Question 1

Write a MATLAB program to apply the bilinear interpolation. Download Figure 2.20 (a), M(3692)×N(2812), from the textbook web site as the testing image, then

a) Resize the testing image to 0.5M×0.5N and 0.125M×0.125N, respectively;

b) Resize the 0.5M×0.5N and 0.125M×0.125N images back to the original size of M×N, respectively.

As a minimum requirement of your mini project report, you should use MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio) to compare your results obtained from step b) with the original Figure 2.20 (a). Then analyze your results and provide your observations and conclusions in a typed report.

Note: The Peak Signal to Noise Ratio (PSNR) is defined as:

$$PSNR = 10\log_{10}\left(\frac{G_{Max}^2}{MSE}\right),$$

where G_{Max}^2 is the maximum gray level of the original $M\times N$ testing image $f(x_i, y_j)$, and MSE is the Mean Square Error between $f(x_i, y_j)$ and the restored image $\hat{f}(x_i, y_j)$

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[f(x_i, y_j) - \hat{f}(x_i, y_j)\right]^2.$$

Solution:

Interpolation is a process in which we use the available values or known data to estimate the unknown values at particular locations. This technique is used widely in image processing in order to zoom, shrink, rotate, or correct the images.

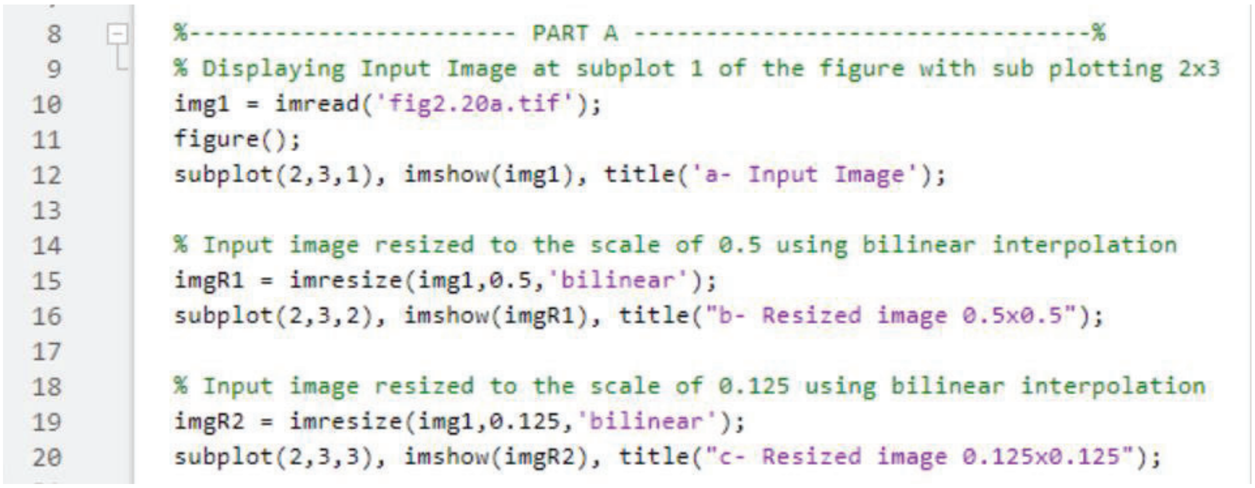
In a ***bilinear interpolation,*** values of the 4-nearest neighboring pixels are used to estimate the values at a specified location.

<u>Given Data:</u>
Testing Image:
Textbook Image Fig 2.20 (a), having rows M or height of 3692 and columns N or width of 2812.

