Assignment 1: Image Enhancement and Histogram

ECE1512 Digital Image Processing and Applications

Hsuan-Ling (Celene) Chen

Student ID: 1002322202

Date due: Thursday, September 29, 2022; 5:00 pm

Date handed in: Thursday, September 29, 2022; 4:00 pm

1.0 Introduction

Image enhancement is the improvement of images to display more information and interpretability for human or image processing use. Doing image enhancement on an image can mean: highlighting interesting detail in images or removing noise from images. In the spatial domain technique involves the direct manipulation of image pixels. The focus of this project is to experiment with intensity transformations using two of the spatial domain technique, log and power-law transform to enhance the image in Fig. 3.8(a) from the book (GW) web site. Then a further histogram equalization is applied to the image.

2.0 Image Enhancement Using Intensity Transformations

2.1 Log Transformation

The log transformation equation is stated as following:

$$processed\ pixel\ value\ =\ c\ *\ log(1\ +\ original\ pixel\ value) \qquad (2.1\text{-}1)$$

Where c is a constant, and assuming that the original pixel values of the input image are positive. From the equation, the log transformation equation simply represents converting each of the input pixel value with its logarithm multiply with a scaling constant, c. The value of c can be calculated by substituting in the maximum output pixel value corresponding to the bit size of the image. The input image is an 8 bit image, which means its pixel value is ranged [0,255]. It is also known from the lecture that grey levels are assumed to be given the range [0.0, 1.0]. Hence, the 1.0 value is used to map with the 255 pixel value in order to obtain a constant, c that will produce a grey level image in the range [0.0, 1.0]. The constant, c is calculated to be 0.1803 after calculation with the given value.

$$c\ =\ output\ pixel/(log(1\ +\ input\ pixel_{max})) \qquad (2.1\text{-}2)$$

The logarithmic relationship between the input and output pixel values maps the low intensity values (darker pixels) of the input image into a wider range of output values showing more details, but compresses brighter pixels to a narrower range of output values. Therefore, the logarithmic transformation is useful to enhance low intensity values, but when applied to high intensity images, information of the image can be lost.

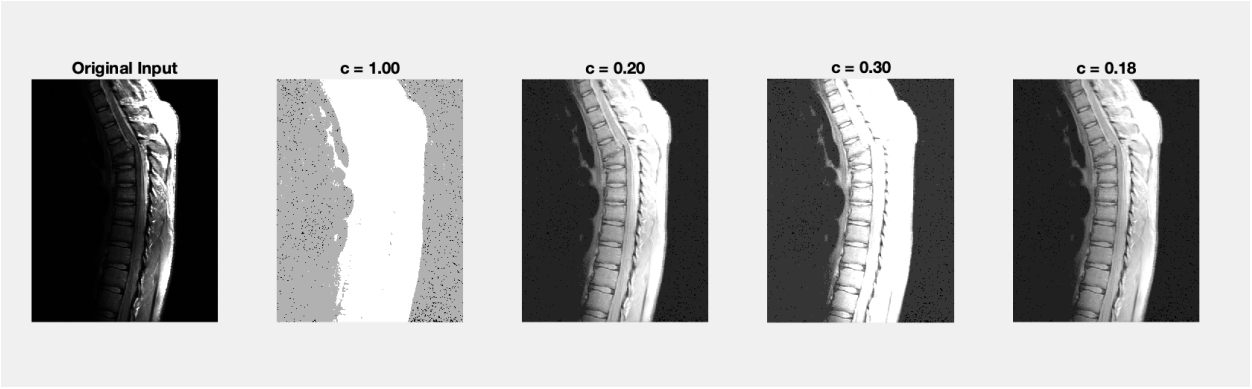

Figure 1. Comparison of Original image before log transformation and c = 1.00, 0.20, 0.30 and 0.19.

The constant equal to 0.18 creates a new image that shows more information than its original.

2.2 Power-Law transformation

The general form of the Power-Law transformation for image processing is the following:

$$output\ pixel\ value\ =\ c^*\ input\ pixel\ value^{gamma} \tag{2.2-1}$$

The function shows an exponential relationship between the input and output pixel values, multiplying with a scaling constant, c. Since the logarithm is the inversion of the exponential value of a number, this means the power-law behaves the same as the log transformation with gamma less than 1 in fractional values. It will map the low intensity input to a wider range, and vice versa for the high intensity input. The opposite will result for gamma more than 1. By setting c constant to 1, and the varying gamma will give a whole family of curves. The following image shows the different gamma values will highlight different details.

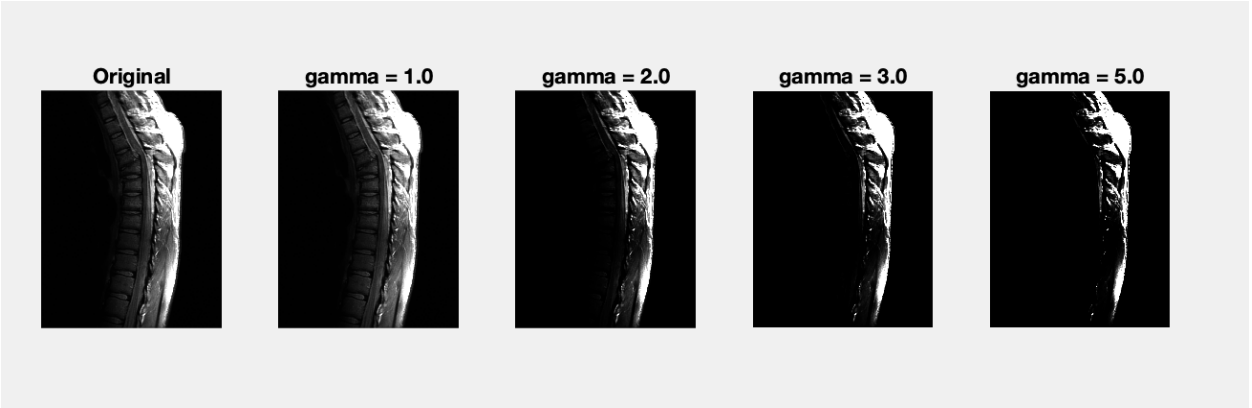

Figure 2. Power-Law Transformation with gamma larger than 1

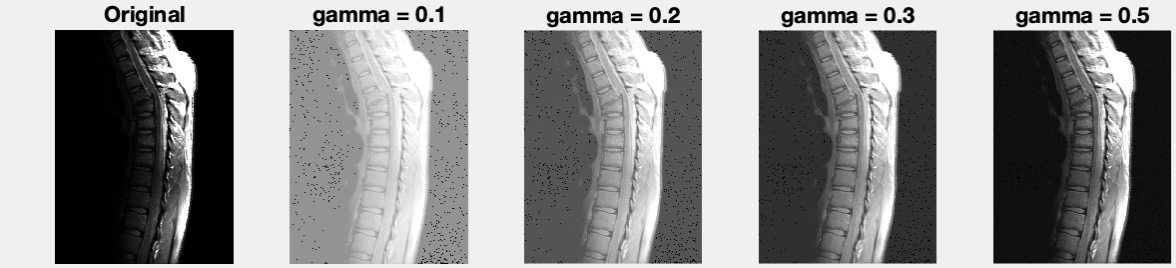

Figure 3. Power-Law Transformation with gamma smaller than 1

From inspection, the best visual result for the transformation is when gamma = 0.5 and c = 1.

3.0 Histogram

The histogram of a digital image consists of the intensity levels at specific pixel values. A usual histogram has intensity value as x-axis and the number of pixels with the intensity at the y-axis.
The histograms are frequently normalized by the total number of pixels in the image. It is a data visualization of the relationship.

3.1 Computer Program for Histogram

Matlab and many other computer applications have built in functions to create the histogram of an image. To write a computer program for computing the histogram of the image, each pixel value from the range [0, 255] is counted and the number of pixels with the intensity level is graphed into a histogram. See appendix A for the histogram code in Matlab.

3.2 Histogram Equalization Technique

The histogram equalization in section 3.3.1 studies about the stretching out of the frequencies in the image in order to improve images that are too dark or too bright. The transfer function discussed needs to be strictly monotonically increasing to guarantee that the ordering of the output intensity values will not have reversal of intensities. It should be in the range of [0, 255] for 8 bit images in order to guarantee that the range of the output will be the same as the input range.

Section 3.3.1 has derived the fundamental result from probability theory that as probability functions $p_r(r)$ and $p_s(s)$ are known, then their transfer function is continuous and differentiable. By substituting the pixel values, the scaled histogram-equalized values are obtained and rounded to the nearest integer.

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = \frac{1}{L-1} \qquad 0 \le s \le L-1 \tag{3.2-1}$$

