Applied Computer Science

Course number – GACS-7205-001

Course Name – Digital Image Processing

Assignment #1

Problem 1

A mini project of Image Interpolation.

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

a. Resize the testing image to 0.5Mx0.5N and 0.125Mx0.125N, respectively
b. Resize the 0.5Mx0.5N and 0.125Mx0.125N images back to the original size of M x 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.

Solution

a. Original image

An interpolation is a method to be used to zoom and shrink images. In this exercise, I resized the original image to 0.5Mx0.5N and then to 0.125Mx0.125N respectively. In this case, I used bilinear interpolation, in which I used the four nearest neighbors to estimate the intensity at a given location.

For this approach, I started calculating the size of the original image. I used the function *size* provided by Matlab and saved the height and width. Then, I used the function *imresize* and set the parameters of the original image, the new size value and the resize method to use (bilinear). The image obtained didn't show a big difference from the original image, but the small changes I observed in the clarity of the image that I considered improved after the resizing.

Image resized to 0.5Mx0.5N:

```
clock = imread('clock.tif');
        [height, width, dim] = size(clock);
        imshow(clock);
        title('Original Size');
        clock_05 = imresize(clock,0.5,'bilinear');
        figure;
        imshow(clock_05);
        title('Image size 0.5Mx0.5N');
```

In the second scenario, when I resized to 0.125Mx0.125N. I used the function *imresize* as done before and set the parameters of the original image, the new size value and the resize method

to use (bilinear). The image result showed a decrease in terms of quality, I realised that some numbers blurrier, and the clarity of the small numbers was poor. I considered the image appearance had a reduction after the resizing.

Image resized to 0.125Mx0.125N

```
clock_125 = imresize(clock,0.125,'bilinear');
        figure;
        imshow(clock_125);
        title('Image size 0.125Mx0.125N');
```

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

After the process described above, I had to resize the images to the original size. To complete that, I used the same function resize, but this time I used the height and width parameters of the original image and I set bilinear as the method. The result exhibited a little change in the image resolution in the borders of the clock and the small numbers.

Image resized from 0.5Mx0.5N to the original size of M x N:

```
clock_back05 = imresize(clock_05,[height, width],'bilinear');
figure;
imshow(clock_back05);
title('Image back size from 0.5Mx0.5N to original');
```

In contrast with the previous resizing, this time the changes were more evident. The numbers, the borders and the colors of the clack looked different and affected by the resizing.

Image resized to 0.125Mx0.125N to the original size of M x N:

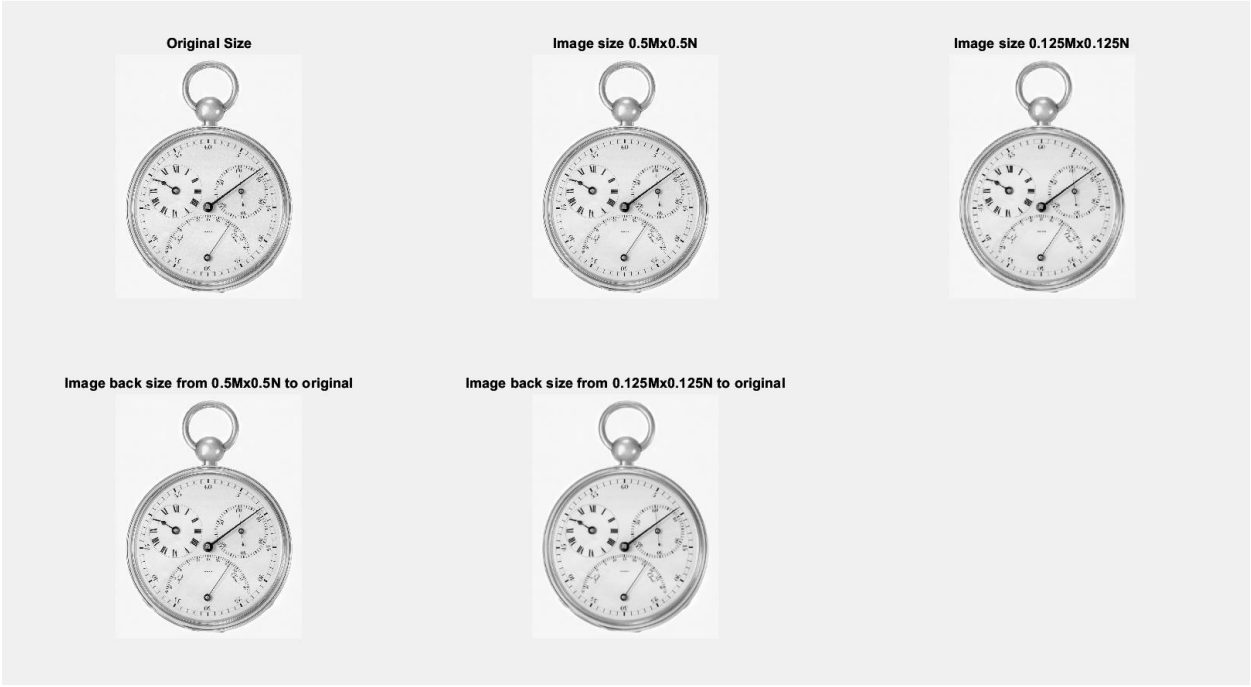
```
clock_back125 = imresize(clock_125,[height, width],'bilinear');
 figure;
 imshow(clock_back125);
 title('Image back size from 0.125Mx0.125N to original');
```

