The Report of MP3

I. Basic Concepts

1.1 Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. Thus, in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

It is very useful for scientific images like thermal, satellite or x-ray images, often the same class of images to which one would apply false-color. Also, histogram equalization can produce undesirable effects (like visible image gradient) when applied to images with low color depth. For example, if applied to 8-bit image displayed with 8-bit gray-scale palette it will further reduce color depth (number of unique shades of gray) of the image.

1.2 Algorithm and Implementation

In a discrete grayscale image {X}, define n as the total number of pixels and n_i as the number of occurrences of gray level i. Then, the probability of an occurrence of a pixel of level i in the image is

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \ \ 0 \leq i < L$$

where L is the total number of gray levels in the image (256 for 8-bit image). Next, we define the cumulative distribution function corresponding to $p_x(i)$ as

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

which is also the image's accumulated normalized histogram.

We would like to create a transformation of the form y = T(x) to produce a new image {Y}, with a flat histogram. Such an image would have a linearized cumulative distribution function (CDF) across the value range,

$$cdf_y(i) = \ iK$$

The properties of the CDF allow us to perform such a transform as following.

$$cdf_y(Y') = \ cdf_y(T(K)) = cdf_x(k)$$

In order to map the values back into their original range, it should time the length of the image back to the original range.
However, the result might be imperfect. That is the lighting of the image could be uneven and some parts might be brighter than others. Here comes out a method named of lighting correction, which is equal to tilting the image so that it would be equal-lighting. The implementation of such method is to find a fitting plane of curve for the existing intensity of the image. There are two ways to implement such method. One is linear, and the other is quadratic.
For linear,

$$\begin{bmatrix} r_1 & c_1 & 1 \\ r_2 & c_2 & 1 \\ & \dots & \\ r_N & c_N & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} I_1(r_1 & c_1) \\ I_2(r_2 & c_2) \\ & \dots \\ I_N(r_N & c_N) \end{bmatrix}$$

Through the operational formula of the matrix, we could get the fitting plane. After compensation, we could get the final image.
For quadratic,

$$\begin{bmatrix} r_1^2 & r_1c_1 & c_1^2 & r_1 & c_1 & 1 \\ r_2^2 & r_2c_2 & c_2^2 & r_2 & c_2 & 1 \\ & & \dots & & & \\ r_N^2 & r_Nc_N & c_N^2 & r_N & c_N & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} I_1(r_1 & c_1) \\ I_2(r_2 & c_2) \\ & \dots \\ I_N(r_N & c_N) \end{bmatrix}$$

Through the operational formula of the matrix, we could get the fitting plane. After compensation, we could get the final image.

1.3 The purpose of the functions

The function in the MP3 is defined as

Img_out = HistoEqualization (img_in)

where the img_in and img_out are the original image and the final image after histogram processing. After the histogram processing, we could get a better contrast adjustment.

II. Results and Analysis

2.1 Test Results

After running the code and test the image 'moon.bmp', there comes the results as followed.
The initial image is shown as Figure 1.

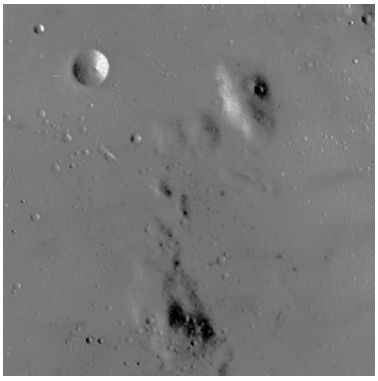

Figure 1 Initial image of 'moon,bmp'

After Histogram Equalization, the tackled image is shown as Figure 2.

Figure 2 Image of 'moon,bmp' after Histogram Equalization

When applying the linear lighting correction method, the final result is shown as Figure 3. And the Figure 4 is the fitting plane.

Figure 3 Image of 'moon,bmp' after linear lighting correction

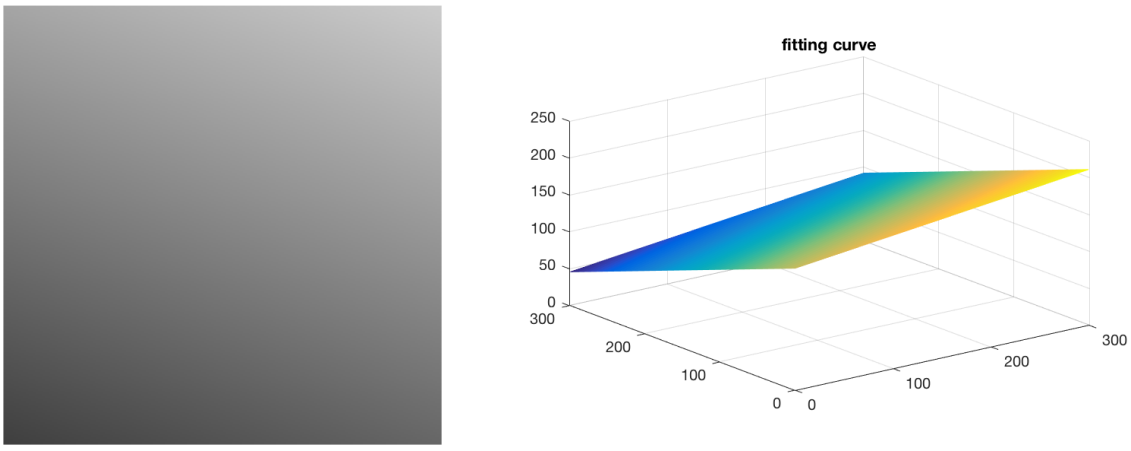

Figure 4 Fitting Plane of linear correction

When applying the quadratic lighting correction method, the final result is shown as Figure 5. And the Figure 6 is the fitting curve. The fitting curve is drawn from a tool online, because there

is something wrong with my matlab when drawing such curve. I cannot output the right fitting curve, thus I had to draw it online.