Part a:
Resize testing image to 0.5Mx0.5N and 0.125Mx0.125N

Part b:
Resize resultant images of *Part a*, back to the original size of testing image MxN i.e., 3692x2812

MATLAB Code:



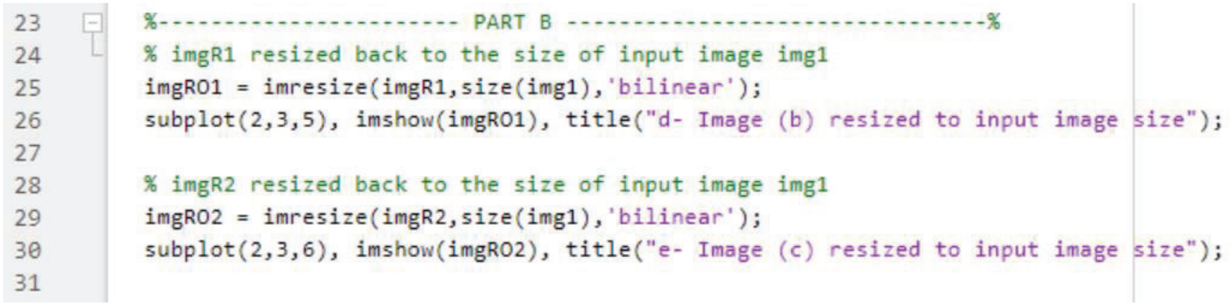
```
 8  %---------------------- PART A -------------------------------%
 9  % Displaying Input Image at subplot 1 of the figure with sub plotting 2x3
10  img1 = imread('fig2.20a.tif');
11  figure();
12  subplot(2,3,1), imshow(img1), title('a- Input Image');
13  
14  % Input image resized to the scale of 0.5 using bilinear interpolation
15  imgR1 = imresize(img1,0.5,'bilinear');
16  subplot(2,3,2), imshow(imgR1), title("b- Resized image 0.5x0.5");
17  
18  % Input image resized to the scale of 0.125 using bilinear interpolation
19  imgR2 = imresize(img1,0.125,'bilinear');
20  subplot(2,3,3), imshow(imgR2), title("c- Resized image 0.125x0.125");
```

FIGURE 1: CODE FOR PART A

Here, after reading the testing image into the variable img1 at line 9, it is displayed in the figure at sub plot 1 via line 12.

Next, the imresize function provided by MATLAB is used at line 15 and 19 which takes the image to be resized, the desired scale or size for resizing and the interpolation method name that should be used to perform resizing.

These resized images (imgR1 and imgR2) are displayed at subplots 2 and 3.



```
23  %---------------------- PART B -------------------------------%
24  % imgR1 resized back to the size of input image img1
25  imgRO1 = imresize(imgR1,size(img1),'bilinear');
26  subplot(2,3,5), imshow(imgRO1), title("d- Image (b) resized to input image size");
27  
28  % imgR2 resized back to the size of input image img1
29  imgRO2 = imresize(imgR2,size(img1),'bilinear');
30  subplot(2,3,6), imshow(imgRO2), title("e- Image (c) resized to input image size");
31  
```

FIGURE 2: CODE FOR PART B

Now for the *Part b*, imresize function is used again at line 25 and 29 in order to resize the images obtained in *Part a* back to the size of original testing image img1. Therefore, here for the scale parameter, it takes the size(img1) which returns the size of the original input or testing image.

The images resized back to the original size (imgRO1 and imgRO2) are displayed at subplots 5 and 6 (line 26 and 30).

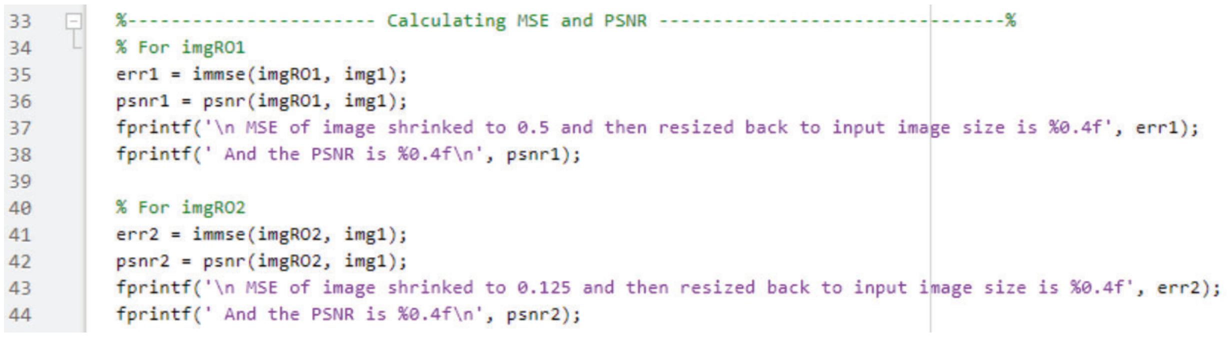
Now as in the requirement of the problem given in the question, **MSE and PSNR** are to be calculated to compare the results of *Part b*, with the original testing image.

These two measures ***are used to compare the quality of reconstructed image*** w.r.t to original image.

So, the **MSE** indicates the cumulative or mean squared error between the original and the resized image, therefore *lower the value of MSE lower the error will be*.

On the other hand, **PSNR** is the calculation of Peak Signal to Noise Ratio. The signal is basically referred to as the original data or input image in this case and noise is generally the error introduced during the reconstruction of the input image. Therefore, *a higher value of PSNR indicates greater quality of output image*.

These measures can be calculated using the built-in functions immse() and psnr() provided in MATLAB as follows:



```
%----------------------- Calculating MSE and PSNR --------------------------------%
% For imgRO1
err1 = immse(imgRO1, img1);
psnr1 = psnr(imgRO1, img1);
fprintf('\n MSE of image shrinked to 0.5 and then resized back to input image size is %0.4f', err1);
fprintf(' And the PSNR is %0.4f\n', psnr1);

% For imgRO2
err2 = immse(imgRO2, img1);
psnr2 = psnr(imgRO2, img1);
fprintf('\n MSE of image shrinked to 0.125 and then resized back to input image size is %0.4f', err2);
fprintf(' And the PSNR is %0.4f\n', psnr2);
```

FIGURE 3: CODE FOR MSE AND PSNR CALCULATIONS

Results & Conclusion

After running the program file q1.m, the reconstructed images for the *Part a* and *Part b* along with the input images are displayed as shown below:

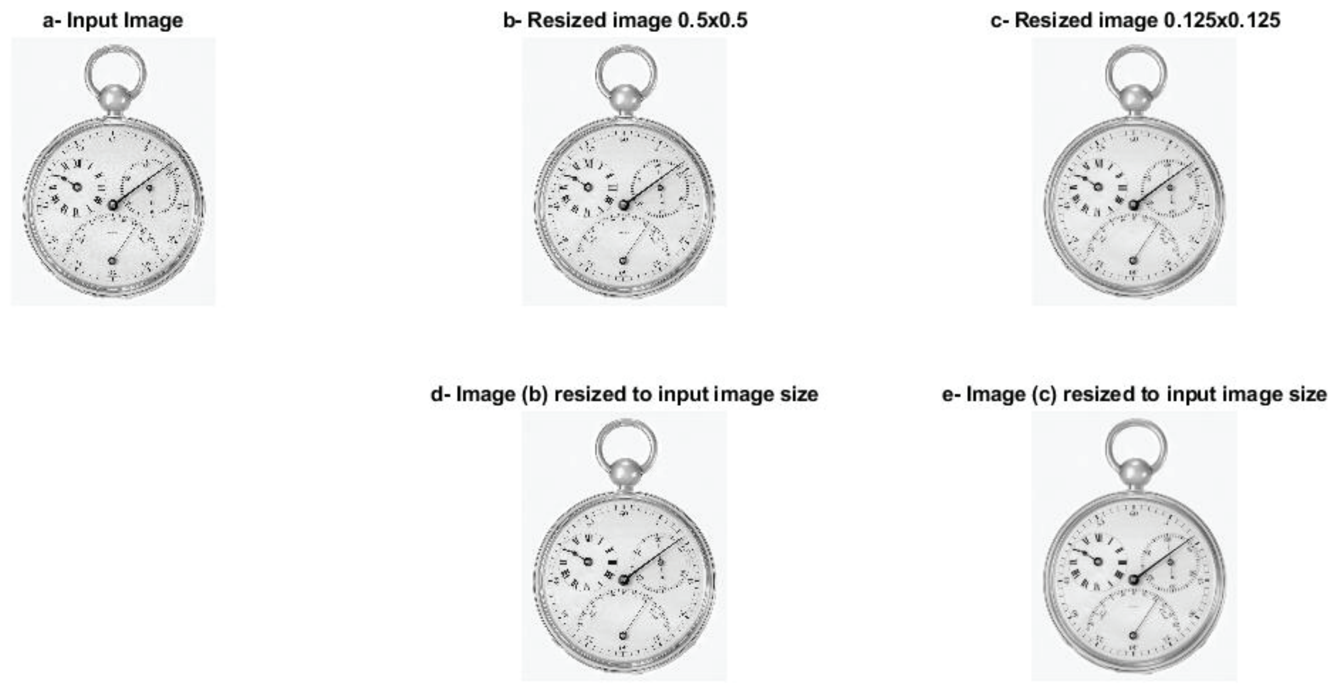

a- Input Image
b- Resized image 0.5x0.5
c- Resized image 0.125x0.125
d- Image (b) resized to input image size
e- Image (c) resized to input image size

FIGURE 4: OUTPUT OF CODE IN FIGURE 1 & FIGURE 2

From the above results, we can see that as we shrink the image (reduce the dpi) the quality of the images is reduced. Figure 4(b) is obtained by resizing the Figure 4(a) i.e. the input image to 0.5x0.5 and then resized back to the input image size to get the Figure 4(d).

Similarly, Fig 4(c) is when input image is resized to 0.125x0.125 and then further resizing it back to the input image size we get Fig 4(e).

To discuss how the quality of images vary, below figure shows the images in Figure 4 in a zoomed manner with focus on the right side of the compass where we have some more details including needles and digits.