To validate the output for both images, I calculated the MSE(Mean Square Error) which provides the total of error in statistical models. I used the *immse* functions to calculate those measures. In the end, I got the average squared difference between the original image against the resized images. The MSE increases if the model error increases. In this scenario, I observed that the image with a greater error is the one that was resized to 0.125Mx0.125N, getting a value of 250.6032, while the image resized to 0.50Mx0.5N has only 15.4233, which as it demonstrated an error, it is not as bigger as the error of the former.

MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio) calculation:

```
%// Mean Square Error
err_resize05 = immse(clock_back05, clock);
fprintf(['\n The mean-squared error for image resize 0.5Mx0.5N ' ...
    'is %0.4f\n'], err_resize05);

err_resize0125 = immse(clock_back125, clock);
fprintf(['\n The mean-squared error for image resize 0.125Mx0.125N' ...
    'is %0.4f\n'], err_resize0125);

%// Peak Signal to Noise Ratio

peaksnr_resize05 = psnr(clock_back05, clock);

fprintf(['\n The Peak-SNR value for image resize 0.5Mx0.5N ' ...
    'is %0.4f'], peaksnr_resize05);

peaksnr_resize0125 = psnr(clock_back125, clock);

fprintf(['\n The Peak-SNR value for image resize 0.125Mx0.125N is' ...
    ' %0.4f'], peaksnr_resize0125);
```

```
The mean-squared error for image resizes 0.5Mx0.5N is 15.4233
The mean-squared error for image resizes 0.125Mx0.125Nis 250.6032

The Peak-SNR value for image resize 0.5Mx0.5N is 36.2490
The Peak-SNR value for image resize 0.125Mx0.125N is 24.1409
```



In addition to the MSE, I calculated the Peak Signal to Noise Ration. This measure helps to express and compare the effects of image enhancement on image quality. PSNR is used to identify which of both images produced a better result after resizing. For this calculation, I used the Matlab function *psnr*, and I set as its parameters the resized images and the original images.

The results indicate that the Peak-SNR value for image resize 0.5Mx0.5N is 36.2490, and the Peak-SNR value for image resize 0.125Mx0.125N is 24.1409. The former that has the higher PSNR has the better quality of the images resized.

Problem 2

A mini project of Histogram Equalization.

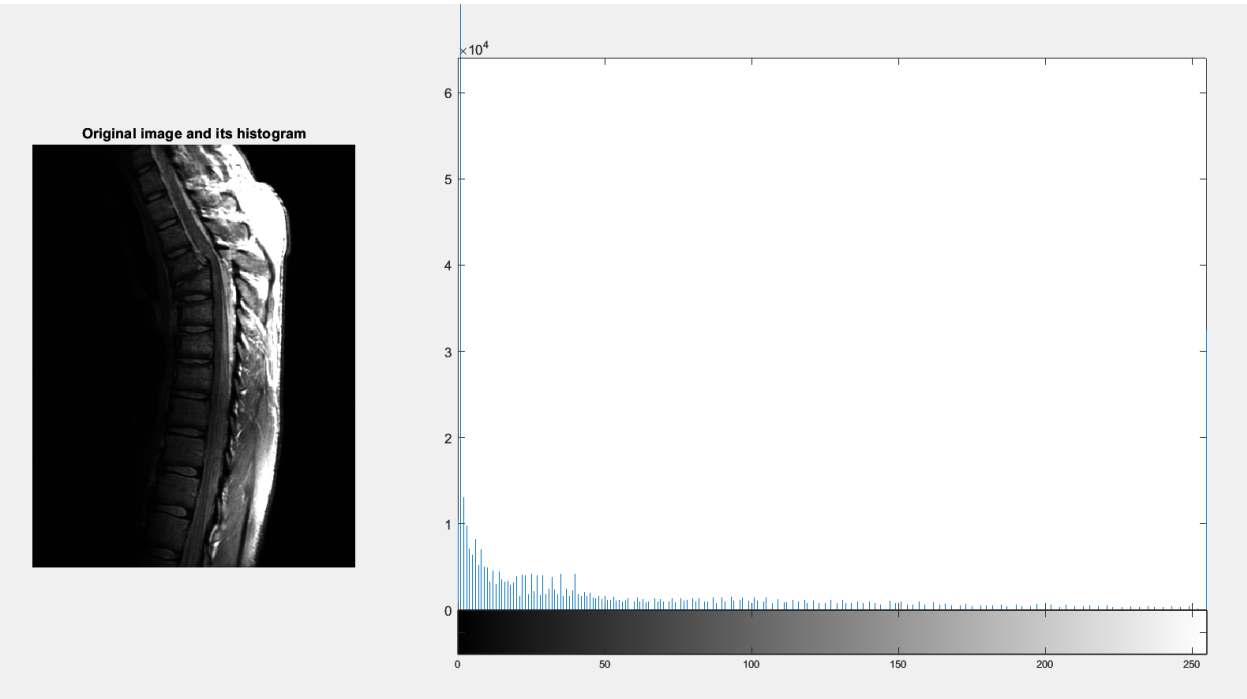
Write a MATLAB program to compute the histogram of an image. Implement the histogram equalization techniques discussed in Chapter 3. Download Figure 3.8 (a) from the textbook web site and perform histogram equalization on it.

