Lab Report 1

Aim

To go through the basics of image processing tools like reading, displaying, converting colour and doing some arithmetic operations on the medical images.

Theory

Reading and Displaying an Image

- Clear the MATLAB workspace of any variables and clear the command window using the commands `clear; close all;` and `clc;`
- Use `imread()` to read image files into a matrix in MATLAB. Once you imread an image, it is stored as an ND-array in memory.
- Use `size()` to see the property of image matrix like dimensions and colour bit in MATLAB.
- Use `imshow()` to show the image.

CODE

```
img=imread('read.png');
[x,y]=size(img);
imshow(img);
```

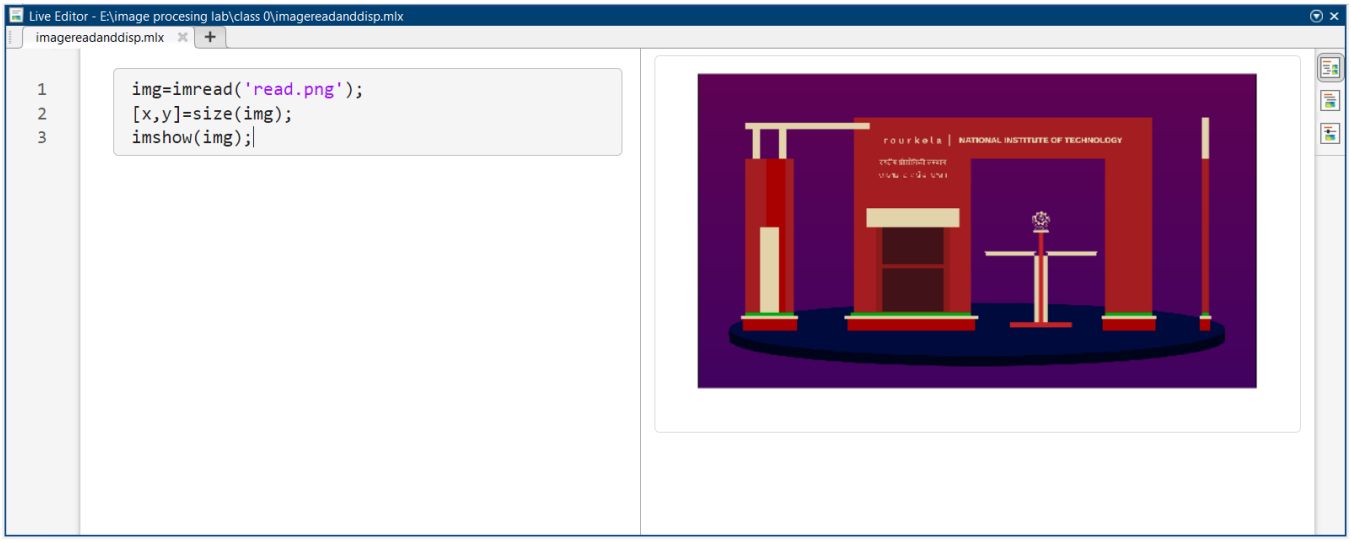


Figure 1: use of imread( ) and imshow( ) function in MATLAB

Image Arithmetic

- As images are represented in a matrix format to perform image arithmetic the size of images should be same. Operation on two images leads to a new image
- Use `imadd()` to add two image files. The corresponding value of the matrix of two images are added and in return we get a new image. We can also use a constant value instead of image, this will add the

image pixels with a constant. The two images should be of same dimension to do the addition as it involves the matrix addition.

- Use *imsubtract()* to subtract two image files.
- Use *immultiply()* to multiply two files. The corresponding value of the matrix of two images are multiplied and the resultant image is a new image. The two images should be of same dimension to do the multiply as it involves the matrix. We can also use a constant value instead of image, this will multiply the image pixels with a constant. If the pixel values are fractional then it will round it off to nearest value
- Use *imdivide()* to divide two or more image files.

CODE

```
img1=imread('mypic1.png')
img2=imread('mypic2.png')
j=imadd(img1,img2)
subplot(1,3,1);imshow(img1);title("image 1")
subplot(1,3,2);imshow(img2);title("image 2")
subplot(1,3,3);imshow(j);title("added images")
```