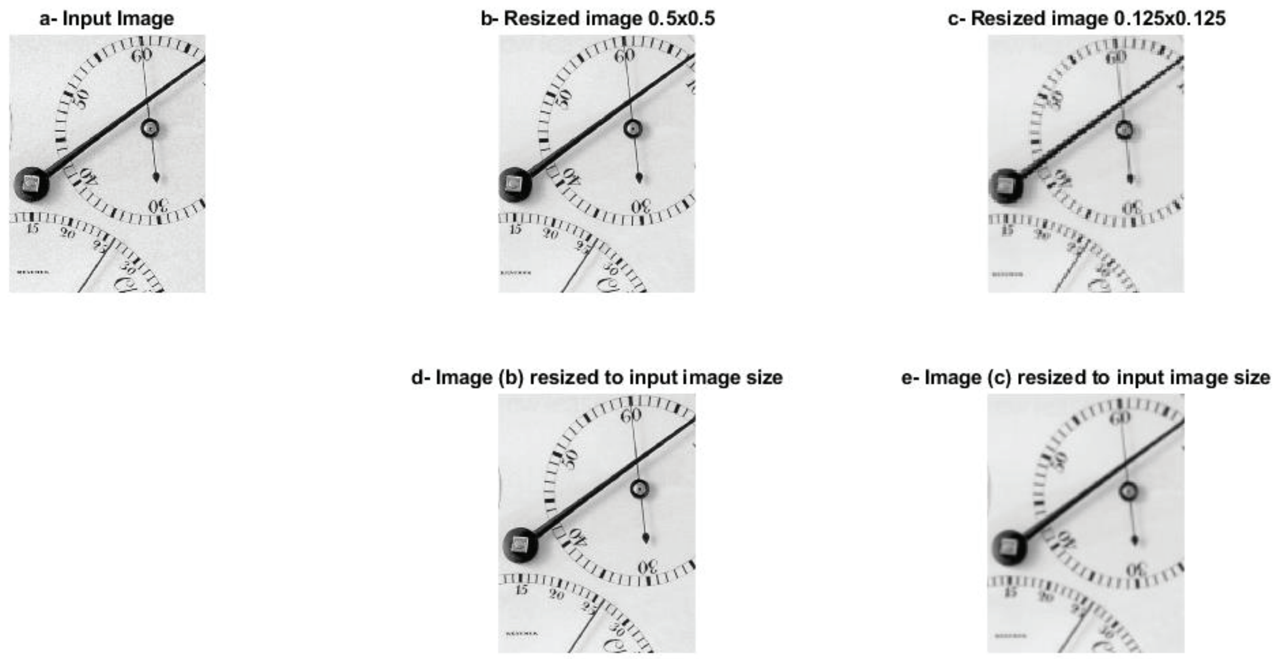

a- Input Image
b- Resized image 0.5x0.5
c- Resized image 0.125x0.125
d- Image (b) resized to input image size
e- Image (c) resized to input image size

FIGURE 5

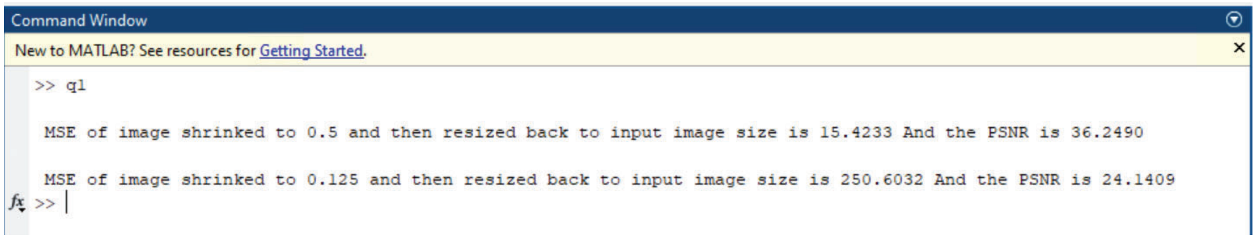
We can see the image (a) is most detailed, from texture of the background to the needles and digits. **Images (b), (d)** are quite similar to (a) except that the background and digits visibility is quite diminished with slight distortion in the needle pointing at 60, but it's ***an acceptable compression of input image***.

Images (c), (d) appears to be washed off and blurry when compared with (a), the needles are distorted, digits are hard to read especially on the lower dial of the compass. The portioning of both dials is also distorted and appears to be washed off, so it is a ***lossy compression*** *resulting in compromised image quality*.

Hence, ***it can be concluded*** that reducing the dimensions of an image affects the quality of the image content and can make it appear washed off and unreadable below certain level such that resizing it back to the original size doesn't guarantee the recovery of the original quality as it is prone to loss/noise.

Now if we look at the results of MSE and PSNR calculations of image d and e above conclusion can be verified.

So, if we run the program q1.m then the result of code in Figure 3 gets printed on the console as shown below:


Command Window

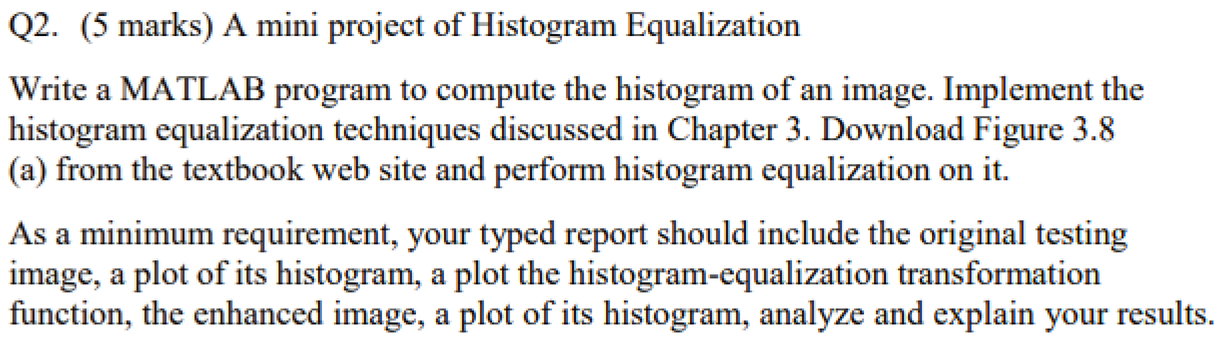
New to MATLAB? See resources for Getting Started.

```
>> q1

 MSE of image shrinked to 0.5 and then resized back to input image size is 15.4233 And the PSNR is 36.2490

 MSE of image shrinked to 0.125 and then resized back to input image size is 250.6032 And the PSNR is 24.1409
fx >> |
```

As stated earlier, the lower MSE and the higher PSNR indicates a better-quality image.

So, from the above results for Figure 4, image (d) we have MSE 15.4233 and PSNR 36.2490. Whereas for image(e) we have MSE 250.6032 and PSNR 24.1409. Hence, ***the image (d) is better quality compression of the input image (a) as compared to the image(e).***

Question 2


Q2. (5 marks) A mini project of Histogram Equalization

Write a MATLAB program to compute the histogram of an image. Implement the histogram equalization techniques discussed in Chapter 3. Download Figure 3.8 (a) from the textbook web site and perform histogram equalization on it.

As a minimum requirement, your typed report should include the original testing image, a plot of its histogram, a plot the histogram-equalization transformation function, the enhanced image, a plot of its histogram, analyze and explain your results.

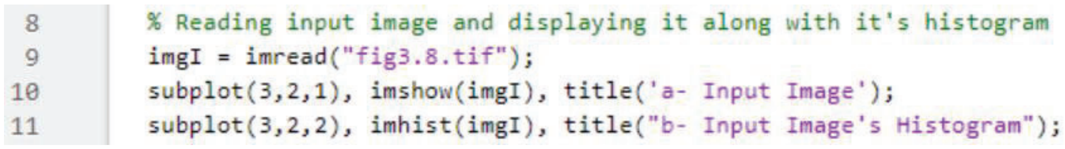
Solution:

Histogram Equalization is a process widely used in image processing to adjust the contrast of an image using the image histogram and a transformation function which if known can be used to recover the original histogram.

Generally, a ***histogram*** is a graph which is used to represent the frequency or number of occurrences of something over a given range. Similarly, the image's histogram basically represents the number of pixels against each value of intensity level for an image having intensity levels in the range of [0, L-1].

A histogram for an image can be easily obtained in MATLAB using the **imhist()** function for which we just have to specify the image and optionally can mention the number of bins a histogram should have.

Below is the snippet of code from the file q2.m to read the test image file fig3.8.tif into variable imgI (line 9), then displays this image at sub plot 1 (line 10), and the line 11 displays the histogram of this input image at subplot 2.

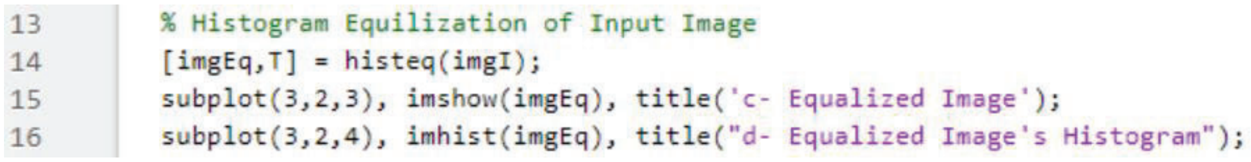


```
8     % Reading input image and displaying it along with it's histogram
9     imgI = imread("fig3.8.tif");
10    subplot(3,2,1), imshow(imgI), title('a- Input Image');
11    subplot(3,2,2), imhist(imgI), title("b- Input Image's Histogram");
```

If the image is grayscale, imhist() function takes the default bins to be 256, i.e. 0-255

Histogram Equalization is task which attempts to equally distribute the number of pixels over the whole intensity range in order to *obtain a well-balanced contrast image that has a wider tones of gray in case of a gray scale image*, *resultantly having a uniform or flattened histogram* over the intensity levels range. A transformation function can be determined automatically for histogram equalization that should produce an image for which the histogram is flattened of uniform.

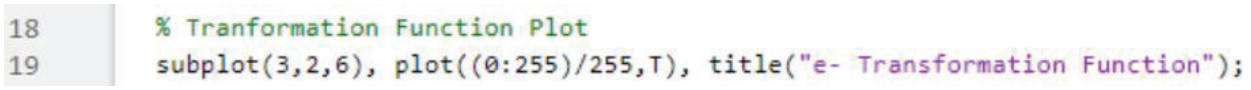
To achieve histogram equalization of an image in MATLAB, there's a function provided called **histeq()** which returns the equalized image and we can also get the transformation function T used by it for equalization. From q2.m program file the code to perform this task is as follows:



```
13    % Histogram Equilization of Input Image
14    [imgEq,T] = histeq(imgI);
15    subplot(3,2,3), imshow(imgEq), title('c- Equalized Image');
16    subplot(3,2,4), imhist(imgEq), title("d- Equalized Image's Histogram");
```

At Line 14 using histeq(imgI), we get the equalized image of imgI and store it in variable imgEq along with the transformation function in variable T. Then line 15 displays the equalized image at subplot 3 and imhist is used at line 16 to get the corresponding histogram of equalized image at subplot 4.

Now we can plot the transformation curve for pixels at each intensity level in input image against the pixels at corresponding intensity level in an equalized image. As the intensity level range for gray scale image is 0 to 255, so the following code is used to plot the transformation function's T curve for the input and equalized images' pixels over the range of intensity levels:



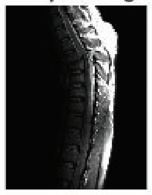
```
18    % Tranformation Function Plot
19    subplot(3,2,6), plot((0:255)/255,T), title("e- Transformation Function");
```

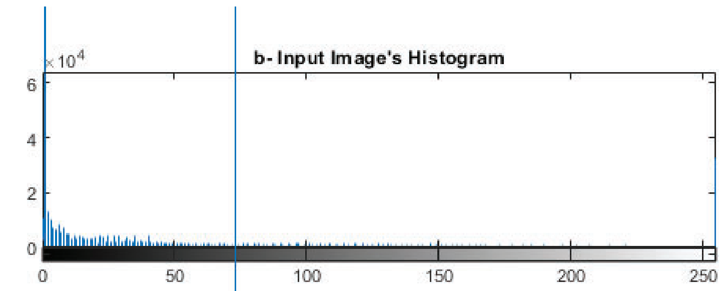
<u>**Results & Observation:**</u>

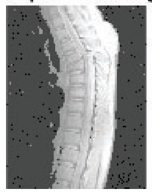
Running the q2.m which includes the code snippets explained above gives us the following figures in the result:

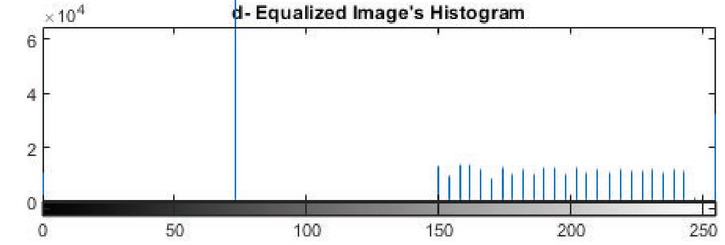
a) Input Image
b) Input Image's Histogram
c) Equalized Image
d) Equalized Image's Histogram
e) Transformation Function's curve

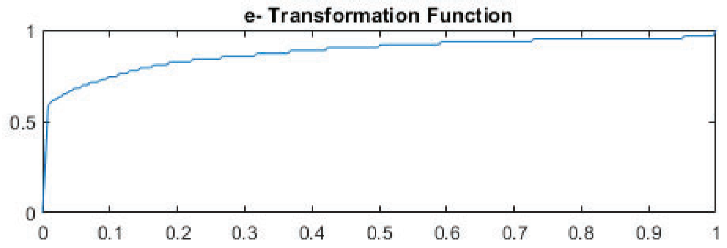
The output of the q2.m is shown below:





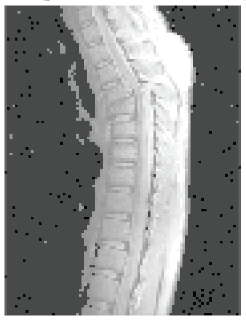






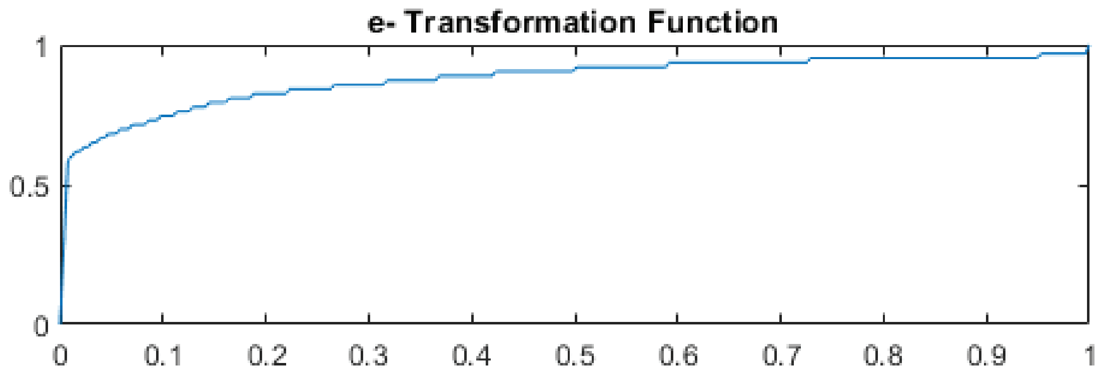
For the input image (a) we can see it is mostly dark as it's an X-ray of a bone. So, the corresponding ***histogram (b)*** *depicts the same graphically in which we have most of the pixels concentrated to the left corner* where the intensity levels are lower which represent the darker gray tones. So in such images *some useful information might not be easily viewable or completely hidden*.

Image (c) is the equalized image obtained for the image(a). This equalized image is ***more detailed and useful in order to examine the bone structure*** for the medical purposes. If we closely look at this picture, we can clearly see the ladder like structure of the bone which was not prominent in figure (a) and some part of it was also hidden.



The **histogram of equalized image** in image(d) shows mapping of the concentrated pixels of original image, into the upper or higher end of the gray scale intensity levels for a net equalized effect and hence the corresponding equalized image appears to be brighter, which might not be preferred in some cases.

Lastly the image(e) is the transformation curve



If we observe this curve, we can see a quick rise from 0 to 0.6 which is basically due to higher concentration of pixels at the darker gray scale intensity levels in the input image's histogram. After applying the transformation function, these concentrated darker pixels are shifted towards higher or brighter end of gray scale intensity level range for a net equalized affect.

Hence, due to this transformation the equalized image appears to be brighter now, which can be useful to examine the image in some cases like for the x-rays, but this transformation is not ideal in all cases due to concentration of pixels on the higher end instead of equally distributed.