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

A histogram is a graphical representation of the intensity distribution of an image. Histogram equalization is the process used to improve contrast in images. For this experiment first, I read the greyscale image and display the image and its histogram. I used the *imread* function of Matlab. Then, I plotted the image and its histogram to express the number of pixels for each intensity value considered. The original image had low contrast, and most of the pixel values were on the left side of the intensity range. That means that its most intensive pixels are located on that side of the image. That is evident due to that part of the image is darker.

```
I = imread("Fig038.tif")
figure
subplot(1,3,1)
imshow(I)
title('Original image and its histogram')
subplot(1,3,2:3)
imhist(I)
```



The histogram equalization helps us to adjust the contrast of the image. For this problem, I used the function *histeq* provided by MATLAB, and I plotted the new image and its histogram after the equalization. The new histogram displays the contrast-adjusted performed by the histogram equalization. This method increases the global contrast of the image and illuminates the darker sides of the images, resalting usable data by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

```
J = histeq(I);
figure
subplot(1,3,1)
imshow(J)
title('Image and its histogram after performed the histogram equalization')
subplot(1,3,2:3)
imhist(J)
```

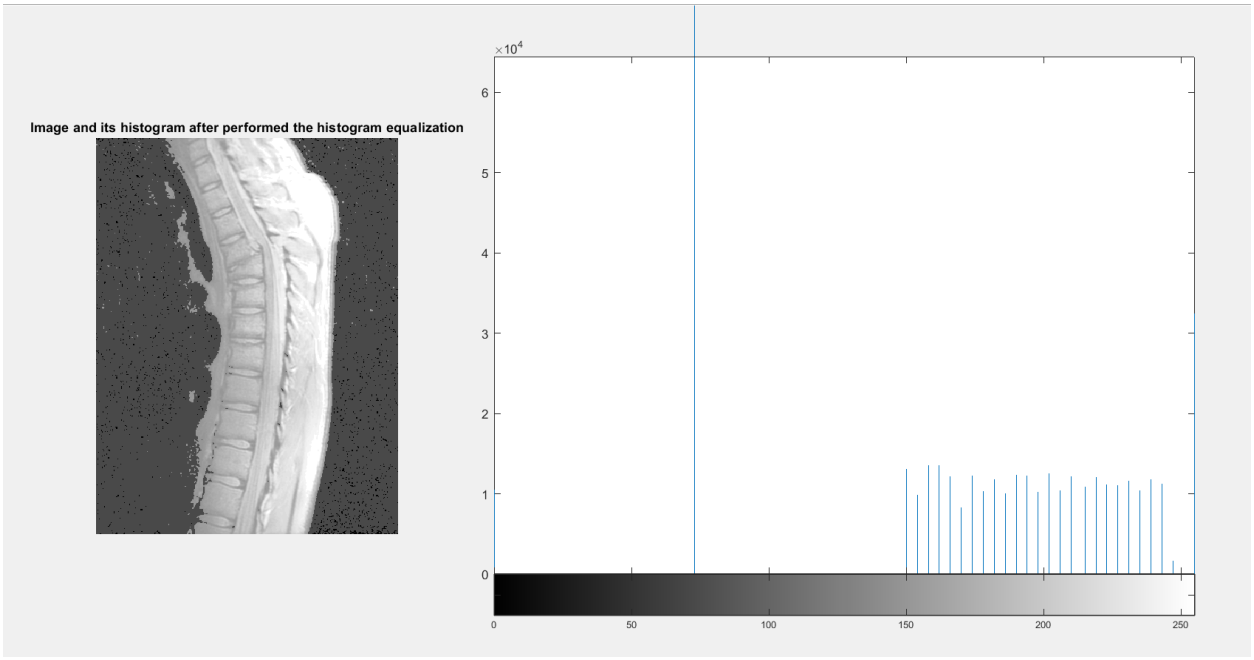


Image comparison: