**Lab Report 5**

## **Aim**

To apply frequency domain filters for image enhancement in MATLAB.

## **Theory**

##### In image processing, frequency domain filters are used to modify or enhance the frequency content of an image. An image can be represented in the frequency domain using the Fourier transform, which decomposes the image into its constituent frequencies. Frequency domain filters in image processing can be broadly classified into two categories: low-pass filters and high-pass filters. Low-pass filters attenuate high-frequency components in the image, while high-pass filters attenuate low-frequency components. Low-pass filters are commonly used in image smoothing, where they are used to remove high-frequency noise from the image while preserving the lower-frequency content. The most common type of low-pass filter used in image processing is the Gaussian filter, which attenuates high-frequency components according to a Gaussian distribution. High-pass filters, on the other hand, are used to enhance the high-frequency content in an image, making the edges and details of the image more visible. Commonly used high-pass filters in image processing include the Laplacian filter, the Sobel filter, and the Canny filter. Other types of frequency domain filters used in image processing include band-pass filters, which allow a certain range of frequencies to pass through while attenuating other frequencies, and band-stop filters, which attenuate a certain range of frequencies while passing other frequencies. Frequency domain filters in image processing are useful in a wide range of applications, including medical imaging, remote sensing, and industrial inspection, where image quality is critical and noise or interference can have a significant impact on the results.

##### **LOW PASS AND HIGH PASS FILTER**

**Low Pass Filter:**

In image processing, low-pass filters are used to remove high-frequency noise and to smooth images by attenuating high-frequency components while preserving low-frequency components. The most common type of low-pass filter used in image processing is the Gaussian filter. The Gaussian filter is a frequency domain filter that attenuates high-frequency components according to a Gaussian distribution. It is a popular choice for image smoothing because it has a simple mathematical form, is easy to implement, and produces visually pleasing results. The Gaussian filter works by convolving the image with a Gaussian kernel, which is a two-dimensional bell-shaped curve centred at the origin. The size of the kernel and the standard deviation of the Gaussian distribution determine the degree of smoothing and the amount of detail preserved in the image. Another commonly used low-pass filter in image processing is the mean filter, which replaces each pixel in the image with the average of its neighbouring pixels. The size of the neighbourhood or the window used for averaging determines the degree of smoothing, with larger neighbourhoods resulting in greater smoothing and more detail loss. Low-pass filters in image processing can be used to reduce noise, blur, or hide details in an image. They are commonly used in applications such as image denoising, feature extraction, and image segmentation. It is important to note that excessive smoothing can result in loss of important details and edges in the image. Therefore, the choice of the filter and the parameters used for filtering should be carefully tuned to achieve the desired level of smoothing while preserving important features in the image.

**High Pass Filter:**

In image processing, high-pass filters are used to enhance the high-frequency content of an image by attenuating the low-frequency components. High-pass filters are commonly used for edge detection and sharpening of images. The most common type of high-pass filter used in image processing is the Laplacian filter. The Laplacian filter enhances edges and details in an image by detecting areas where the brightness changes rapidly. It does this by convolving the image with the second derivative of the Gaussian function, which is a measure of the rate of change of the image intensity. Another commonly used high-pass filter is the Sobel filter, which is used for edge detection in images. The Sobel filter works by computing the gradient of the image intensity in the x and y directions, and then combining these gradients to obtain a measure of the edge strength. Other types of high-pass filters used in image processing include the Prewitt filter, the Roberts cross filter, and the Canny filter. Each of these filters has its own advantages and disadvantages, and the choice of filter depends on the specific application and the desired outcome. High-pass filters can be useful in a wide range of applications, including medical imaging, industrial inspection, and remote sensing, where image detail and clarity are critical. However, it is important to note that excessive filtering can result in the loss of important image information, such as subtle details and textures. Therefore, the choice of filter and the parameters used for filtering should be carefully tuned to achieve the desired level of enhancement while preserving important image features.

##### **CODE**

% LOW PASS FILTER

fprintf("PLEASE SELECT AN IMAGE\n")

i=uigetfile('\*.\*');

j=imread(i)

k=rgb2gray(j);

imshow(k)

[r,c]=size(k)

l=fft2(k);

m=fftshift(l);

n=abs(m);

o=log(1+n);

imshow(o,[]);

Z=zeros(r,c);

for R=1:r

for C=1:c

if (R-r/2)^2+(C-c/2)^2 <= 50^2

Z(R,C)=1;

end

end

end

imshow(Z)

new=m.\*Z;

new1=fftshift(new);

new2=ifft2(new1);

imshow(abs(new2),[]);

edge=abs(new2);

sharp=double(k)+edge;

imshow(sharp,[]);

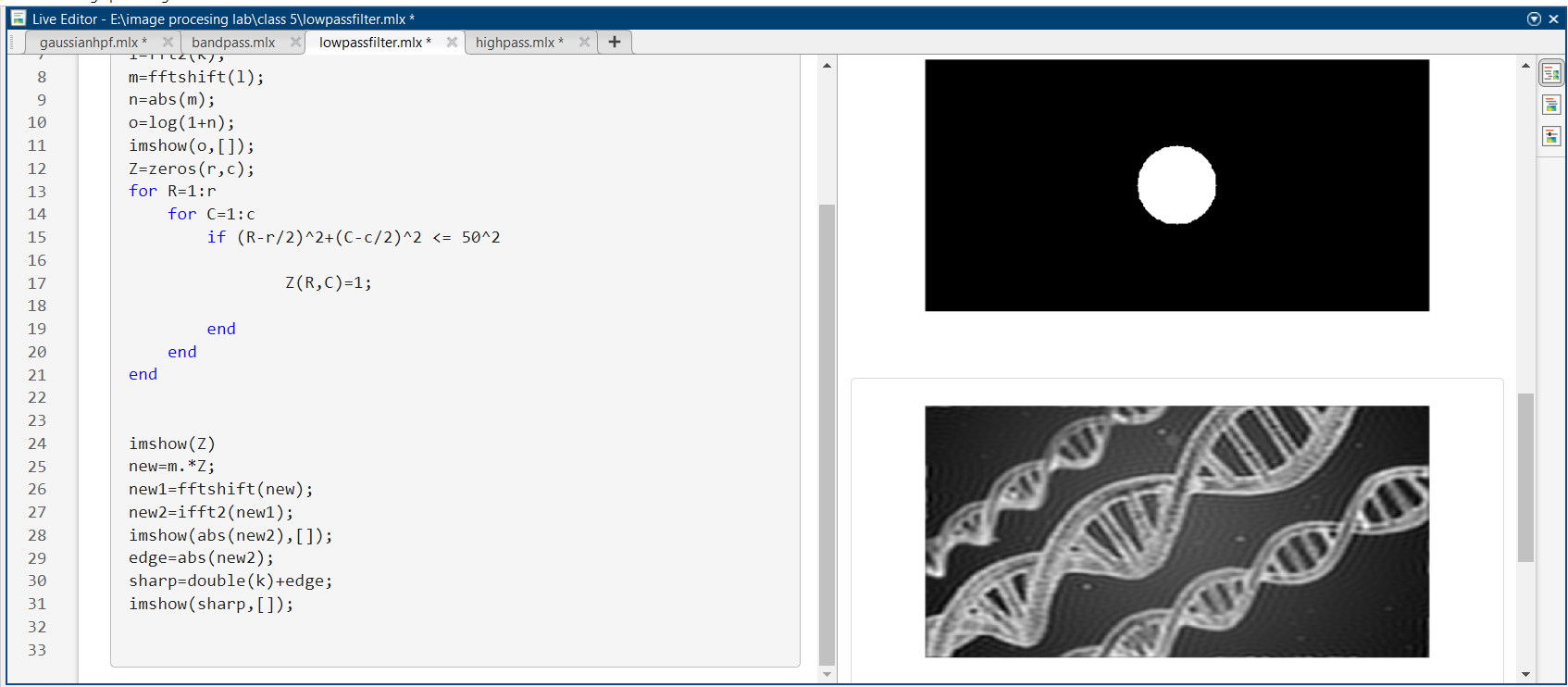
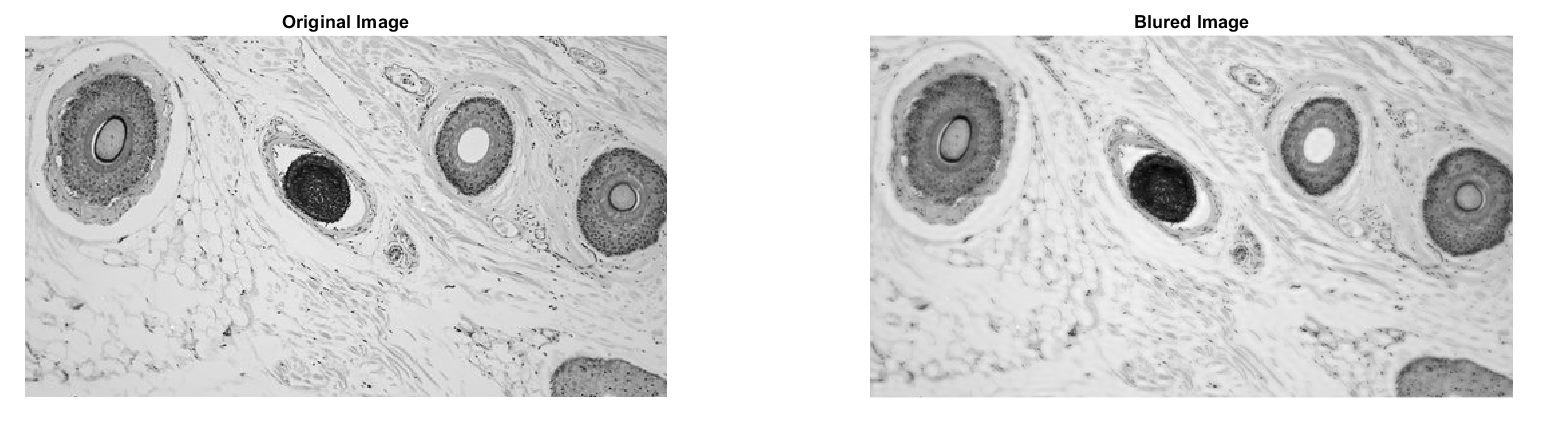