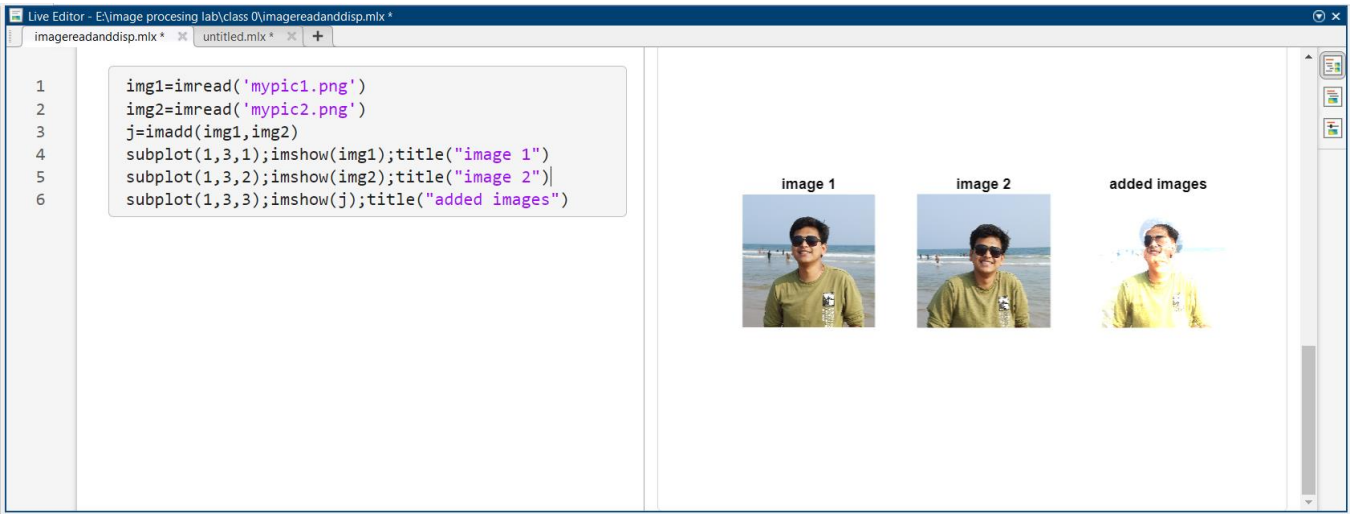


Figure 2: use of imadd( ) to add two images in in MATLAB

CODE

```
img1=imread('mypic1.png')
img2=imread('mypic3.png')
j=imsubtract(img1,img2)
subplot(1,3,1);imshow(img1);title("image 1")
subplot(1,3,2);imshow(img2);title("image 2")
subplot(1,3,3);imshow(j);title("subtracted images")
```

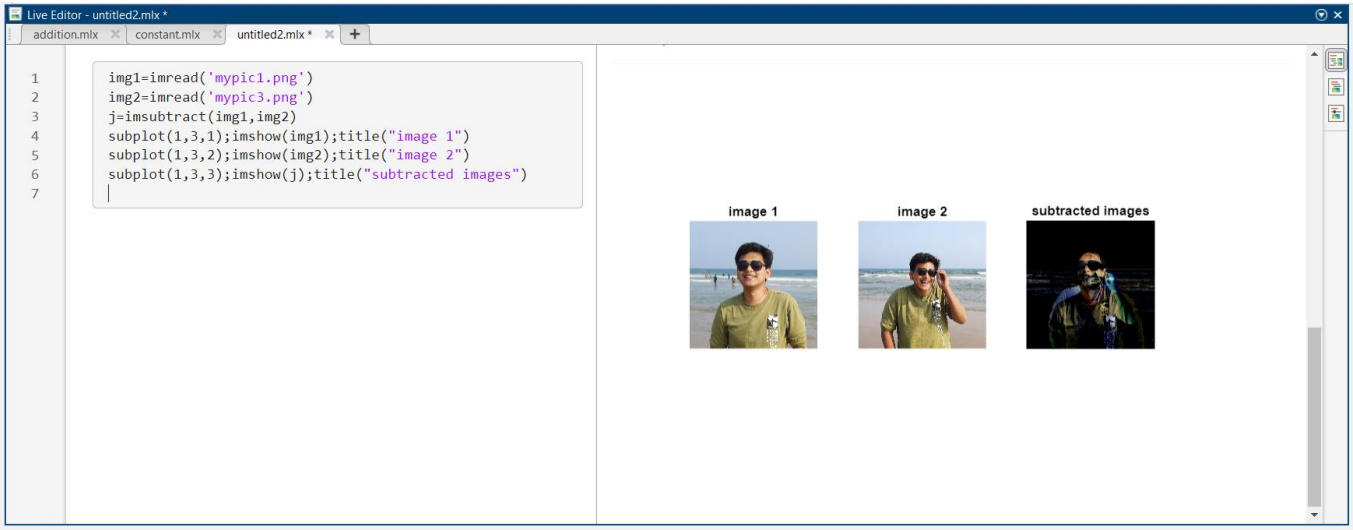


Figure 3: use of imsubtract( ) to subtract two images in in MATLAB

CODE

```
img1=imread('mypic1.png')
i=imadd(img1,-100);
j=imadd(img1,-50);
k=imadd(img1,50);
l=imadd(img1,100);
subplot(3,2,1);imshow(img1);title('Original Image');
subplot(3,2,3);imshow(j);title('constant value=-50');
subplot(3,2,4);imshow(i);title('constant value=-100');
subplot(3,2,5);imshow(k);title('constant value= 50');
subplot(3,2,6);imshow(l);title('constant value= 100');
```

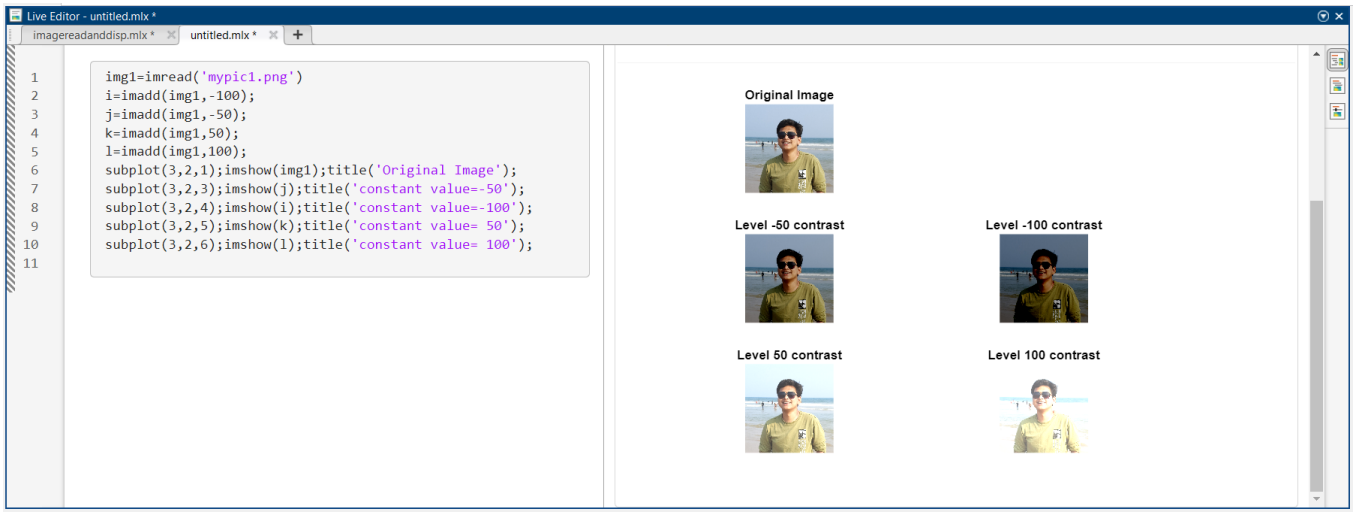


Figure 4: use of imadd( ) to add constant to image in MATLAB

CODE

```
img1=imread('mypic1.png')
img2=imread('mypic2.png')
j=immultiply(img1,img2)
subplot(1,3,1);imshow(img1);title("image 1")
subplot(1,3,2);imshow(img2);title("image 2")
subplot(1,3,3);imshow(j);title("multipled images")
```

Figure 5: use of immultiply( ) to multiply two images in in MATLAB

CODE

```
img1=imread('mypic1.png')
img2=imread('mypic2.png')
j=imdivide(img1,img2)
subplot(1,3,1);imshow(img1);title("image 1")
subplot(1,3,2);imshow(img2);title("image 2")
subplot(1,3,3);imshow(j);title("divided images")
```

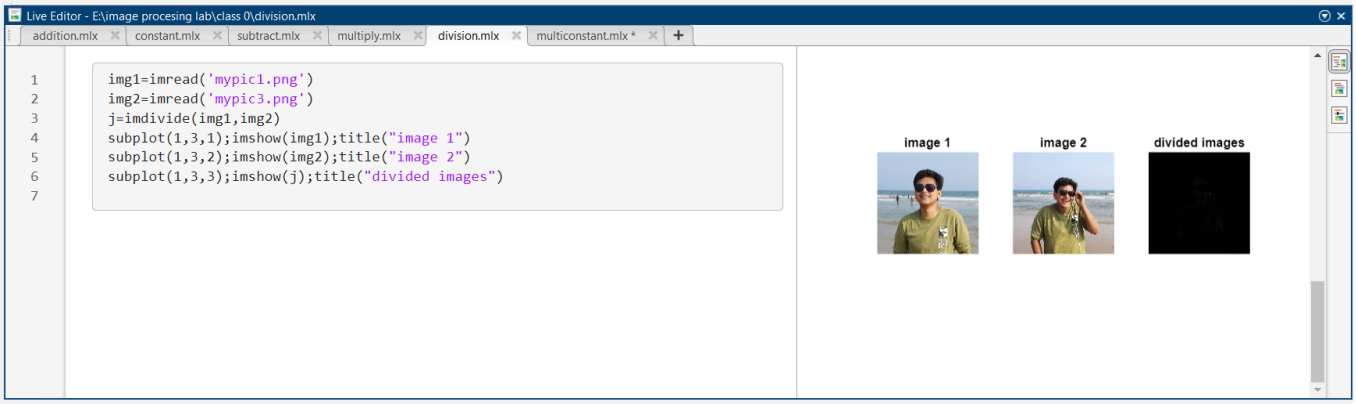


Figure 6: use of imdivide( ) to divide two images in in MATLAB

CODE

```
img1=imread('mypic1.png')
i=immultiply(img1,0.5);
j=immultiply(img1,0.25);
k=immultiply(img1,2.5);
l=immultiply(img1,5);
```

```
subplot(3,2,1);imshow(img1);title('Original Image');
subplot(3,2,3);imshow(j);title('constant value= 0.5');
subplot(3,2,4);imshow(i);title('constant value= 0.25');
subplot(3,2,5);imshow(k);title('constant value= 2.5');
subplot(3,2,6);imshow(l);title('constant value= 5');
```

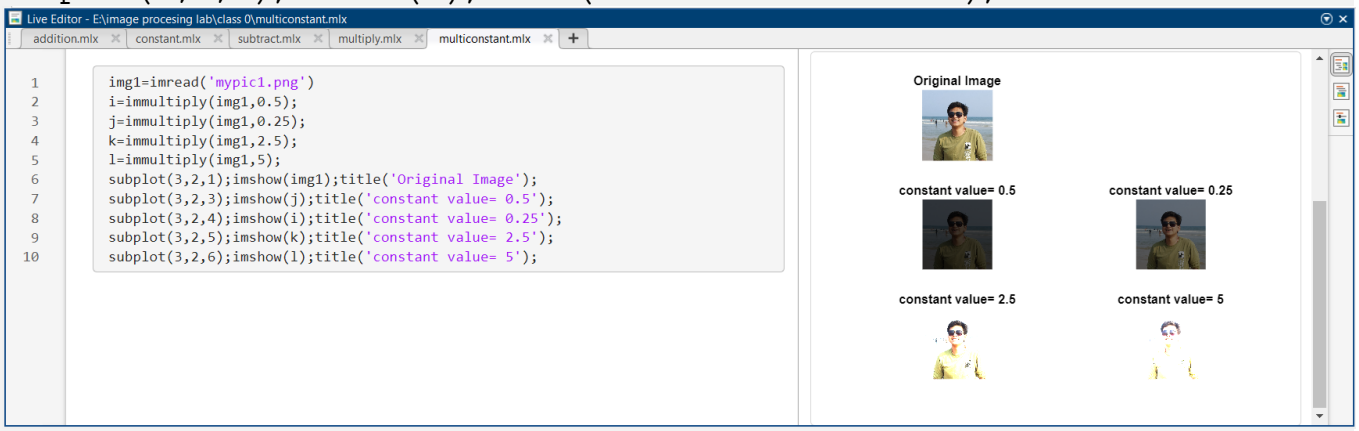


Figure 7: use of immultiply( ) to multiply a constant to images in in MATLAB

Image Histogram

- An image histogram is a gray-scale value distribution showing the frequency of occurrence of each gray-level value.
- *imhist()* displays a plot of the histogram. If the input image is an indexed image, then the histogram shows the distribution of pixel values above a color bar of the colormap cmap.
- The following code describes how to plot a histogram of an image without the inbuilt function.

CODE

```
j=imread('xray.jpg'); % here we are reading the image
i=rgb2gray(j); % here we are converting the RGB image to gray scale
[rows,column]=size(i); % we are accessing the image size
histvalue=zeros(1,255);% creating a zero matrix to store the frequency
for Rows =1:rows % to traverse through each row of the image
   for Columns=1:column % to access each element of the row
       x=i(Rows,Columns); % assigning a variable the intensity value
       histvalue(1,x+1)=histvalue(1,x+1)+1; %updating the frequency
   end
end
%plotting the original image and the histogram
k=0:1:254;
subplot(2,1,1); imshow(j);title('The image')
subplot(2,1,2); plot(k,histvalue);title('Histogram of image');grid on;
```