Figure 5 Image of 'moon,bmp' after quadratic lighting correction

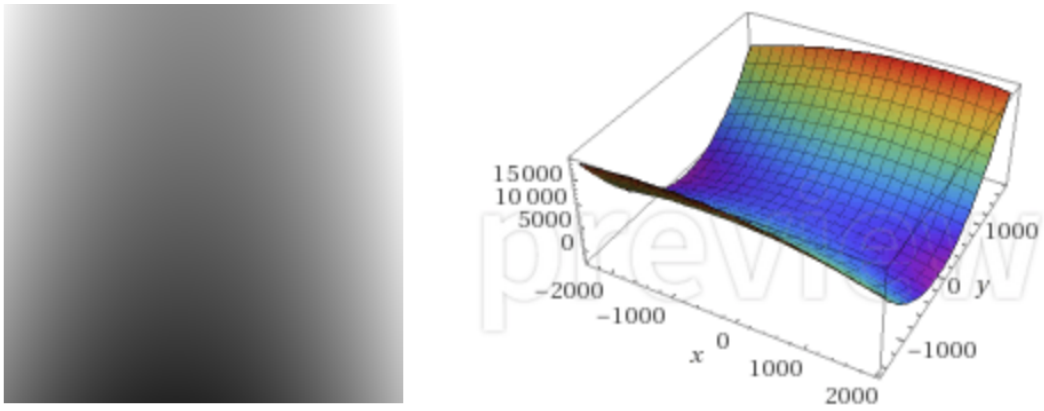

Figure 6 Fitting Curve of quadratic correction

2.2 Result Analysis

From the above results, it could be safely drawn a conclusion that both histogram equalization function and two lighting correction methods are implemented successfully. In addition, it could also prove that lighting correction methods could compensate the brightness of the image in a way.

2.3 Summary

From the MP3, I get a better understanding of the core of Histogram Equalization and two ways to correct the lighting of the image. From the testing results, it is clear to show the effectiveness of the function.

III. Matlab Codes

```
function img_out = HistoEqualization(img_in)
%implement a function for Histogram Equalization, imput of which is the
%input image and output is the tackled image.
Image = rgb2gray(imread(img_in,'bmp'));
```

```
[r , c] = size(Image);
n = r * c;
output = uint8(zeros(r , c)); %It has to be int8
f = zeros(256 , 1); %count the number of pixels in the image with the same intensity
PDF = zeros(256 , 1);%vector for probability density function
CDF = zeros(256 , 1);%vector for cumulative distribution function
Final = zeros(256 , 1);%vector for rounding the final intensity of each pixel

%loop for probability density function
for i = 1 : r
    for j = 1 : c
        value = Image(i , j);
        f(value + 1) = f(value +1) + 1;
        PDF(value + 1) = f(value + 1) / n;
    end
end

%loop for cumulative distribution function
L = 255; %it has to be 255 because the range of the 8-bit image is 0-255
CDF(1) = PDF(1);
for i = 2 : size(PDF)
    CDF(i) = CDF(i-1) + PDF(i);
    Final(i) = round(CDF(i) * L);
end

%output the histogram equalized image
for i = 1 : r
    for j = 1 : c
        output(i,j) = Final(Image(i,j) + 1);
    end
end

%img_out = imshow(output); %The output of HistoEqualization without
                           %Lighting Correcting,enable it when testing for
                           %Histogram Equalization

%Lighting Correction using linear method & quadratic method
X = ones(256*256,3); %linear data vector
U = ones(256*256,6); %quadratic data vector
Y = zeros(256*256,1); %vector for storing the intensity of image
k = 1;
for i = 1 : r
    for j = 1 : c
        X(k,1) = i;
        X(k,2) = j;

        U(k,1) = i*i;
        U(k,2) = i*j;
        U(k,3) = j*j;
        U(k,4) = i;
        U(k,5) = j;

        Y(k,1) = output(i,j);
        k = k + 1;
    end
end

a = (X'*X)^-1 * X' * Y; %parameter vector for fitting plane
b = (U'*U)^-1 * U' * Y; %parameter vector for fitting curve
Y1 = X * a;
Y2 = U * b;
Y_line = (Y + Y1)/2; %compensate for the original intensity in the linear method
Y_quad = (Y + Y2)/2; %compensate for the original intensity in the quadratic method

%output the final image after lighting correction
l = 1;
for i = 1:r
    for j = 1:c
        output(i,j) = Y_quad(l,1);% testing for linear, you should change it to Y_line
        l = l + 1;
    end
end

%plot the fitting plane or curve
[x,y] = meshgrid(1:1:300);
%z = a(1,1) * y + a(2,1) * x + a(3,1);
z = b(1,1) * x^2 + b(2,1) * x*y + b(3,1) * y^2 + b(4,1) * x + b(5,1) * y + b(6,1);
mesh(x,y,z);
```

```
title('fitting curve');
img_out = imshow(output);
end
```

Testing command: >> a = HistoEqualization('moon.bmp');