot(1,2,2)

imshow(r); title('noised image');



Output:



1. **Low pass – Median Filter**

**Code: -**

#Practical4\_A\_Lowpass-Median Filter

pkg load image;

clear all;

close all;

img=imread('hawk1.png');

img\_noisy1=imnoise(img,'salt & pepper' );

[row, col] = size(img\_noisy1);

img\_new1 = zeros(row, col);

for i=2: row-1

for j =2: col-1

temp = [img\_noisy1(i-1, j-1),

img\_noisy1(i-1, j),

img\_noisy1(i-1, j + 1),

img\_noisy1(i, j-1),

img\_noisy1(i, j),

img\_noisy1(i, j + 1),

img\_noisy1(i + 1, j-1),

img\_noisy1(i + 1, j),

img\_noisy1(i + 1, j + 1)] ;

temp = sort(temp);

img\_new1(i, j)= temp(4);

endfor

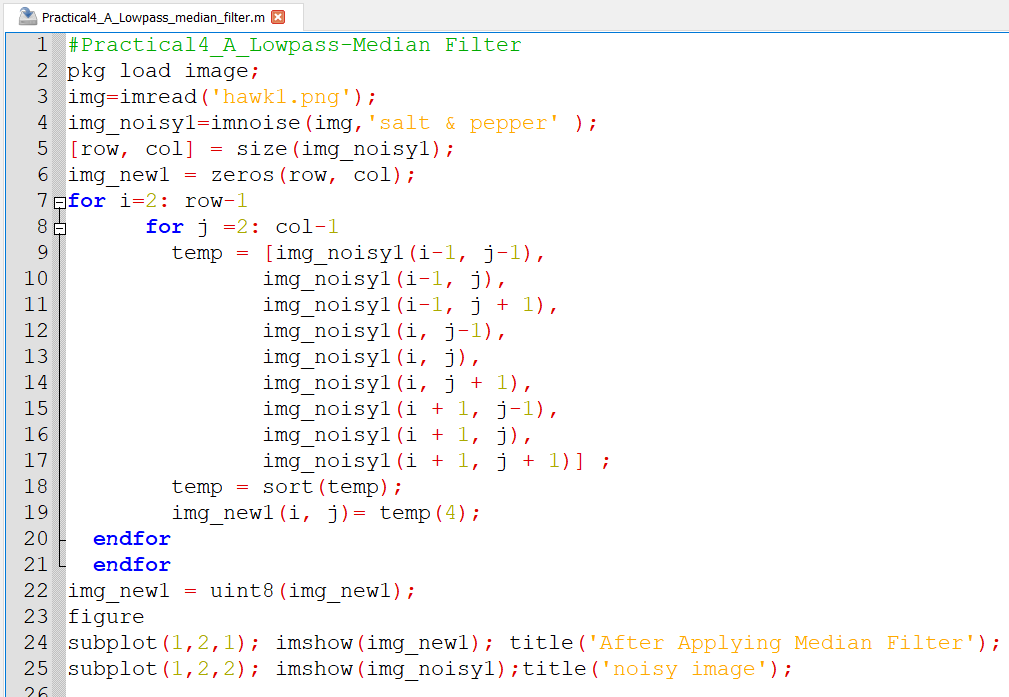
endfor

img\_new1 = uint8(img\_new1);

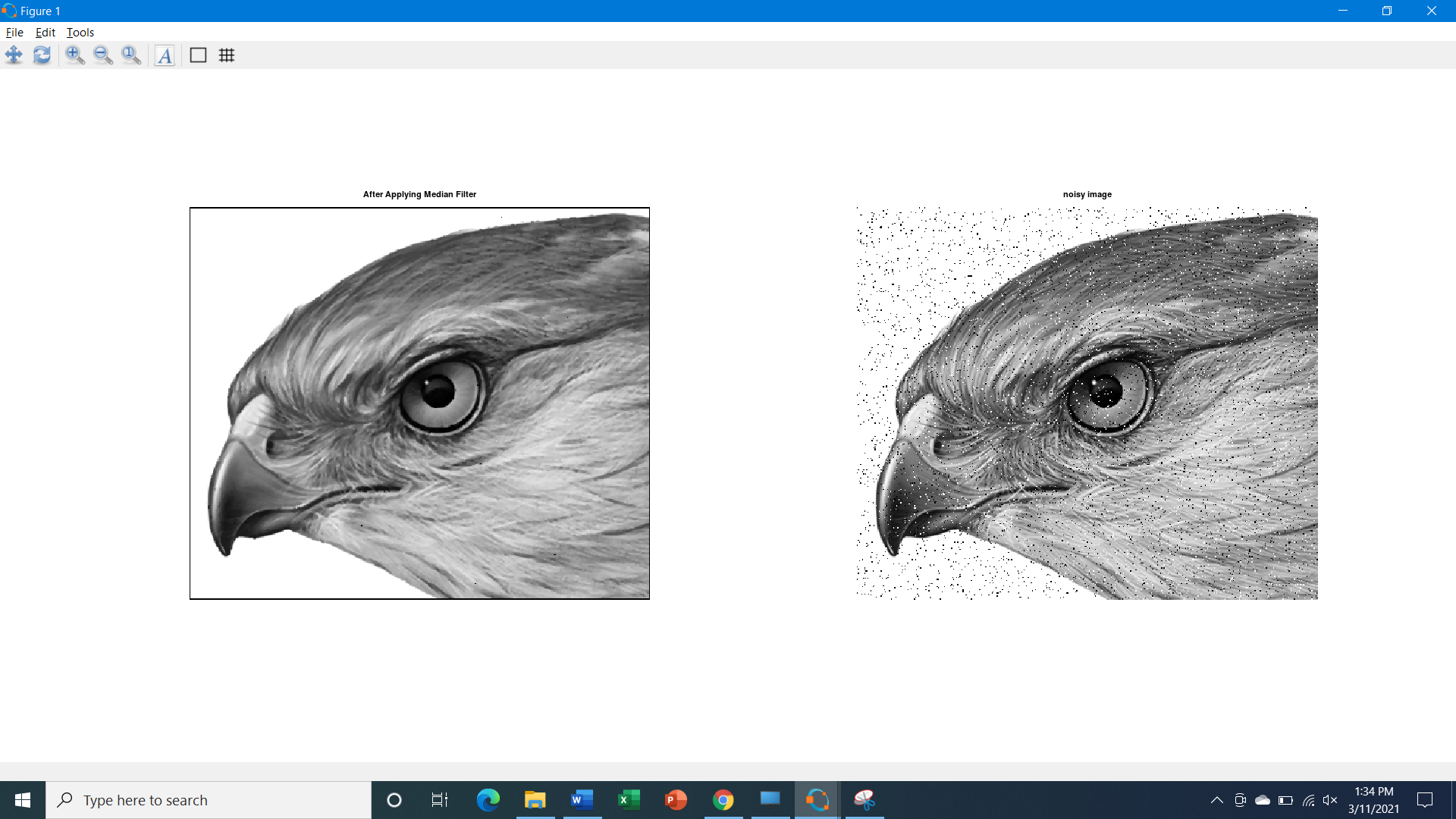
figure

subplot(1,2,1); imshow(img\_new1); title('After Applying Median Filter');

subplot(1,2,2); imshow(img\_noisy1);title('noisy image');



Output:



1. **High pass filters/ Sharpening filters**
2. **High-pass filter- The Laplacian Filter.**

**Code: -**

#Practical4\_B\_Highpass\_The Laplacian Filter

pkg load image;

clear all;

close all;

img=imread('coins.png');

size(img);

figure,

subplot(2,2,1);imshow(img); title('original Image');

%Preallocate the matrices with zeros

I1=img;

I=zeros(size(img));

I2=zeros(size(img));

%Filter Masks

F1=[0 2 0;2 -8 2; 0 2 0];

#F2=[1 1 1;1 -8 1; 1 1 1];

%Padarray with zeros

img=padarray(img,[1,1]);

img=double(img);

size(img);

%Implementation of the equation in Fig.D

for i=1:size(img,1)-2

for j=1:size(img,2)-2

I(i,j)=sum(sum(F1.\*img(i:i+2,j:j+2)));

end

end

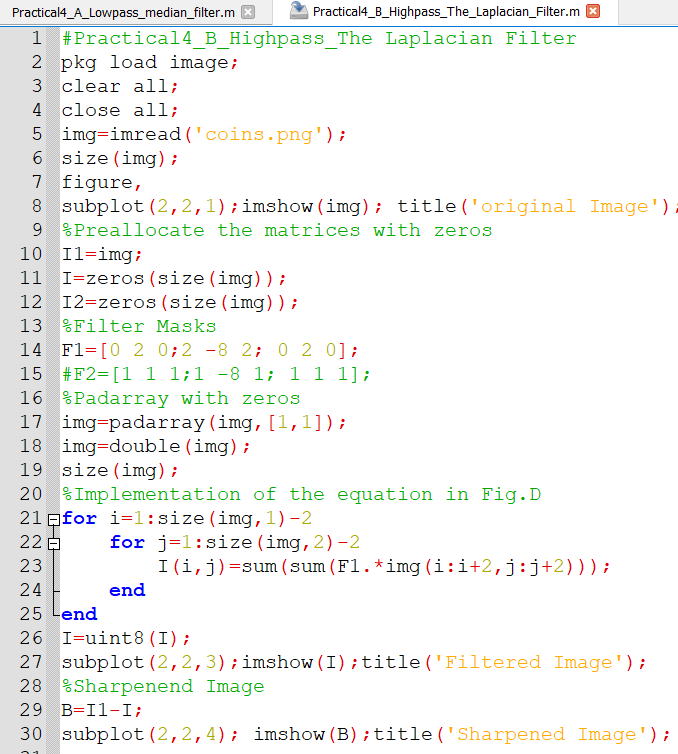
I=uint8(I);

subplot(2,2,3);imshow(I);title('Filtered Image');

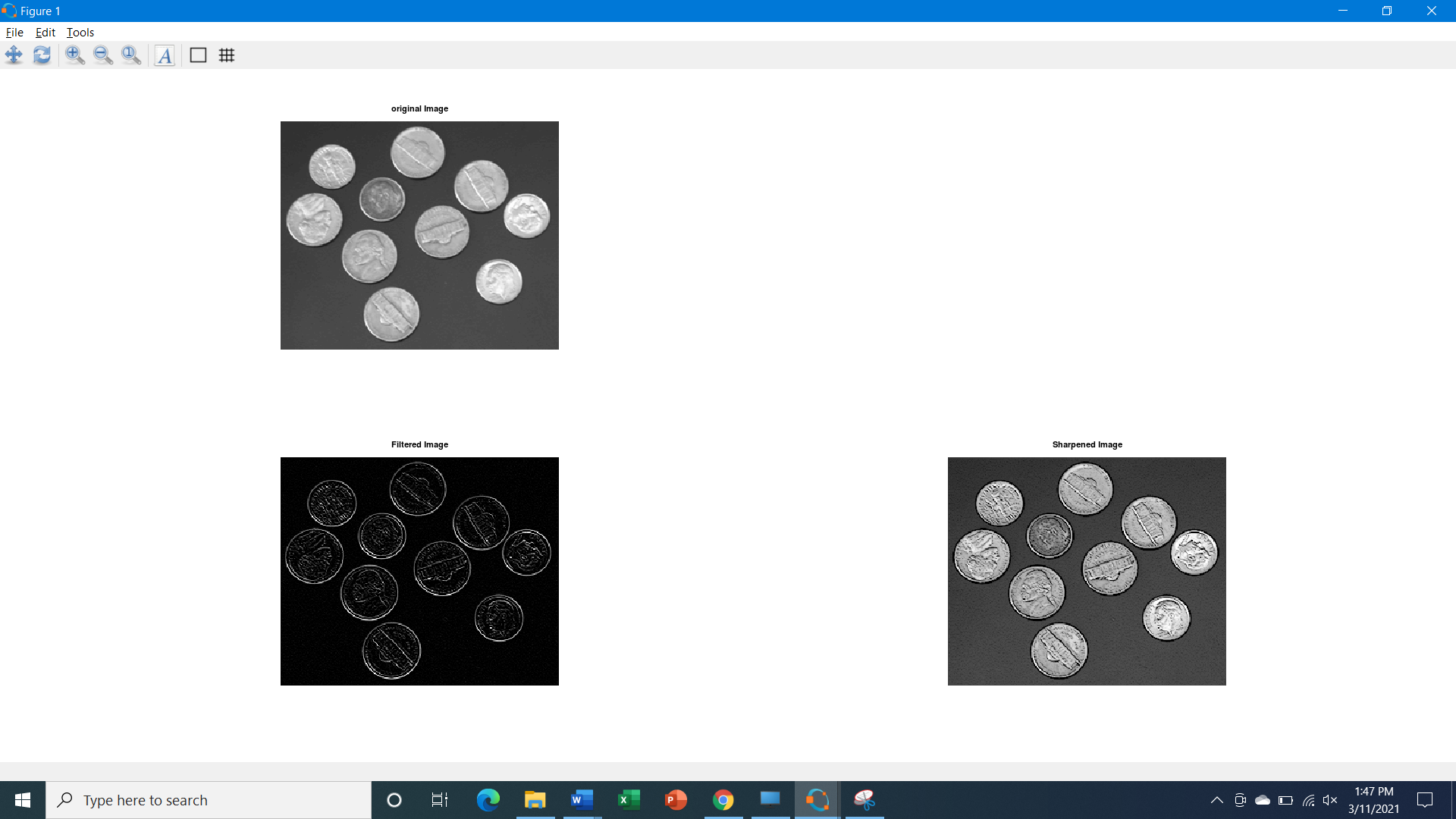
%Sharpenend Image

B=I1-I;

subplot(2,2,4); imshow(B);title('Sharpened Image');



Output:



1. **High-pass filter – Sobel, Robert, Prewitt Operator.**

**Code: -**

#Practical4\_B\_Highpass\_Sobel\_Robert\_Prewitt Operator for detecting Edges

pkg load image;

clear all;

close all;

img=imread("peppers.png");

sobel = edge(img,'Sobel');

figure 1,

subplot(2,2,1); imshow(img); title('Original Image');

subplot(2,2,2); imshow(sobel); title("Edge detection using sobel filter");

robert = edge(img,'Roberts');

prewitt = edge(img,'Prewitt');

subplot(2,2,3)

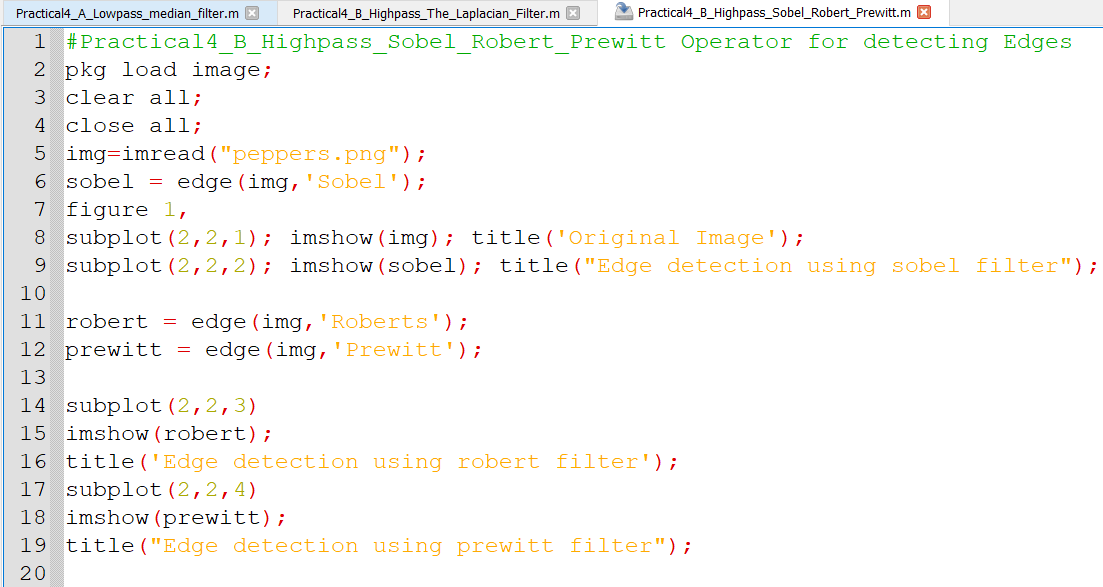
imshow(robert);

title('Edge detection using robert filter');