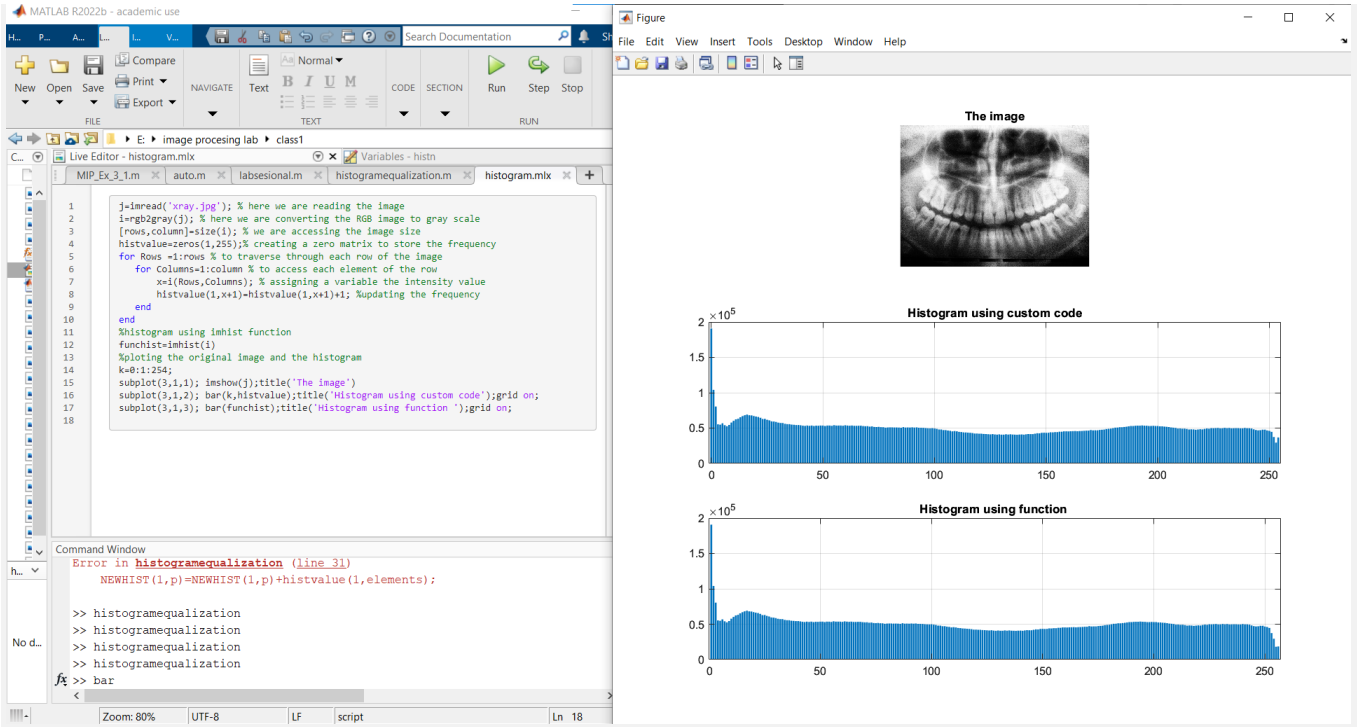


Figure 8: code for plotting the histogram of an image in MATLAB

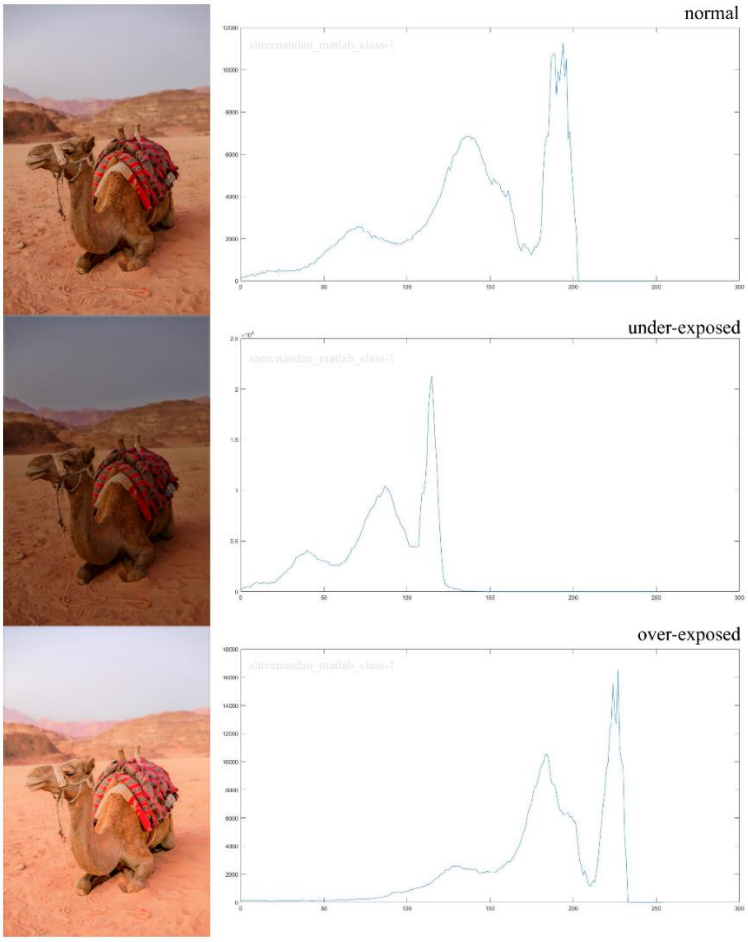


Figure 9: Variation in the histograms of normal, underexposed and over exposed images.

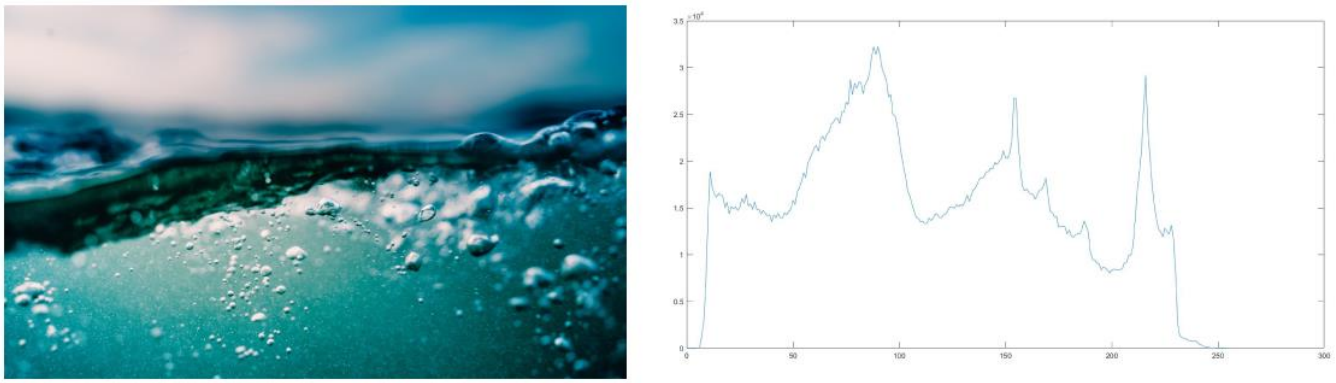


Figure 10: Histograms of underwater images.

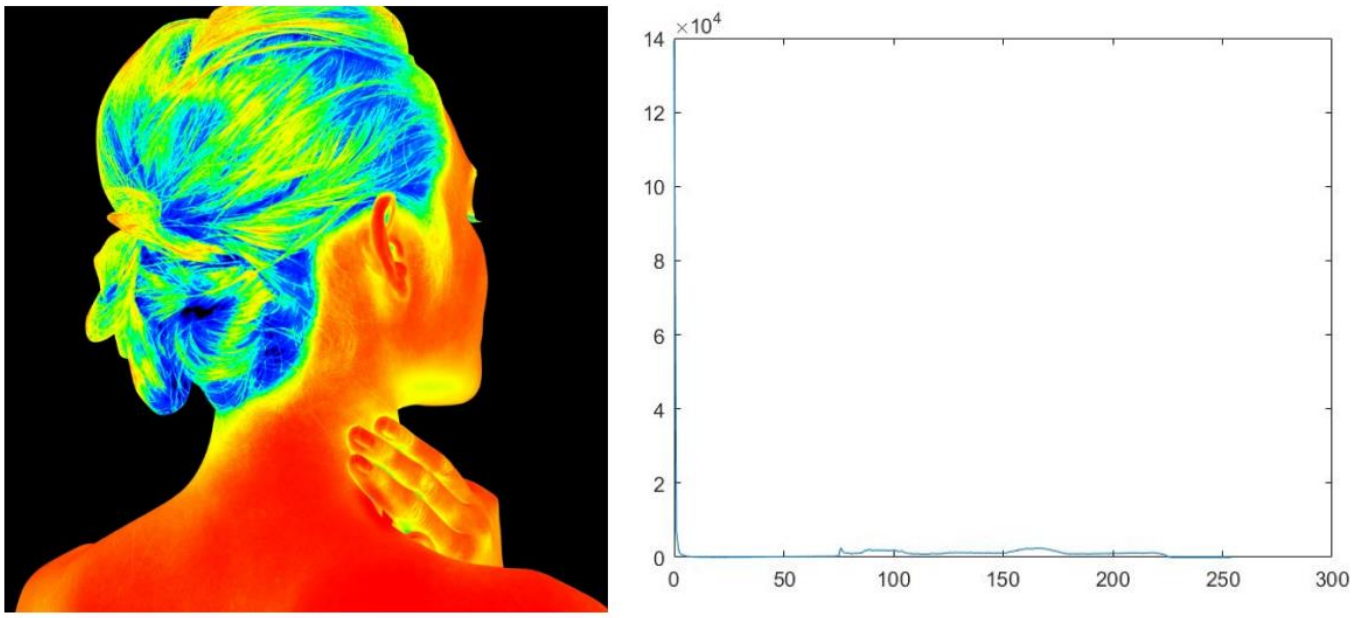


Figure 11: Histogram of thermal images.

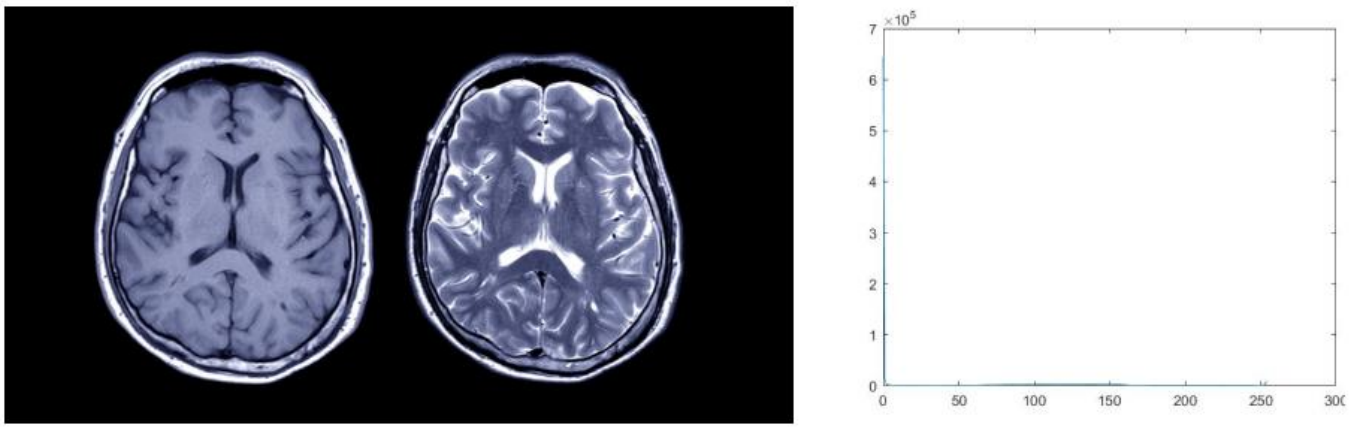


Figure 12: Histogram of medical image (MRI both T1 and T2 waited images).

Exercise(Assignment)

Exrcise1.

- B = flip(A,dim) reverses the order of the elements in A along dimension dim. For example, if A is a matrix, then flip(A,1) reverses the elements in each column, and flip(A,2) reverses the elements in each row.

CODE

```
img=imread("mypic3.png");
i1=flip(img,1); % this flips the iamge horizontally
i2=flip(img,2); %this flips the image vertically
i3=flip(i1,2);%this flip the image both horizontally and vertically


subplot(2,2,1);imshow(img);title('Original Image');
subplot(2,2,2);imshow(i1);title('horizontally fliped');
subplot(2,2,3);imshow(i2);title('vertically fliped');
subplot(2,2,4);imshow(i3);title('flipped both
wise');subplot(3,2,6);imshow(l);title('constant value= 5');
```