Figure 1: use of low pass filter to smoothen image in MATLAB



*Figure 2: image smoothening using low pass filter in MATLAB*

##### **CODE**

% HIGH PASS FILTER

fprintf("PLEASE SELECT AN IMAGE\n")

i=uigetfile('\*.\*');

j=imread(i)

k=rgb2gray(j);

imshow(k)

[r,c]=size(k)

l=fft2(k);

m=fftshift(l);

n=abs(m);

o=log(1+n);

imshow(o,[]);

Z=zeros(r,c);

for R=1:r

for C=1:c

if (R-r/2)^2+(C-c/2)^2 >= 50^2

Z(R,C)=1;

end

end

end

imshow(Z)

new=m.\*Z;

new1=fftshift(new);

new2=ifft2(new1);

imshow(abs(new2),[]);

edge=abs(new2);

sharp=double(k)+edge;

imshow(sharp,[]);

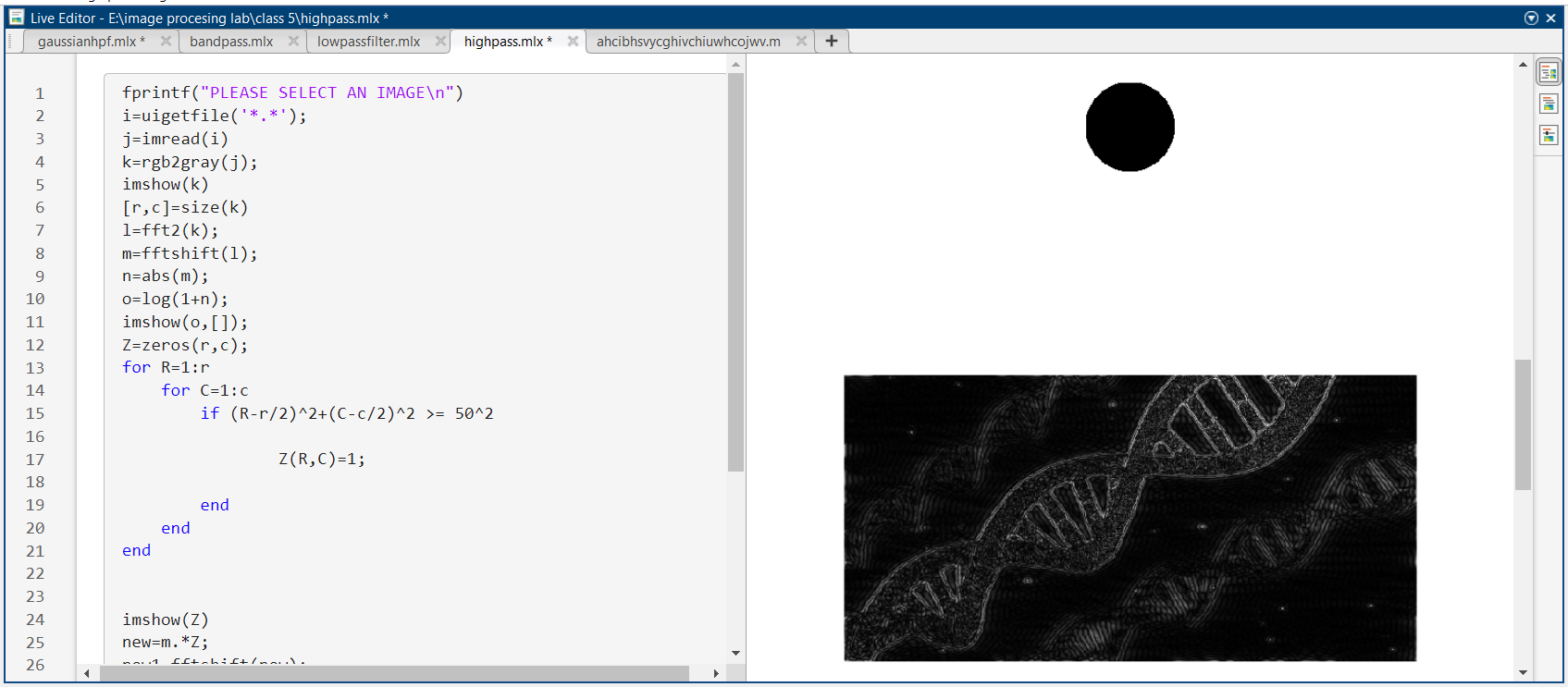


Figure 3: use of high pass filter to detect the edges in MATLAB

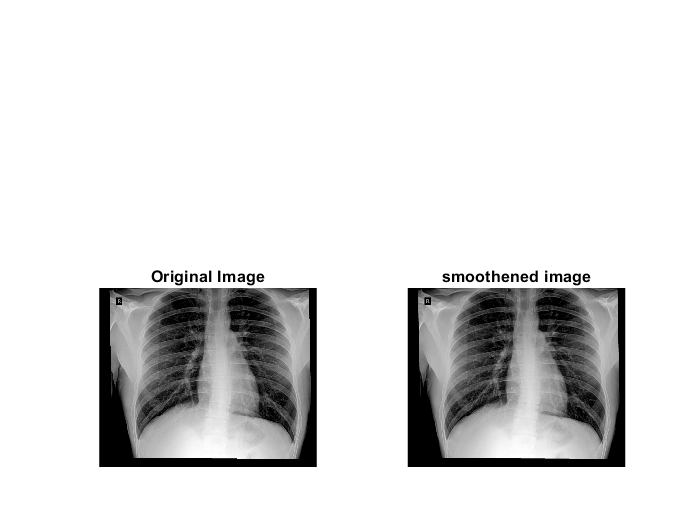


Figure 4: image smoothening using weighted mean filter in MATLAB

##### **POWER LAW TRANFORMATION / GAMMA CORRECTION**

**Gamma Filter:**

The Power law transformation, also known as gamma correction, is a type of image processing technique that alters the intensity values of pixels in an image using a power function. The power function can either increase or decrease the contrast of an image. In this technique, the image's pixel values are raised to a certain exponent (power) that changes the distribution of pixel intensities. The power function can be expressed as:

**s = c \* r^γ**

Where s is the output pixel value, r is the input pixel value, γ is the gamma value, and c is a scaling constant. If the gamma value is less than 1, the resulting image will have increased contrast, making the darker pixels appear lighter and the lighter pixels appear darker. If the gamma value is greater than 1, the resulting image will have decreased contrast, making the darker pixels appear even darker and the lighter pixels even lighter.

Power law transformation is commonly used in image processing to correct for non-linearities in images caused by the imaging system or to enhance image contrast for better visualization.

##### **CODE**

% WEIGHTED MEAN FILTER

fprintf('please Select an image');

y1=uigetfile('\*.\*');

n=input('Please enter the value of gamma for Power law\n');

n=double(n);

J=imread(y1);

y2=rgb2gray(J);

y=double(y2);

%%

y3=y./255;

%%

y4=y3.^n;

%%

y5=y4.\*255;

subplot(1,2,1);imshow(y2);title('Original Image');

subplot(1,2,2);imshow(y5,[]);title('gamma corrected image');

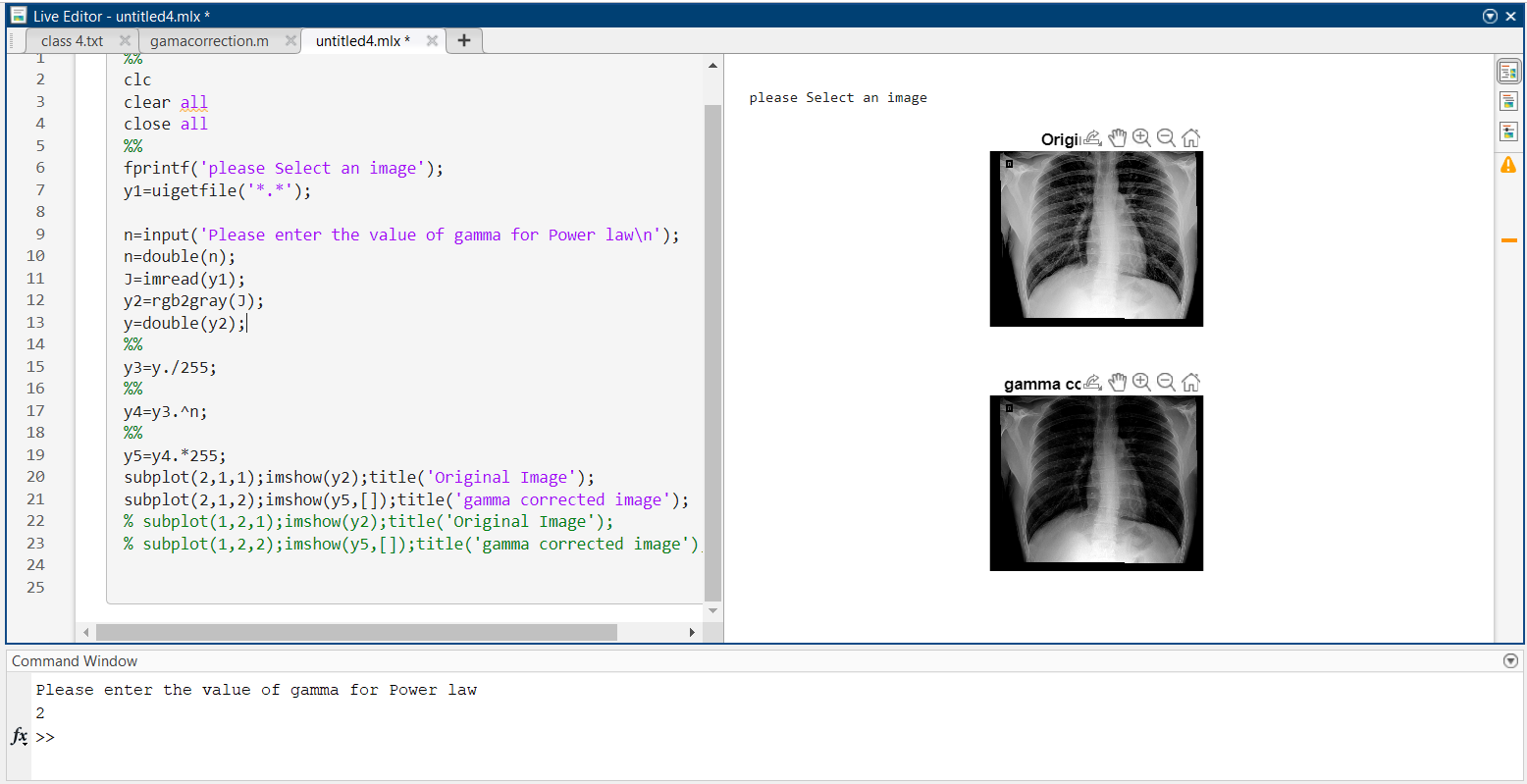


Figure 5: use of gamma filter in MATLAB

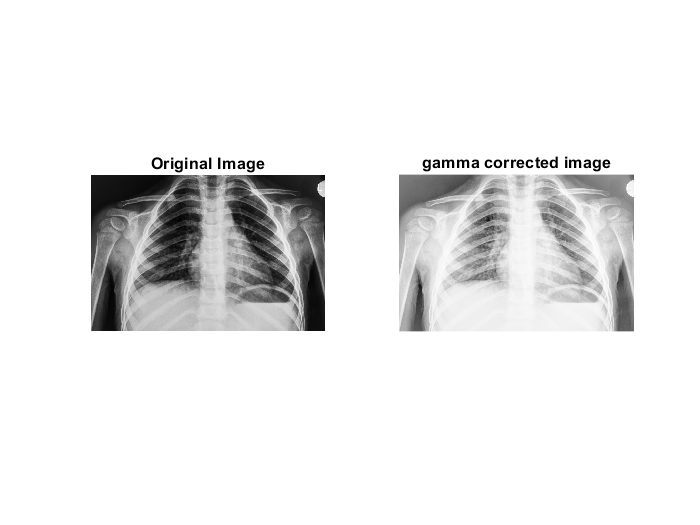


Figure 6: image enhancement using gamma filter in MATLAB

##### **IMAGE NEGATIVE**

**Negative Filter:**

An image negative is an inverted version of an original image. In a negative image, the dark areas of the original image appear light, and the light areas appear dark. This effect is achieved by reversing the brightness and colour values of the original image. New intensity values are given as.

**s = 255-i**

In medical imaging, negative images can be useful for highlighting certain features in an image. For example, a negative image of a CT or MRI scan can be used to enhance the contrast between different tissues or structures, making it easier for a medical professional to identify abnormalities or areas of interest. Negative images can also be used in radiography to highlight the presence of foreign objects, such as metal or glass, that may be difficult to see in a regular X-ray image. In these cases, the negative image can help to differentiate the foreign object from surrounding tissue or bone. Additionally, negative images can be useful for enhancing the visibility of certain types of medical images, such as angiograms or mammograms. By creating a negative image, it is possible to enhance the contrast between blood vessels or breast tissue and background structures, making it easier to identify potential abnormalities.

##### **CODE**

% NEGATIVE FILTER

fprintf('please Select an image');

y=uigetfile('\*.\*');

i=imread(y);

j=rgb2gray(i);

[r,c]=size(j);

newimg=255-i;

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

subplot(1,2,2);imshow(newimg);title('image negative');

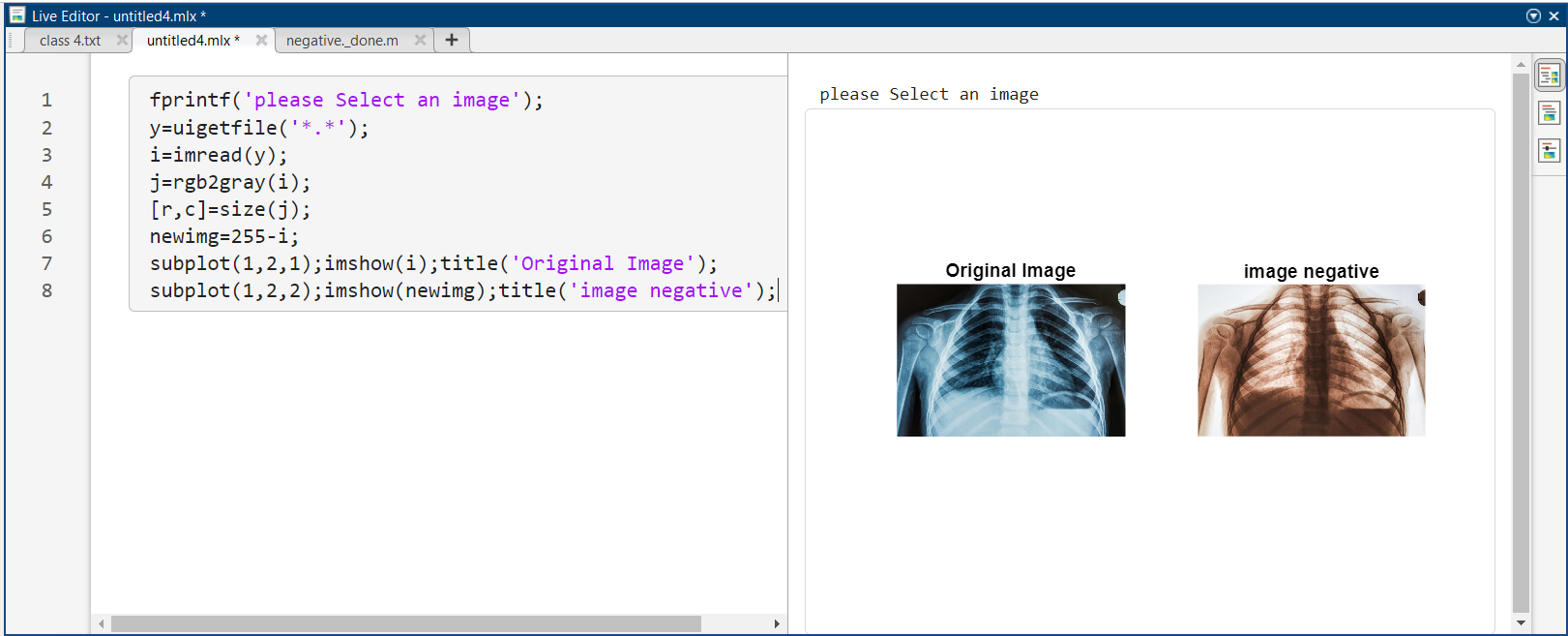


Figure 7: use of image negative filter in MATLAB

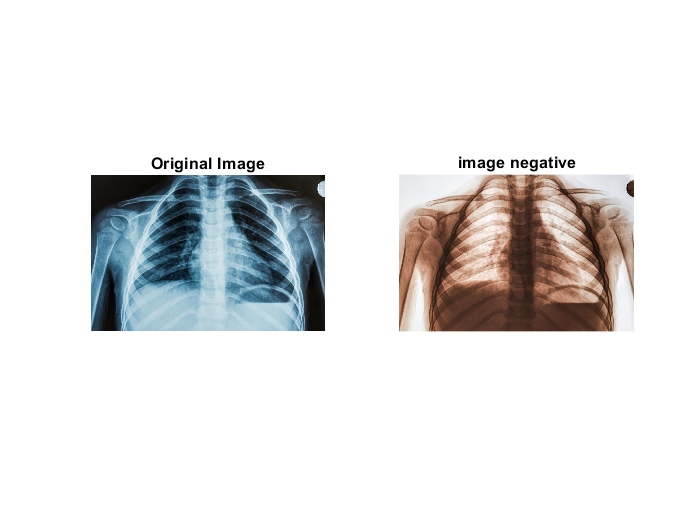


Figure 8: image enhancement using gamma filter in MATLAB

##### **BIT PLANE SLICING**

**Bit Plane Slicing:**

Bit plane slicing is a digital image processing technique used to separate the binary components of an image by extracting each bit plane. In digital imaging, each pixel in an image is represented by a binary code that can be divided into multiple bits. By using bit plane slicing, we can isolate each bit of the binary code, which results in a series of binary images. These binary images represent the contribution of each bit to the original image. For example, in an 8-bit grayscale image, the pixel values range from 0 to 255. We can slice the image into eight-bit planes, where the first bit plane represents the least significant bit (LSB) and the eighth bit plane represents the most significant bit (MSB). The MSB is the most important bit as it has the highest weight and contributes the most to the final pixel value. Each bit plane can be visualized as a binary image, where black pixels represent the bits that have a value of 0, and white pixels represent the bits that have a value of 1. By displaying each bit plane individually or in combination with other bit planes, we can enhance different features of the original image. For instance, the first bit plane would highlight the noise in the image, while the higher bit planes can help us to focus on the edges and details of the image.

Bit plane slicing is used in various applications such as image compression, feature extraction, and image enhancement.

##### **CODE**

% BIT PLANE SLICING

fprintf('please Select an image');

y=uigetfile('\*.\*');

i=imread(y);

% Convert image to grayscale

if size(i, 3) == 3

image = rgb2gray(i);

end

% Extract bit planes

bit\_planes = zeros(size(image, 1), size(image, 2), 8);

for i = 1:8

bit\_planes(:,:,i) = bitget(image, i);

end

% Display bit planes

figure;

for i = 1:8

subplot(2, 4, i);

imshow(bit\_planes(:,:,i), []);

title(['Bit plane ', num2str(i)]);

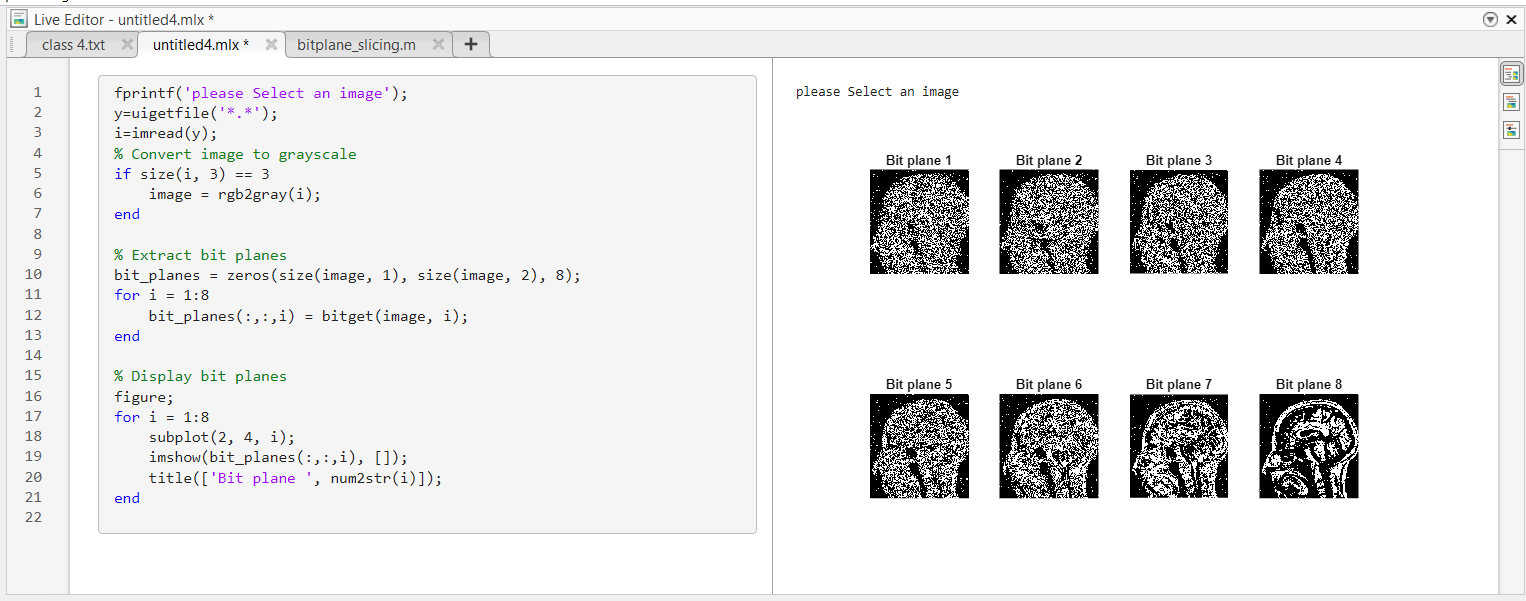


Figure 9: use of image bit slicing in MATLAB

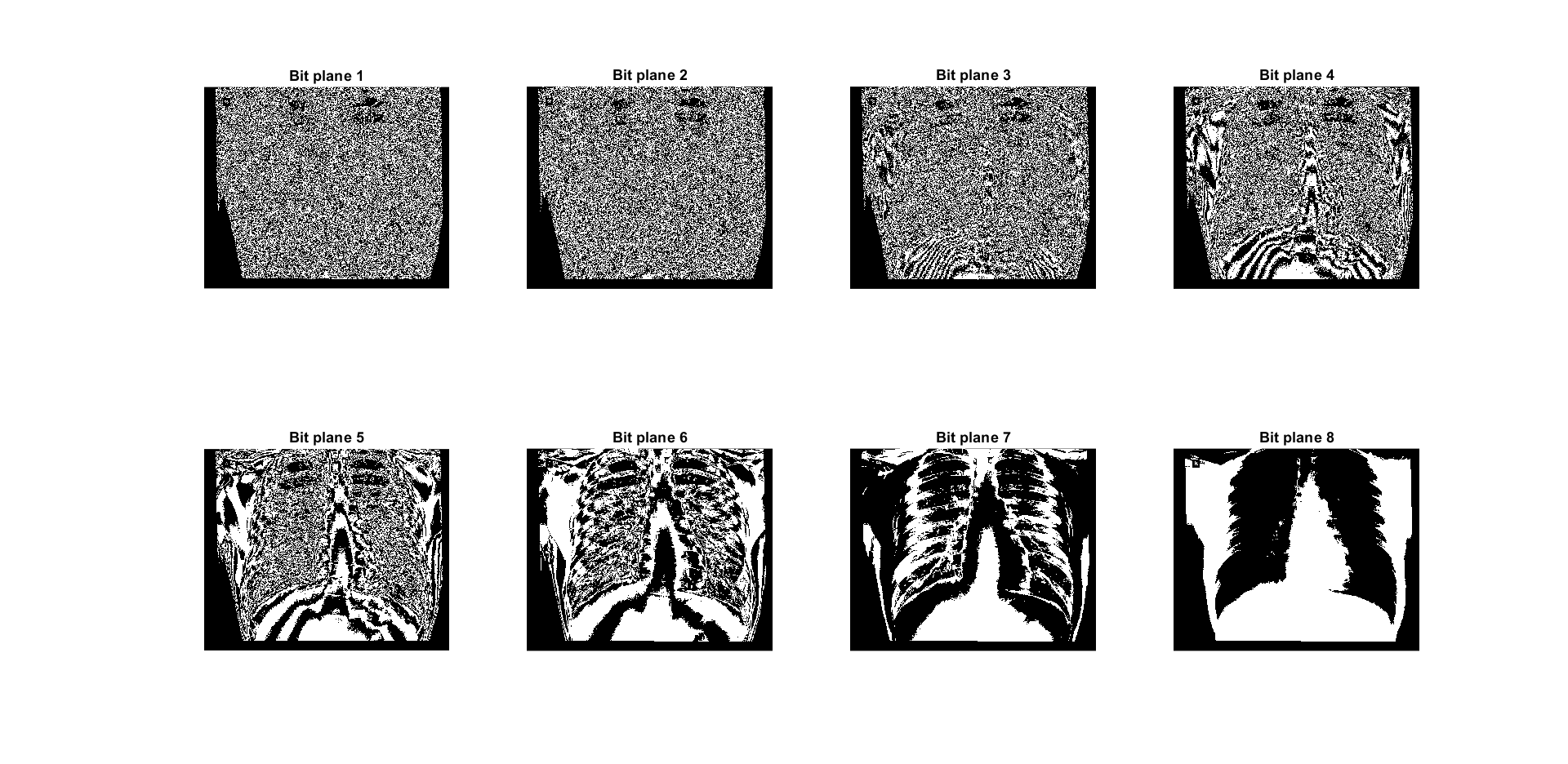


Figure 10: 8 bit sliced image in MATLAB

##### **CONTRAST STRETCHING**

**Contrast stretching:**

Contrast stretching is a technique used in image processing to enhance the contrast of an image. The technique works by expanding the dynamic range of the image so that the brightest and darkest pixels are spread over the entire range of pixel values, thus increasing the contrast. The process involves finding the minimum and maximum pixel values in the image, and then mapping the values in between to a new range. This new range is typically the full range of values that the image can represent, such as 0 to 255 for an 8-bit grayscale image. There are several methods for implementing contrast stretching, including linear stretching, piecewise linear stretching, and histogram equalization. Linear stretching involves simply scaling the pixel values between the minimum and maximum values to fill the entire range. Piecewise linear stretching involves dividing the range into several segments and applying a linear function to each segment. Histogram equalization involves mapping the histogram of the image to a uniform distribution. Contrast stretching is a useful technique for enhancing the visibility of details in an image, particularly when the contrast is low. However, it can also lead to artifacts and noise amplification if not applied carefully. Therefore, it is important to choose an appropriate method and parameter settings for each specific image.

##### **CODE**

% CONTRAST STRETCHING

im=imread("Xray\_share.jpg");

img=rgb2gray(im);

I=double(img);

x=[0 40 150 250];

y=[0 90 170 200];

plot(x,y);

[rows,columns]=size(I);

for r=1:rows

for c=1:columns

if I(r,c )<x(2)

I(r,c )=(y(2)/x(2))\*r;

elseif I(r,c)>=x(2) && I(r,c)<x(3)

I(r,c)=((y(3)-y(2))/(x(3)-x(2)))\*(r-x(2) );

else

I(r,c)=((y(4)-y(3))/(x(4)-x(3)))\*(r-x(3) );

end

end

end

figure,

imshow(I);

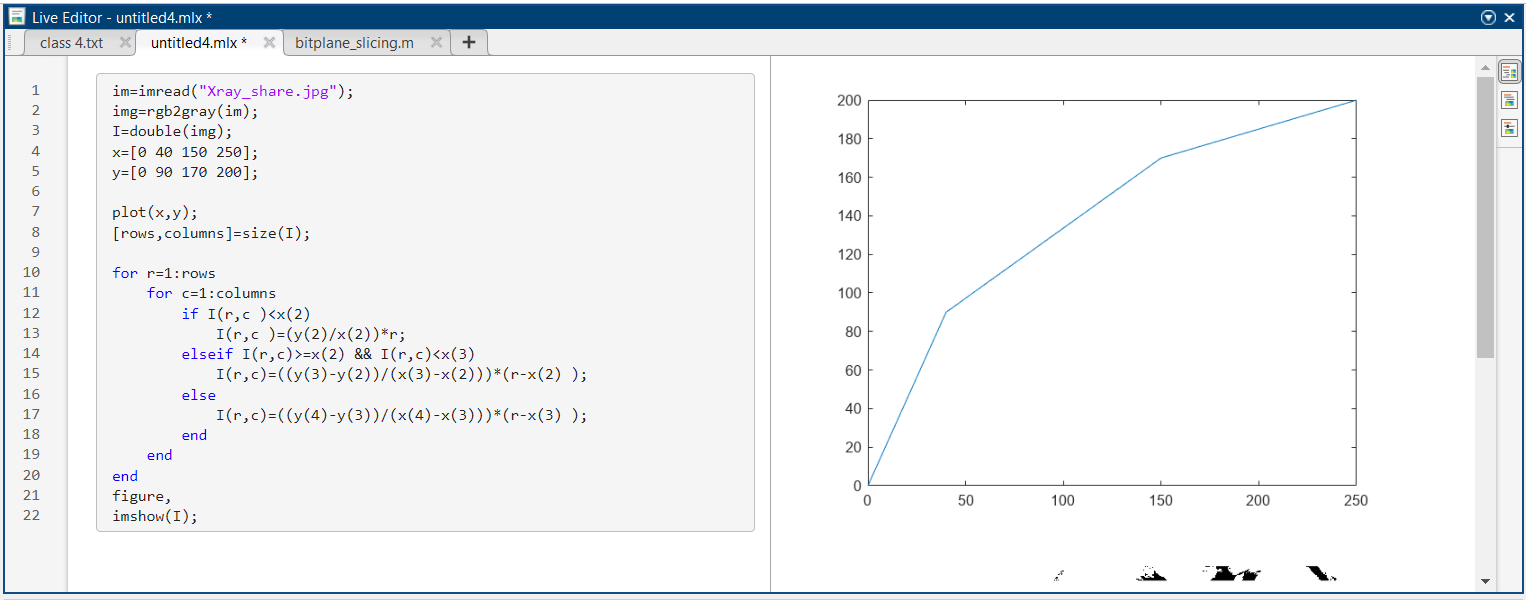


Figure 11: use of contrast stretching in MATLAB

##### **MIN AND MAX FILTER**

**Min and Max filter:**

In image processing, a min filter (also known as erosion) is an operation that replaces each pixel in an image with the minimum value of its neighbouring pixels, using a kernel or structuring element. The result is an image that has had small features or details removed, effectively shrinking the image. On the other hand, a max filter (also known as dilation) replaces each pixel in an image with the maximum value of its neighbouring pixels. This operation has the opposite effect of the min filter, and can be used to enhance or highlight the edges and boundaries in an image.

Both min and max filters are used in image processing for various purposes, such as noise reduction, feature extraction, and object detection. They can be applied to grayscale as well as colour images. The size and shape of the kernel or structuring element used for the filtering process determines the degree of smoothing or sharpening in the resulting image..

##### **CODE**

% MIN FILTER

fprintf('please Select an image');

y=uigetfile('\*.\*');

i=imread(y);

k=rgb2gray(i);

j=imnoise(k,'salt & pepper',0.1);

d = padarray(j,[1 1],0,'both');

[r,c]=size(d);

for R =2:(r-1)

for C=2:(c-1)

if d(R,C)>=250

v=[d(R,C) d(R,C+1) d(R,C-1) d(R+1,C) d(R+1,C-1) d(R+1,C+1) d(R-1,C) d(R-1,C-1) d(R-1,C+1)];

d(R,C)=min(v);

end

end

end

e=d(2:end-1,2:end-1); %removing padding

subplot(1,3,1);imshow(i);title('Original Image');

subplot(1,3,2);imshow(j);title('salt and pepper noised');

subplot(1,3,3);imshow(e);title('salt removed');

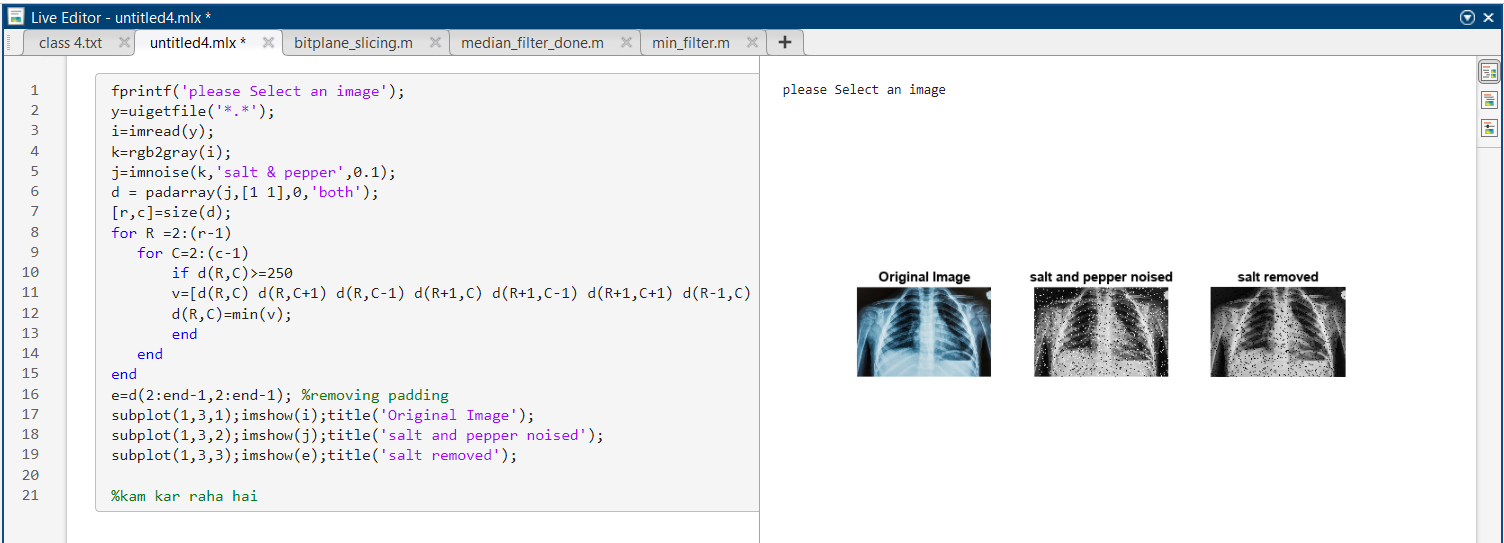


Figure 12: use of min() function for making min filter in MATLAB

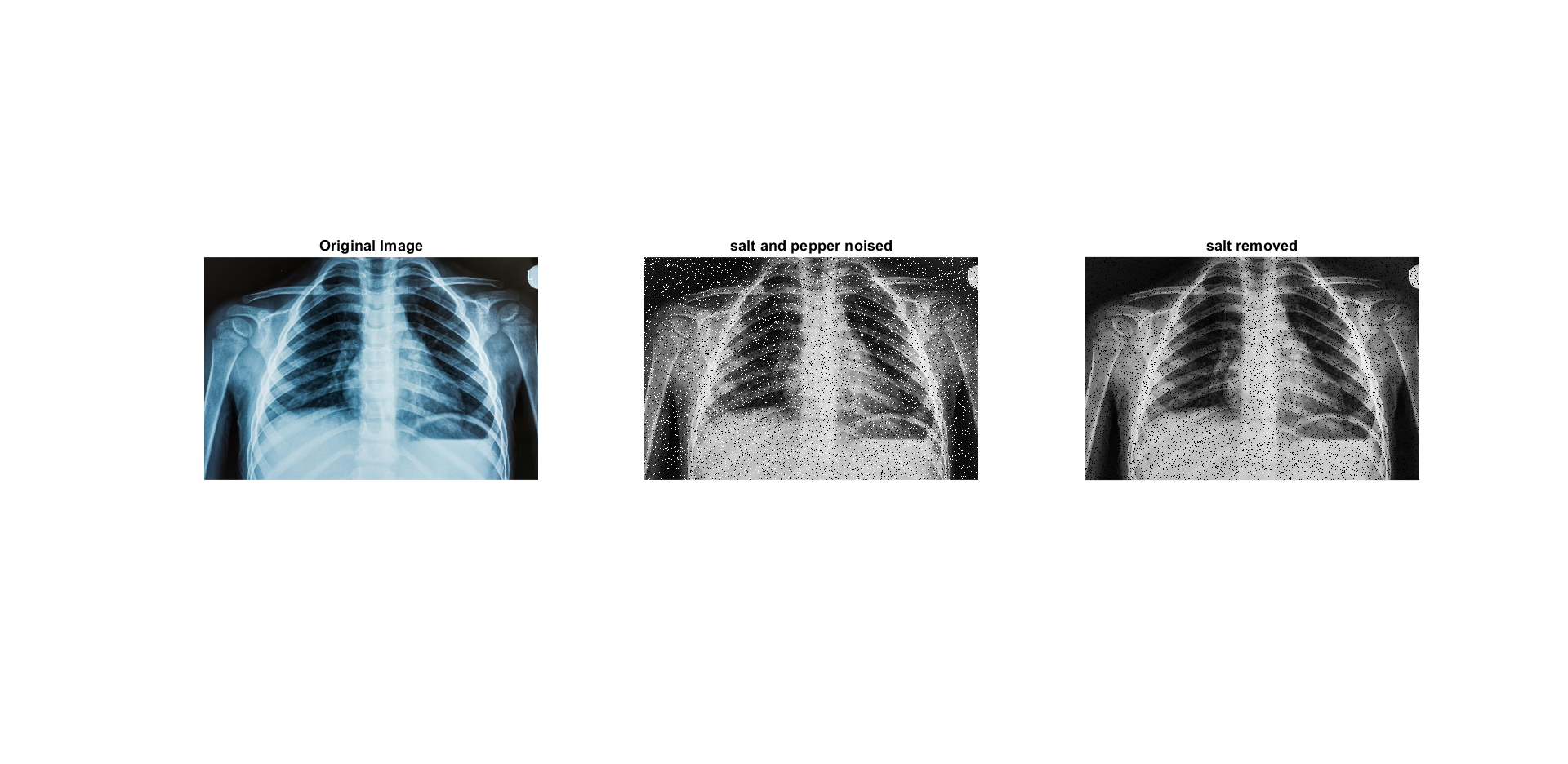


Figure 13: min filter in MATLAB is able to filter only salt noise.

##### **CODE**

% MAX FILTER

fprintf('please Select an image');

y=uigetfile('\*.\*');

i=imread(y);

k=rgb2gray(i);

j=imnoise(k,'salt & pepper',0.1);

d = padarray(j,[1 1],0,'both');

[r,c]=size(d);

for R =2:(r-1)

for C=2:(c-1)

if d(R,C)<=10

v=[d(R,C) d(R,C+1) d(R,C-1) d(R+1,C) d(R+1,C-1) d(R+1,C+1) d(R-1,C) d(R-1,C-1) d(R-1,C+1)];

d(R,C)=max(v);

end

end

end

e=d(2:end-1,2:end-1); %removing padding

subplot(1,3,1);imshow(i);title('Original Image');

subplot(1,3,2);imshow(j);title('salt and pepper noised');

subplot(1,3,3);imshow(e);title('pepper removed');

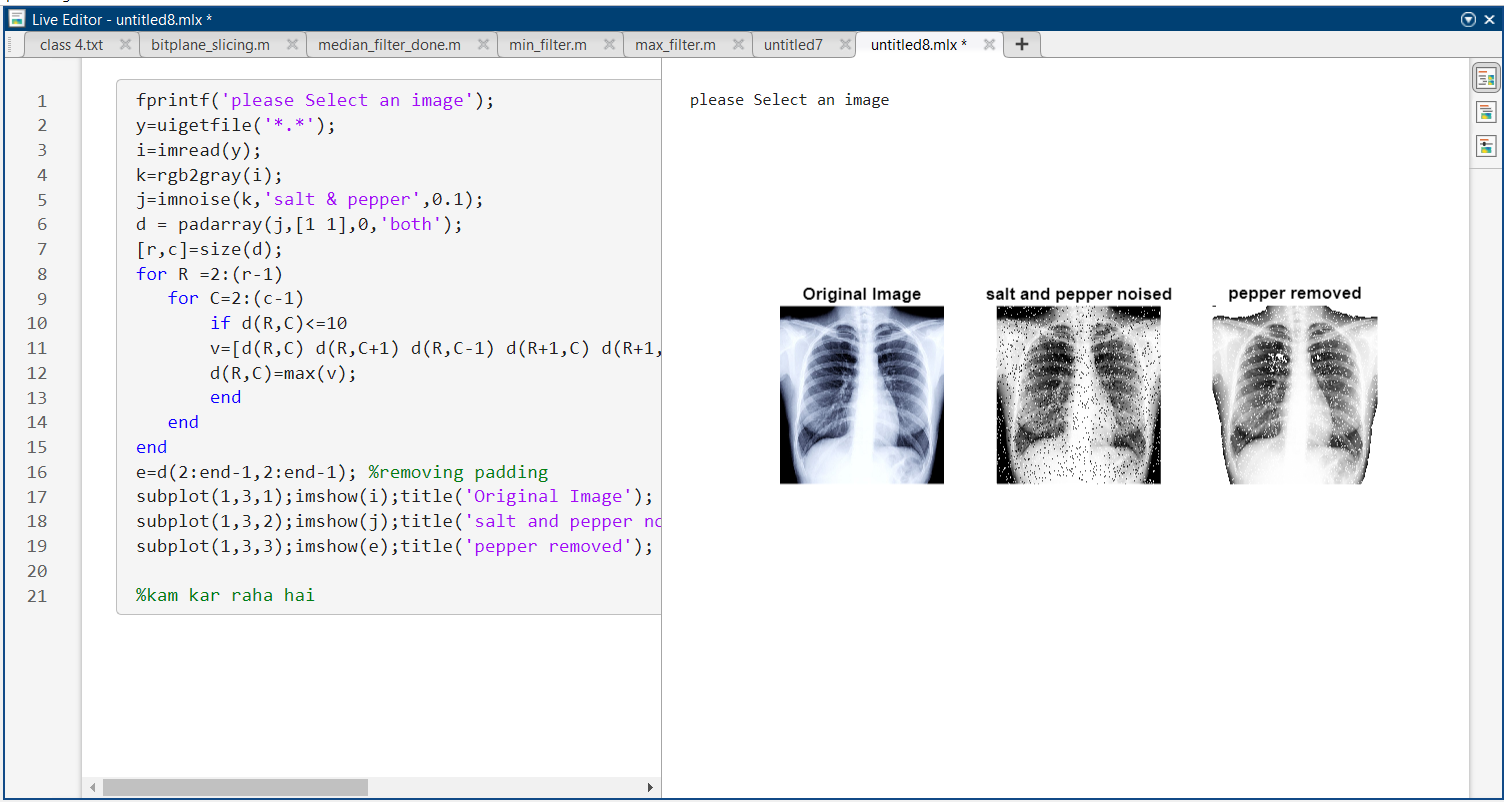


Figure 14: use of max() function for making max filter in MATLAB

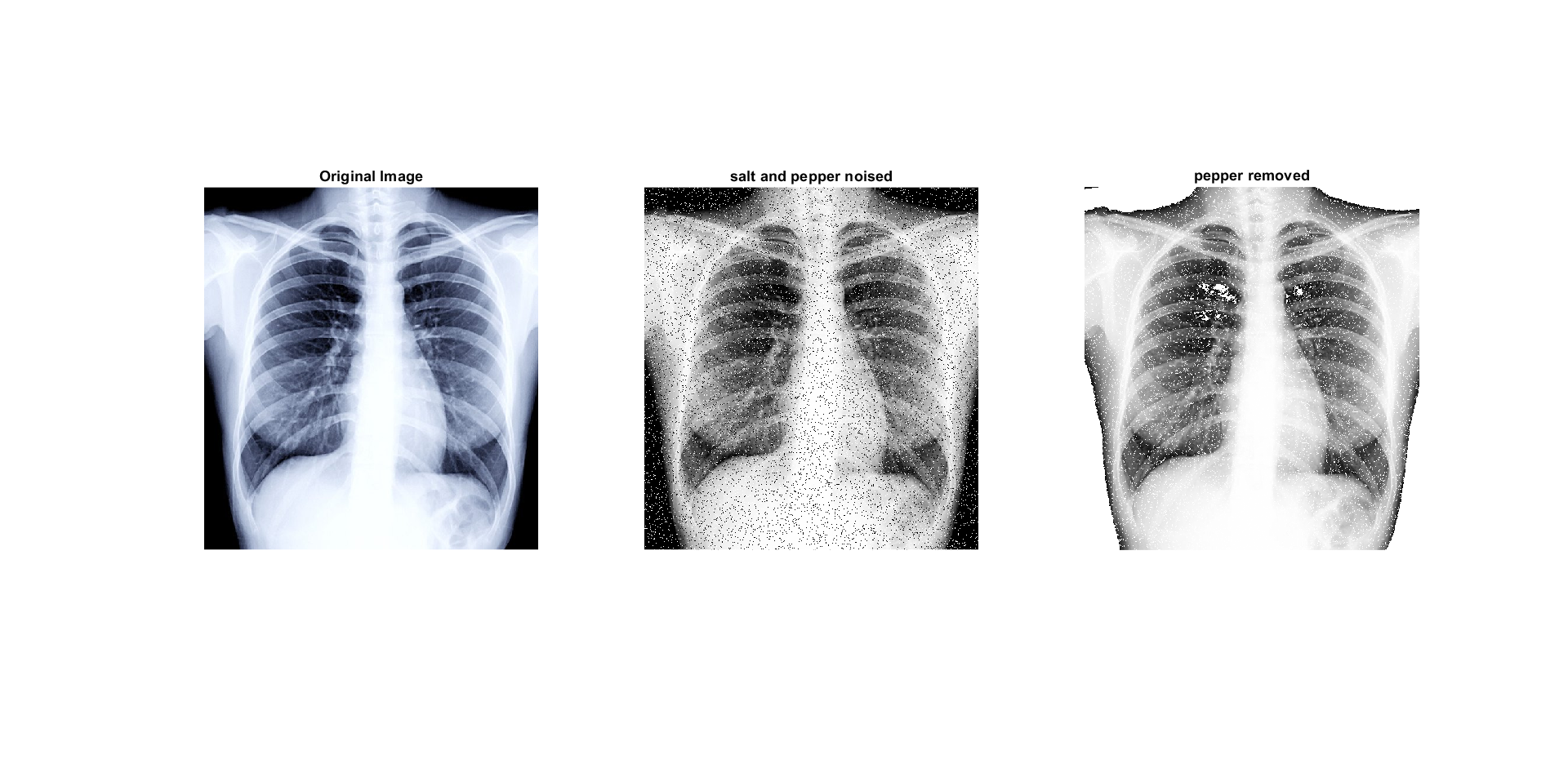


Figure 15: max filter in MATLAB is able to filter only pepper noise.

##### **MEDIAN FILTER**

**Min and Max filter:**

A median filter is a digital signal processing technique used to remove noise from a signal or an image. It works by replacing each pixel's value with the median value of its neighbouring pixels. The process of median filtering involves sliding a window (typically a square or rectangular shape) over the input signal or image. For each pixel within the window, the median value of the pixel's neighbourhood is calculated and used as the new value for that pixel. The size of the window determines the extent of smoothing applied to the signal or image.

Median filtering is commonly used in image processing to remove salt and pepper noise, which appears as random black and white pixels in an image. The median filter can effectively remove this type of noise without blurring or distorting the image's edges and details, making it a popular choice for image denoising.

##### **CODE**

% MEDIAN FILTER

fprintf('please Select an image');

y=uigetfile('\*.\*');

i=imread(y);

k=rgb2gray(i);

j=imnoise(k,'salt & pepper',0.1);

d = padarray(j,[1 1],0,'both');

[r,c]=size(d);

for R =2:(r-1)

for C=2:(c-1)

%if d(R,C)==255 || d(R,C)==0

v=[d(R,C) d(R,C+1) d(R,C-1) d(R+1,C) d(R+1,C-1) d(R+1,C+1) d(R-1,C) d(R-1,C-1) d(R-1,C+1)];

d(R,C)=median(v);

%end

end

end

e=d(2:end-1,2:end-1); %removing padding

subplot(1,3,1);imshow(i);title('Original Image');

subplot(1,3,2);imshow(j);title('salt and pepper noised');

subplot(1,3,3);imshow(e);title('noised removed');

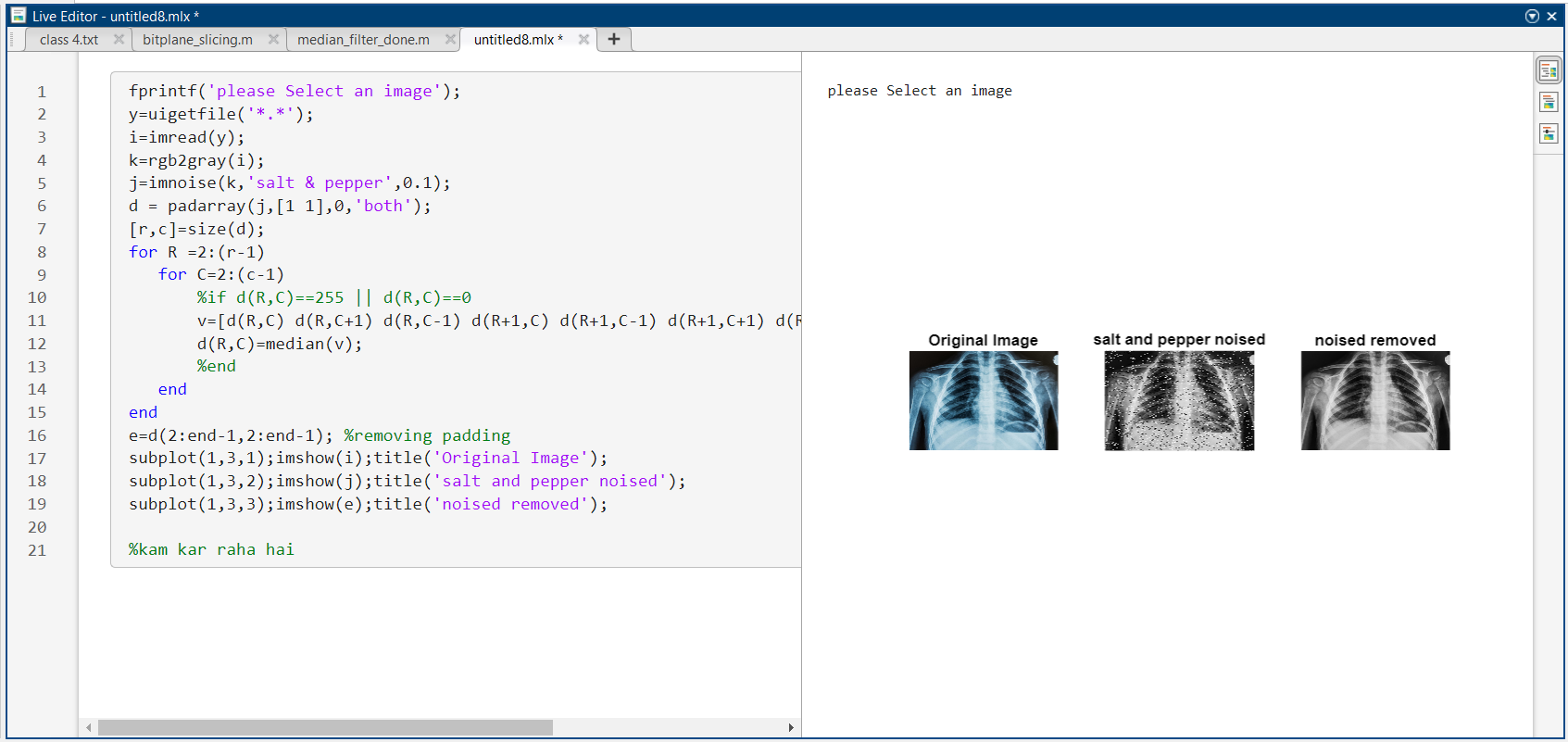


Figure 16: use of median() function for making median filter in MATLAB

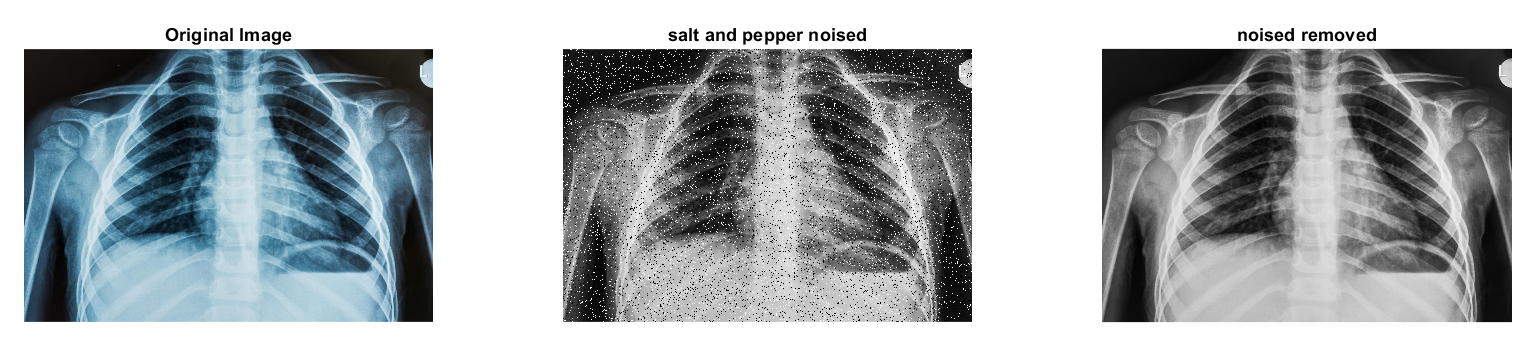


Figure 17: median filter in MATLAB is able to filter both salt and pepper noise.

---------------------------------------------

Shreenandan Sahu |120BM0806