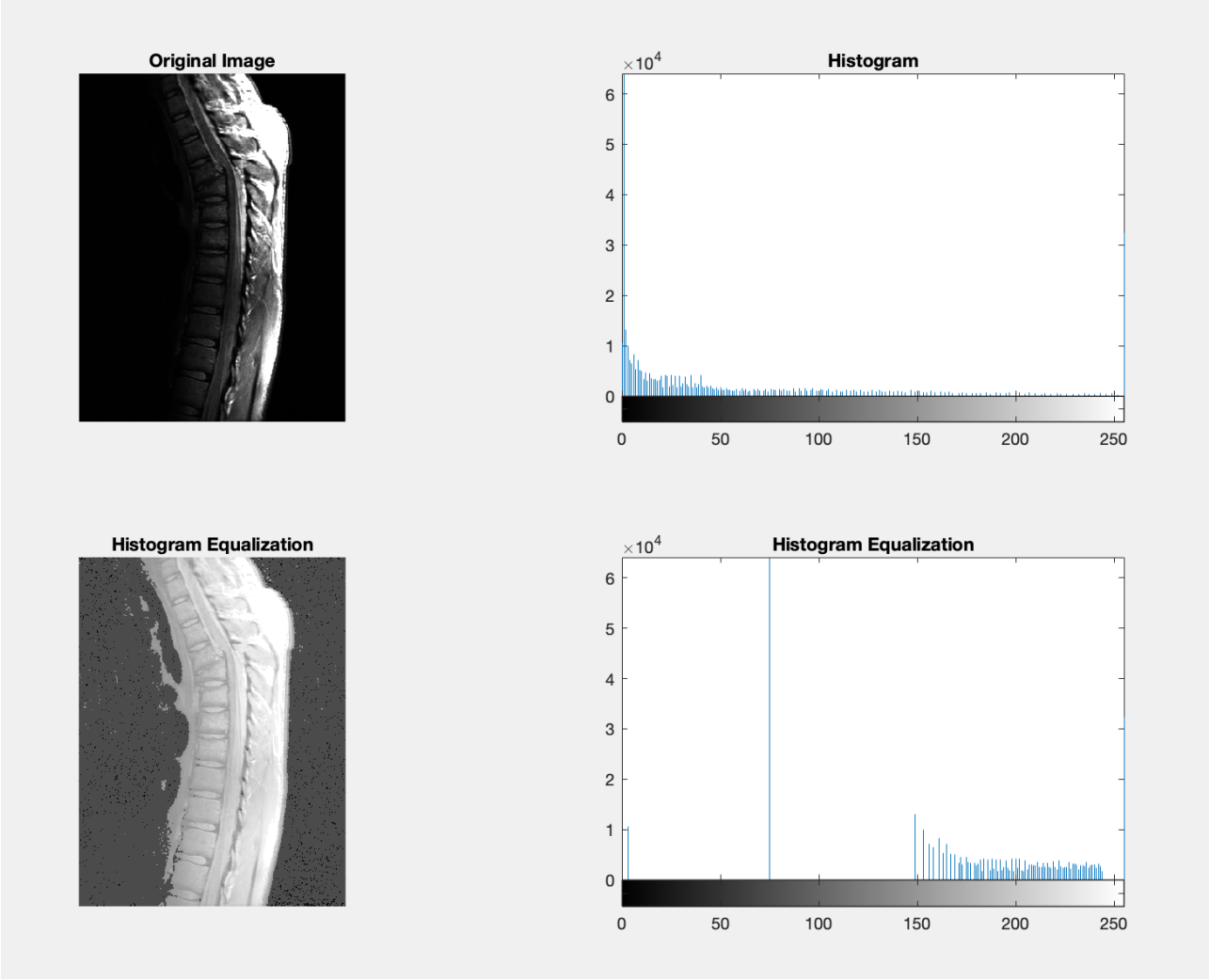

Figure 5. Histogram and Equalized Histogram

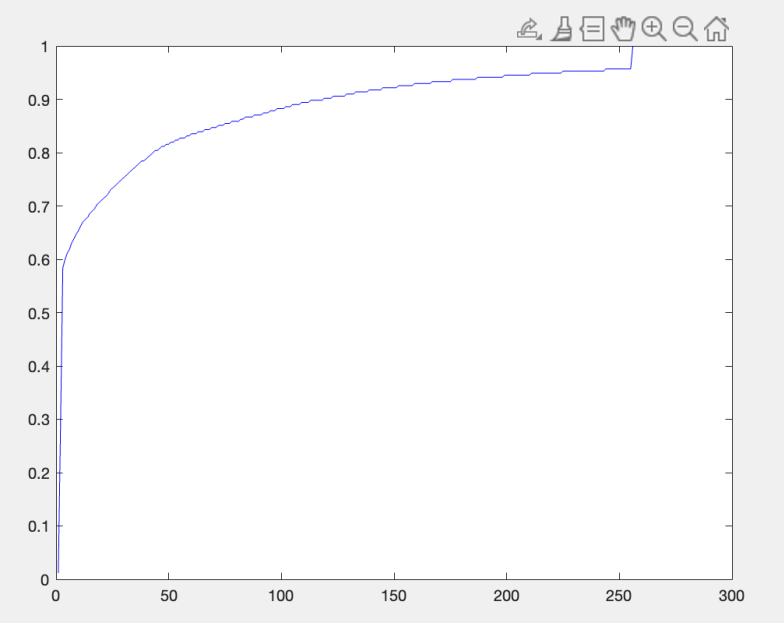

Figure 4. Plot of Histogram-equalization transformation function

Using the equation spreads the histogram of the input images so that the intensity levels of the equalized image span a wide range on the intensity scale. The stretch results in an enhancement in the image.