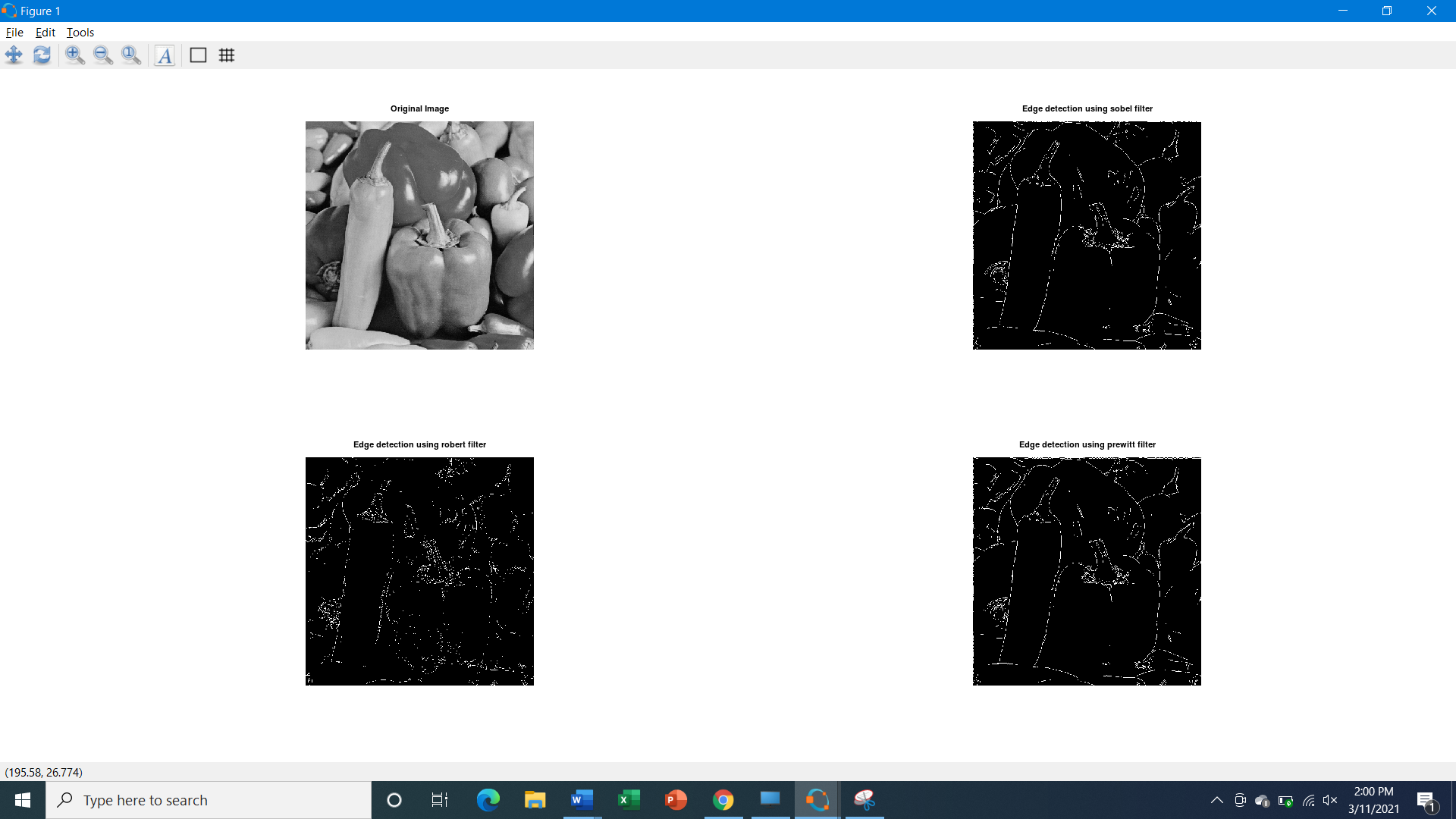
subplot(2,2,4)

imshow(prewitt);

title("Edge detection using prewitt filter");



Output:



**Practical 6: Colour Image Processing**

1. **PseudoColoring**

Code:

#Practical6\_A\_ColorIImage\_Processing\_Pseudocoloring

pkg load image;

clear all;

close all;

img = imread('coins.png');

img = imresize(img,[256 256]);

Output = ones([size(img,1) size(img,2)]);

%COLORMAPS

#maps={'jet(256)';'hsv(256)';'cool(256)';'spring(256)';'summer(256)';'parula(256)';'hot(256)'};

%COLORMAP 1

map = colormap(jet(256));

Red = map(:,1);

Green = map(:,2);

Blue = map(:,3);

R1 = Red(img);

G1 = Green(img);

B1 = Blue(img);

%COLORMAP 2

map = colormap(cool(256));

Red = map(:,1);

Green = map(:,2);

Blue = map(:,3);

%RETRIEVE POSITION OF UPPER TRIANGLE

[x,y]=find(triu(Output)==1);

Output(:,:,1) = Red(img);

Output(:,:,2) = Green(img);

Output(:,:,3) = Blue(img);

for i=1:numel(x)

Output(x(i),y(i),1)=R1(x(i),y(i));

Output(x(i),y(i),2)=G1(x(i),y(i));