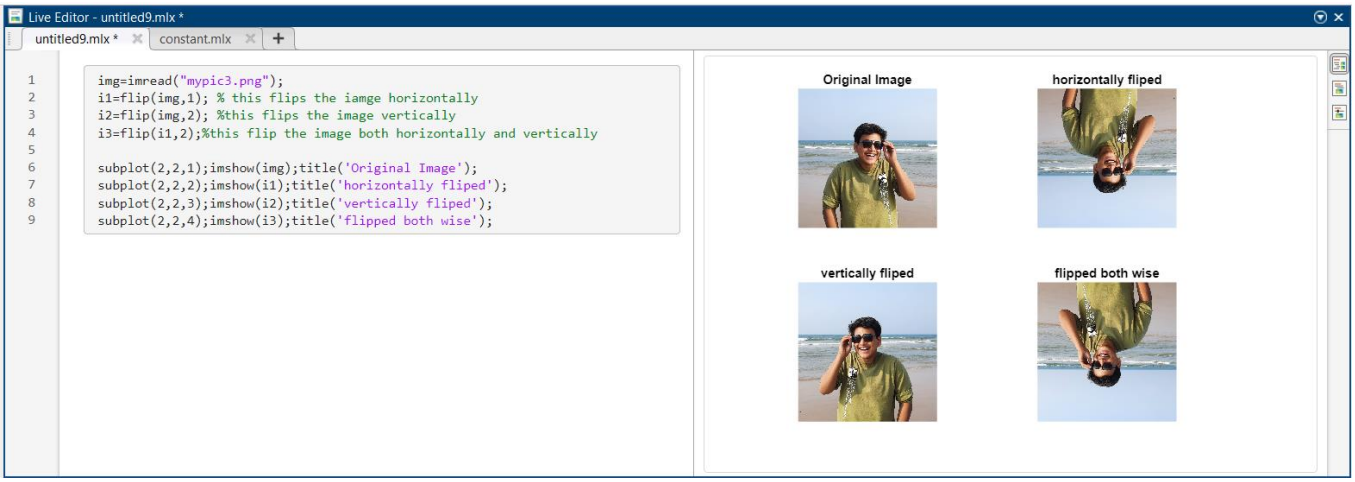
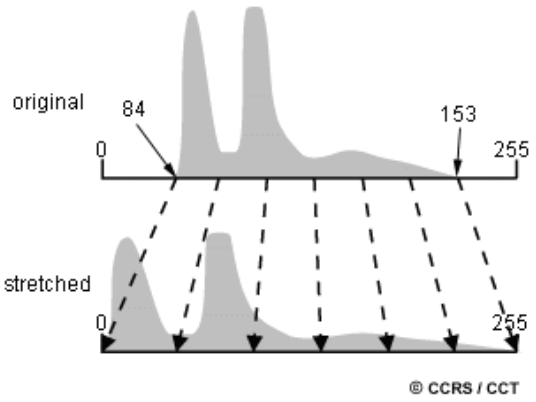


Figure 13: use of flip( ) to flip images in MATLAB

Histogram Equalization



- Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

Figure 14: graphical depiction of histogram equalization.

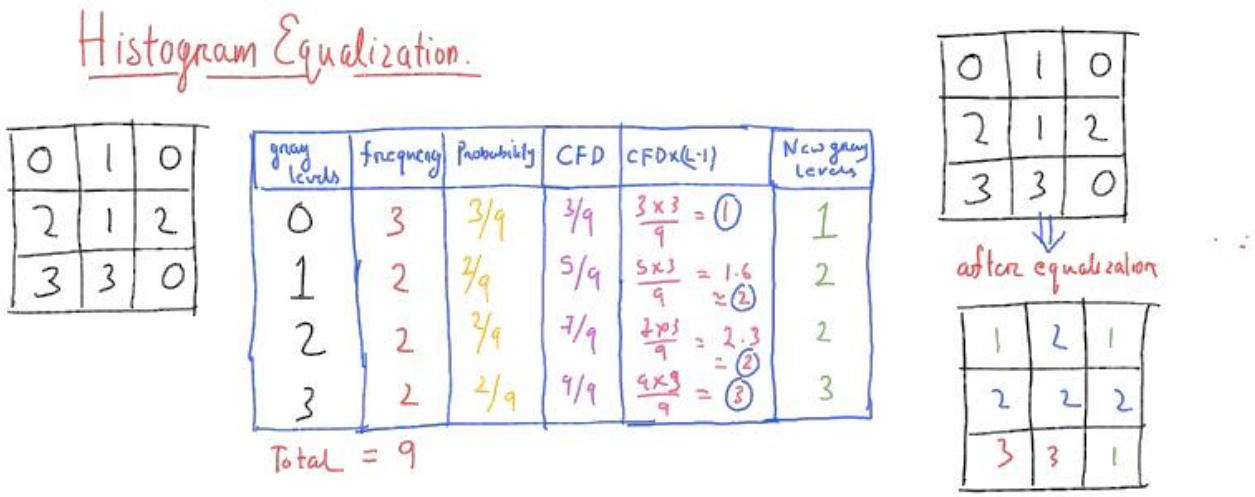


Figure 15: process of finding histogram equalization.

CODE

```
y=uigetfile('*.*');
j=imread(y);
i=rgb2gray(j);
rows=height(i);
column=width(i);
histvalue=zeros(1,256);
for Rows =1:rows
   for Columns=1:column
       x=i(Rows,Columns);
       histvalue(1,x+1)=histvalue(1,x+1)+1;
   end
end
%histogram ends here-----------------------------------
%probability ------------------------------------------
px=zeros(1,256);
for columns=1:256
    px(1,columns)=histvalue(1,columns)/(rows*column);
end
%cfd finding ------------------------------------------
cdf=zeros(1,256);
cumulative=0;
for columns=1:256
    cdf(1,columns)=px(1,columns)+cumulative;
```

```
    cumulative=cumulative+px(1,columns);
end
%cfd normalising ------------------------------------
CDF=255*cdf;
newhist=round(CDF);
NEWHIST=zeros(1,256);
for elements=1:256
    newgraylevel=newhist(1,elements)+1;

NEWHIST(1,newgraylevel)=NEWHIST(1,newgraylevel)+histvalue(1,elements);
end


new=histeq(i);
histn=imhist(new);


figure();
k=0:1:255;
subplot(2,2,1);bar(k,imhist(i));title('Histogram using imhist
function')
subplot(2,2,2);bar(k,histvalue);title('Histogram using custom code')
subplot(2,2,3);bar(histn);title('Histogram eualization using histeq
function')
subplot(2,2,4);bar(k,NEWHIST);title('Histogram eualization using
custom code')
```

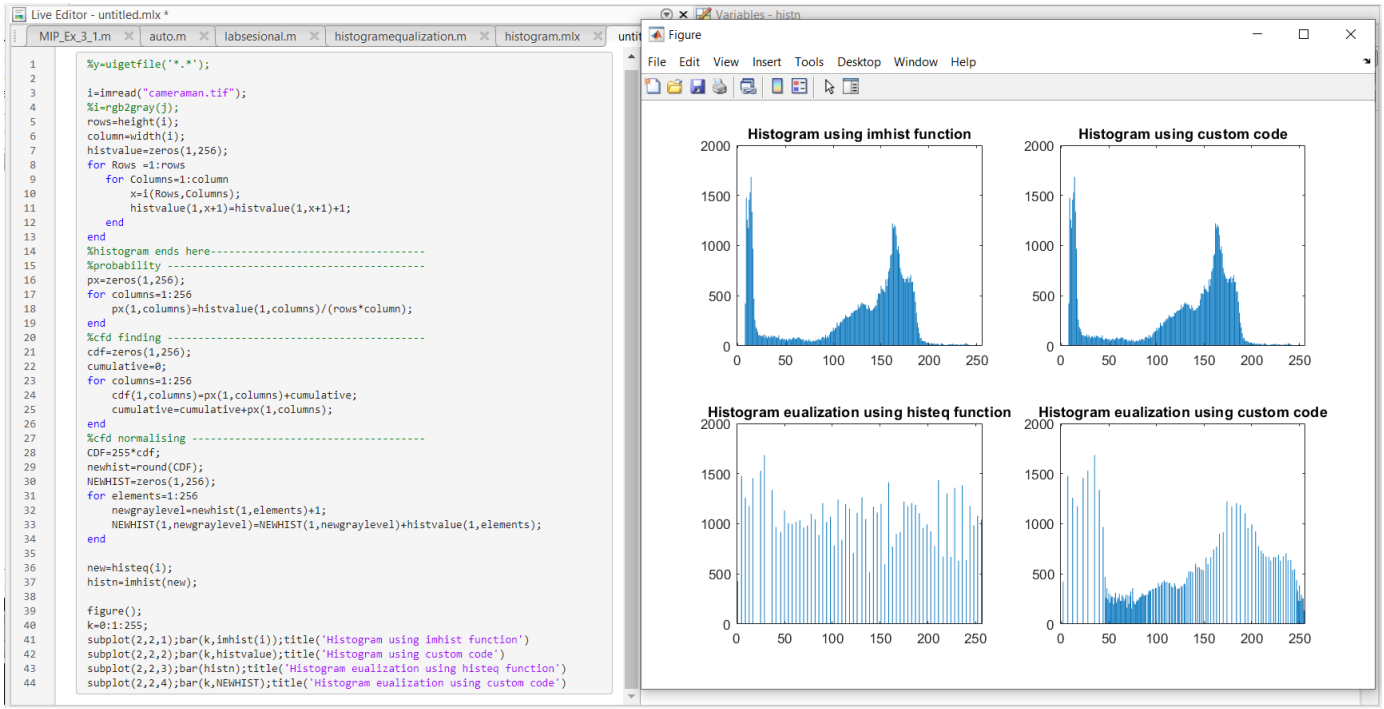


Figure 16: plots showing histograms before and after equalization.

Auto Focus

- Autofocus is the process of improving the image quality on the basis of sharpness of the image. Image which is sharp will have frequency (change in intensity) more as compared to an image of same dimension and same content with less blur.
- By calculating the frequency over all the pixel, we can conclude which image is more focused.

CODE

```
function [temp]=autofocus(b)
[m n]=size(b);
temp=double(0);


for i=1:m-1
    for j=1:n
        k(i,j)=b(i+1,j)-b(i,j); %substracting the adjacent row pixels%
        temp=temp+double(k(i,j));% adding all the differenced pixels%
    end
end
end
```

Function 1:function for calculating the cumulative frequencies.

CODE

```
%% reading the images%%%%%
fprintf('\n upload an image1 \n');
x=uigetfile('*.*');                    %to get access to the image
X=imread(x);                                %read the image
fprintf('\n upload 2nd image\n');
y=uigetfile('*.*');                         % to get access to the
image
Y=imread(y);                               % read the image
s1=size(X);                           % to find the size of the image
if length(s1)==3      % to find it is color image or gray scale image
X=rgb2gray(X);           %if color image convert it into gray scale
end
s2=size(Y);                    % to find the size of the image
if length(s2)==3    % to find it is color image or gray scale image
Y=rgb2gray(Y);       %if color image convert it into gray scale
end
%% auto focusing %%%%%%%%%%%
k1=autofocus(X);                            %call for auto focus
function
```

```
k2=autofocus(Y);
temp=0;
while(k2>k1)
    fprintf('\n present image is better focused than previous image');
    R=input('still u want to check then type 1 if not 0\n'); % giving
input 1 or 0
    if(R==1)
        fprintf('\n upload another image\n');
        x1=uigetfile('*.*');
        X1=imread(x1);              %if input is 1 read another image
        s3=size(X1);                        %to find size of the image
        if length(s3)==3
        X1=rgb2gray(X1);            % changing color image to gray
        end
        k2=autofocus(X1);                    % call for auto focus
function
     else
         fprintf('\n present image  is best focused\n');
         temp=1;
       break;
   end
end
 if(temp==0)
    fprintf('\n previous image is better focused than  present image\n
go to the back step\n');
 endend
```

Figure 17: command window showing which image is more focused.

Shreenandan Sahu |120BM0806