4.0 Conclusion

This project experimented with the log transformation and power-law transformation to enhance the image, creating a more visually interpretable image where more information can be seen as compared with its original. The histogram and the histogram equalization technique have also been implemented to the image, which in turn generated a better image.

References

[GW] R.C. Gonzalez, R.E. Woods, Digital Image Processing, 3rd Edition, ISBN: 978-0-168728- 8, 2008.

Figure 3.8 (a) Magnetic resonance image (MRI) of a fractured human spine

Appendix A

```
clc
close all
% import image
img_input = imread('Fig0308(a)(fractured_spine).tif');
whos img_input
imshow(img_input)
title('Original Input');
%Find max pixel value
max_matrix = max(img_input(:))
min_matrix = min(img_input(:))
calc_c = 1.0/log(1 + double(max(img_input(:))))
% log transformation
% Try changing the constant c
C = [1 0.2 0.3 calc_c];
pos = 1;
figure1 = figure;
subplot(1,length(C)+1, 1), imshow(img_input);
title('Original Input');
for i = 1:length(C)
   img_logtransform = C(i)*log(1+double(img_input));
   %tt= mat2gray(img_logtransform);
   subplot(1,length(C)+1,pos+1), imshow(img_logtransform,'Border','tight');
   img_label = sprintf('c = %.2f', C(i));
   title(img_label);
   pos = pos+1;
end
saveas(figure1,'Q1.jpg')
%imwrite (img_logtransform, 'LogTransform.png');
% Power-law transformation (gamma > 1)
gamma = [1.0 2.0 3.0 5.0];
pos = 1;
figure_pow = figure;
%Plot original img
subplot(1,length(gamma)+1,1), imshow(img_input);
title('Original');
for i = 1:length(gamma)
   pos = pos+1;
   img_powertrans = im2double(img_input).^gamma(i);
   subplot(1,length(gamma)+1,pos), imshow(img_powertrans,'Border','tight');
   img_label = sprintf('gamma = %.1f', gamma(i));
   title(img_label);
end
saveas(figure1,'Q2.jpg')
% Power-law transformation (gamma < 1)
gamma = [0.1 0.2 0.3 0.5];
pos = 1;
```

```
figure_pow = figure;
%Plot original img
subplot(1,length(gamma)+1,1), imshow(img_input);
title('Original');
for i = 1:length(gamma)
   pos = pos+1;
   img_powertrans = im2double(img_input).^gamma(i);
   subplot(1,length(gamma)+1,pos), imshow(img_powertrans,'Border','tight');
   img_label = sprintf('gamma = %.1f', gamma(i));
   title(img_label);
end
saveas(figure1,'Q2.jpg')
%imwrite (img_logtransform, 'LogTransform.png');
%histogram by Matlab Program
figure;
subplot(2,2,1), imshow(img_input);
title('Original Image');
subplot(2,2,2), imhist(img_input);
title('Histogram');
%histogram equlization
hist_eq = histeq(img_input,256);
subplot(2,2,3), imshow(hist_eq)
title('Histogram Equalization');
subplot(2,2,4), imhist(hist_eq);
title('Histogram Equalization');
figure;
[histIm,T] = histeq(img_input,256);
plot(T,'b-')
%Implement Historgram
figure;
create_histogram(img_input)
%title('Test Histogram');
[pixelCounts, grayLevels] = imhist(img_input);
[r,c] = size(img_input);
cdf = cumsum(r*c);
cdf = cdf / sum(cdf); % Normalize
plot(grayLevels, cdf, 'b-');
hold on
%-----------
%function create_histogram to return histogram my_hist
function my_hist = create_histogram(img)
   min_mat = min(img); %min pixel value in image
   max_mat = max(img); %max pixel value in image
   [row,col] = size(img);

   range = (max_mat - min_mat)+1;
   x_pixval = 1:1:255;
   y_pixcount = (zeros(1,255)); %count array
   lvl = 1;
```

```
    while (lvl < 256)
        count = 0;
        for i = 1 : row
            for j = 1 : col
                if img(i,j) == lvl
                    %lvl
                    count = count + 1;
                    %y_pixcount(int(x_pixval(lvl))) = y_pixcount(x_pixval(lvl))+1;
                end
            end
        end
        y_pixcount(1, lvl) = count;
        lvl = lvl+1;
    end
    bar(x_pixval, y_pixcount)
    title('Implemented Histogram');
       grid on;
    xlim([0 255]); %Set the x axis range manually to be 0-255.
end
```