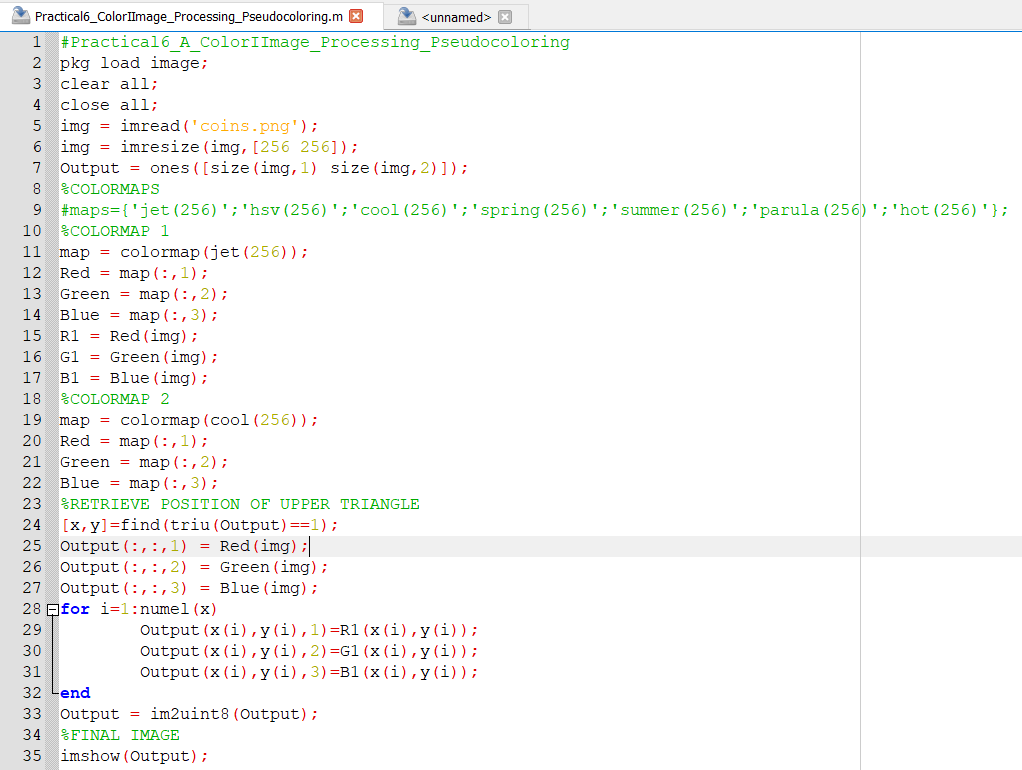
Output(x(i),y(i),3)=B1(x(i),y(i));

end

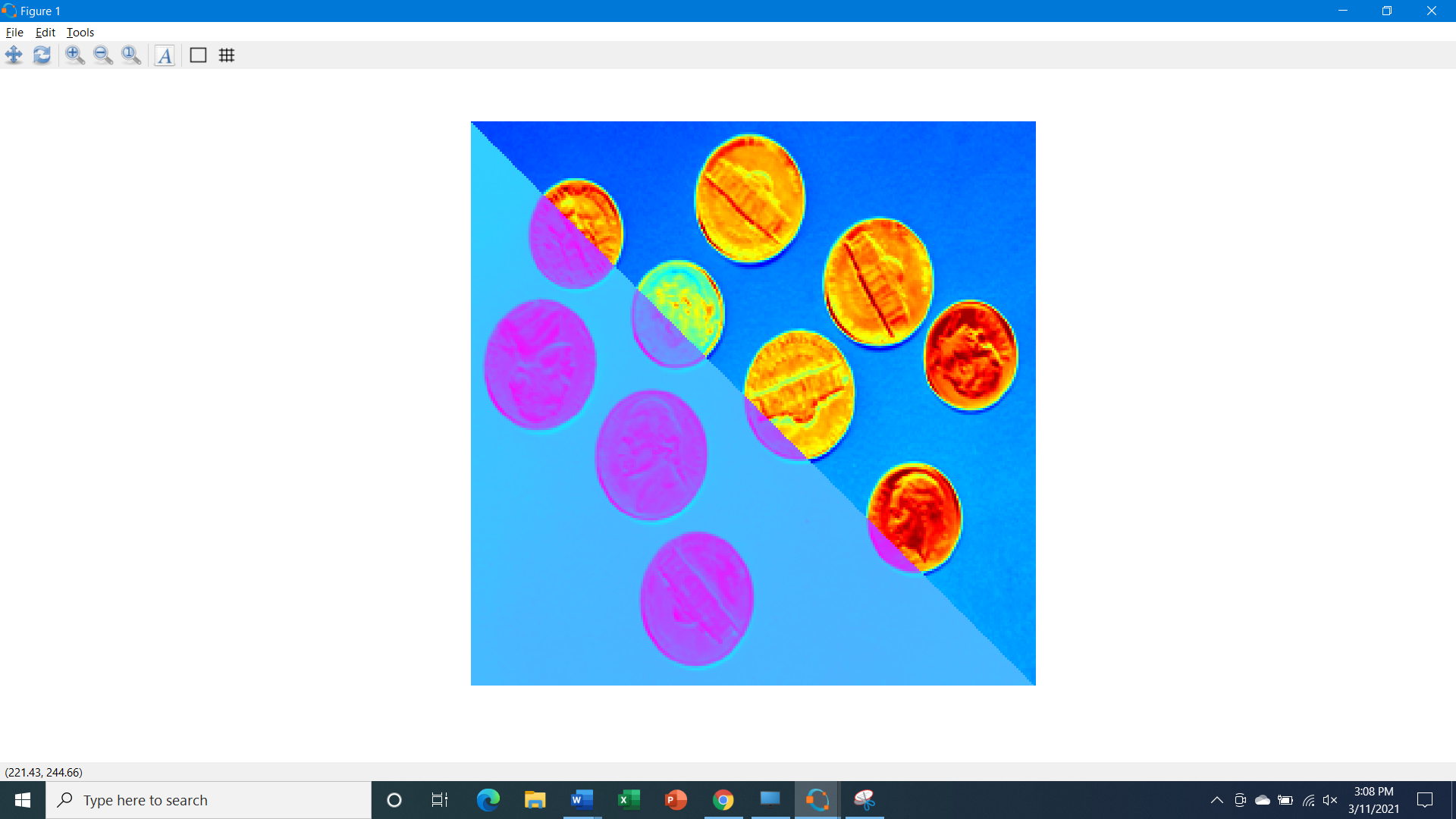
Output = im2uint8(Output);

%FINAL IMAGE

imshow(Output);



Output:



1. **Intensity Slicing**

**Code:**

#Practical6\_B\_ColorIImage\_Processing\_Intensity\_Slicing

pkg load image;

clear all;

close all;

img=imread('hawk.png');

k=rgb2gray(img);

[x y z]=size(k);

% z should be one for the input image

k=double(k);

for i=1:x

for j=1:y

if k(i,j)>=0 && k(i,j)<50

m(i,j,1)=k(i,j,1)+25;

m(i,j,2)=k(i,j)+50;

m(i,j,3)=k(i,j)+60;

end

if k(i,j)>=50 && k(i,j)<100

m(i,j,1)=k(i,j)+55;

m(i,j,2)=k(i,j)+68;

m(i,j,3)=k(i,j)+70;

end

if k(i,j)>=100 && k(i,j)<150

m(i,j,1)=k(i,j)+52;

m(i,j,2)=k(i,j)+30;

m(i,j,3)=k(i,j)+15;

end

if k(i,j)>=150 && k(i,j)<200

m(i,j,1)=k(i,j)+50;

m(i,j,2)=k(i,j)+40;

m(i,j,3)=k(i,j)+25;

end

if k(i,j)>=200 && k(i,j)<=256

m(i,j,1)=k(i,j)+120;

m(i,j,2)=k(i,j)+60;

m(i,j,3)=k(i,j)+45;

end

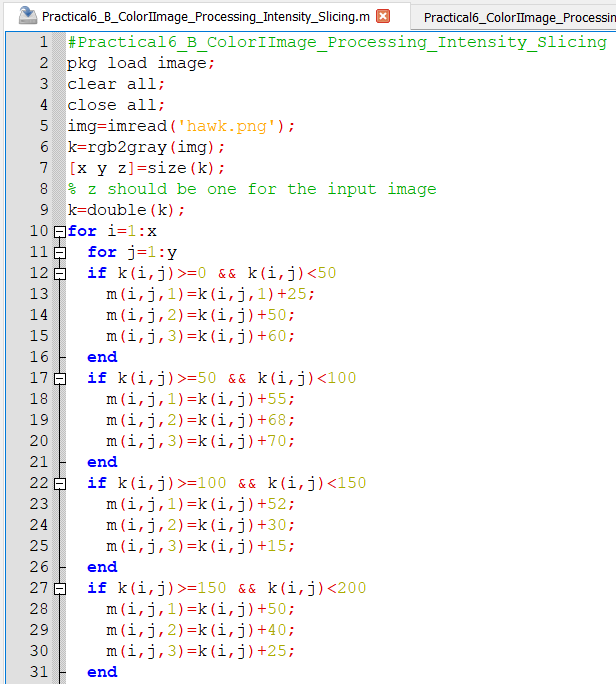
end

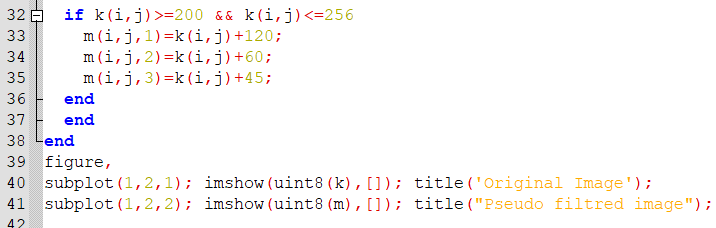
end

figure,

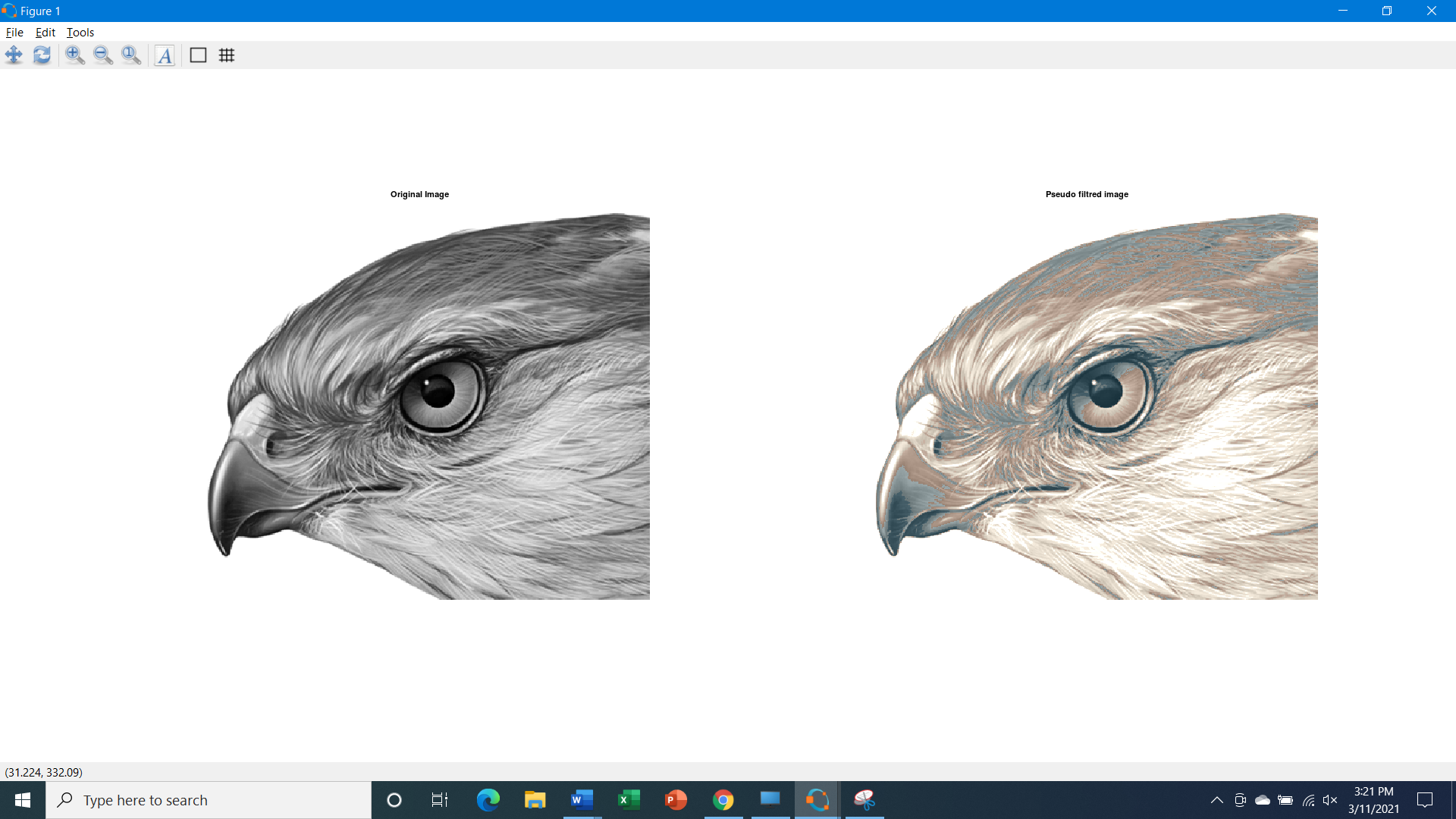
subplot(1,2,1); imshow(uint8(k),[]); title('Original Image');

subplot(1,2,2); imshow(uint8(m),[]); title("Pseudo filtred image");





Output:



1. **Pseudo Image**

**Code:-**

#Practical6\_C\_ColorIImage\_Processing\_Pseudo\_Image

pkg load image;

clear all;

close all;

img = imread('hawk1.png');

red = 0.66\*img;

green=0.25\*img;

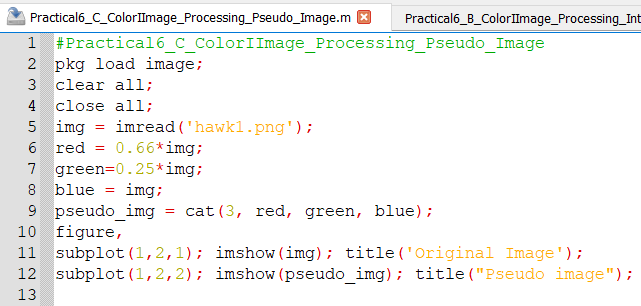
blue = img;

pseudo\_img = cat(3, red, green, blue);

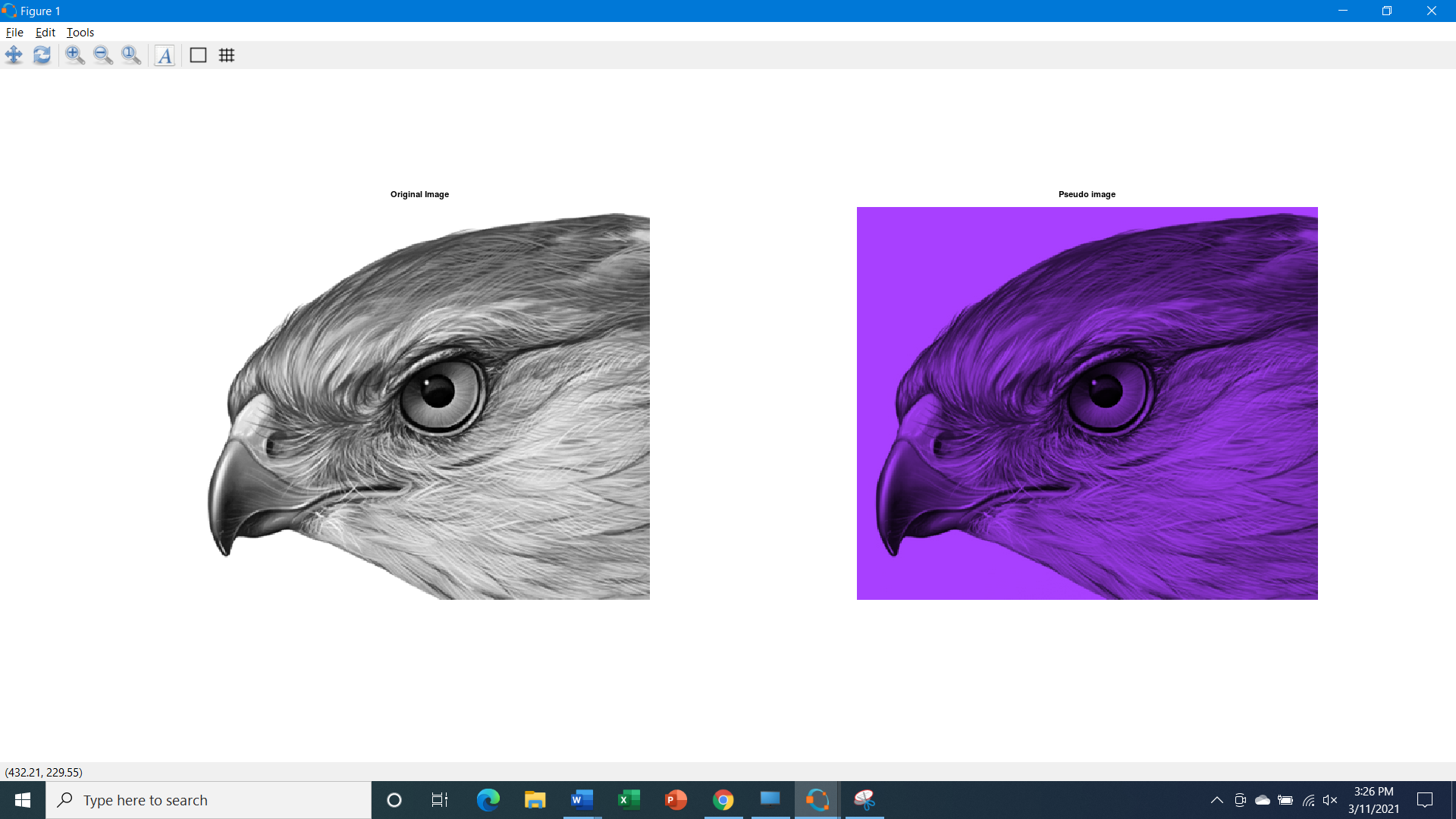
figure,

subplot(1,2,1); imshow(img); title('Original Image');

subplot(1,2,2); imshow(pseudo\_img); title("Pseudo image");



**Output:**



**Practical -7  Image Compression Techniques and watermarking**

1. **Implement Huffman Coding**

Code:

#Practical7\_A\_Image\_compression\_Technique\_Huffman

pkg load image;

pkg load communications;

clear all;

close all;

sig = repmat([3 3 1 3 3 3 3 3 2 3],1,50);

symbols = [1 2 3];

p = [0.1 0.1 0.8];

dict = huffmandict(symbols,p);

hcode = huffmanenco(sig,dict);

dhsig = huffmandeco(hcode,dict);

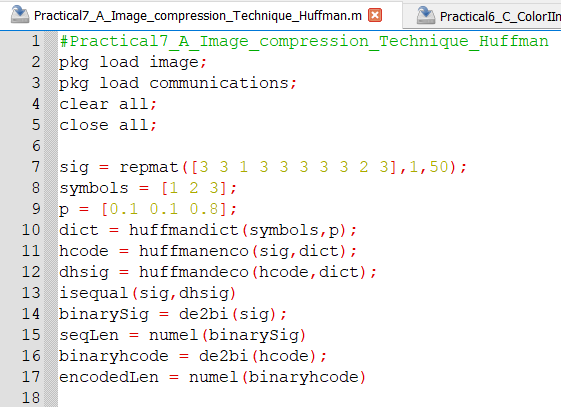
isequal(sig,dhsig)

binarySig = de2bi(sig);

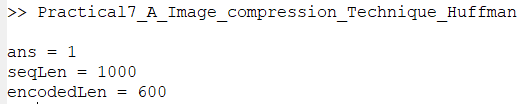
seqLen = numel(binarySig)

binaryhcode = de2bi(hcode);

encodedLen = numel(binaryhcode)



**Output:**



1. **WaterMarking**

Code:

#Practical7\_B\_WaterMarking

pkg load image;

clear all;

close all;

img=imread('lena\_color\_512.tif');

fr=imresize(img,[560 560]);

#Watermarking Image

w=imread('watermark.png');

#Again Resized the Watermarking Image

wr=imresize(w,[560 560]);

#Applied watermarking

alpha=0.7;

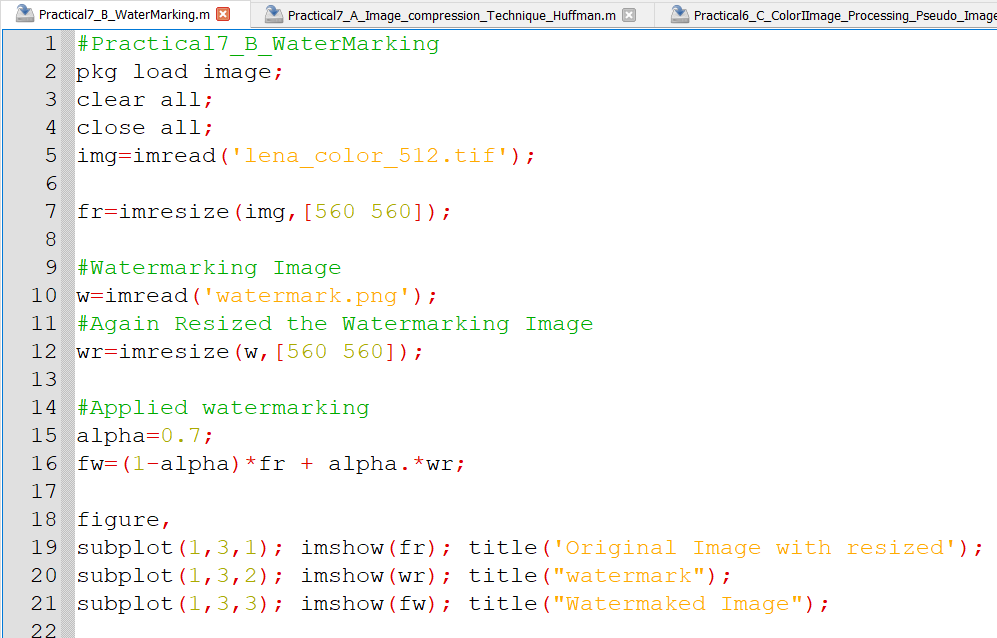
fw=(1-alpha)\*fr + alpha.\*wr;

figure,

subplot(1,3,1); imshow(fr); title('Original Image with resized');

subplot(1,3,2); imshow(wr); title("watermark");

subplot(1,3,3); imshow(fw); title("Watermaked Image");



**Output:**

