Histogram Equalization

Histogram equalization is a technique for adjusting image intensities to enhance contrast.

Let f be a given image represented as a m_r by m_c matrix of integer pixel intensities ranging from 0 to L − 1. L is the number of possible intensity values, often 256. Let p denote the normalized histogram of f with a bin for each possible intensity. So

$$p_n = \frac{\text{number of pixels with intensity n}}{\text{total number of pixels}} \qquad \text{n = 0,1,…,L-1}$$

The histogram equalized image g will be denoted by

$$g_{i,j} = floor((L-1)\sum_{n=0}^{f_{i,j}} p_n)$$

where floor() rounds down to the nearest integer. This is equivalent to transforming the pixel intensities, k, of f by the function

$$T(k) = floor((L-1)\sum_{n=0}^{k} p_n)$$

This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity 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. First We have to calculate the frequencies of intensity at specific pixels. By dividing each frequency with total value of intensity from all the pixels, we get the pdf function. By adding the pdfs in a cumulative manner, we get the cdf function. To normalize the cdf, we multiply each cdf with L-1 where L is the range of the grayscale image which is 256. The intensities of the original image are then mapped to the new intensity value from the normalized cdf and output in a new image.

Implementation code in matlab

```
img = imread('HE.tif');

row = size(img,1);

col = size(img,2);

numberOfPixel = row*col;

figure, imshow(img);

title('Original Image');

% figure,imhist(rgb2gray(img));

% title('Original Histogram');
```

```
img2Histogram = uint8(zeros(row,col));
frequency = zeros(256,1);
%probabilityFunction = zeros(256,1);
probabilityCumulative = zeros(256,1);
cumulative = zeros(256,1);
output = zeros(256,1);
for i = 1:row
   for j = 1:col
      value = img(i,j);
      frequency(value+1) = frequency(value+1)+1;
      %probabilityFunction(value+1) = frequency(value+1)/numberOfPixel;
   end
end
sum = 0;
for i = 1:size(frequency);
   sum = sum+frequency(i);
   cumulative(i) = sum;
   probabilityCumulative(i) = cumulative(i)/numberOfPixel;
   output(i) = round(probabilityCumulative(i)*255);
end
for i = 1:row
   for j = 1:col
      img2Histogram(i,j) = output(img(i,j)+1);
   end
end
he = histeq(img);
figure,imshow(img2Histogram);
title('Edited Image');
% figure,imhist(img2Histogram);
```

```
% title('Edited Histogram');
figure,imshow(he);
title('BuiltIn Function');
```

Figure 1: Original Image

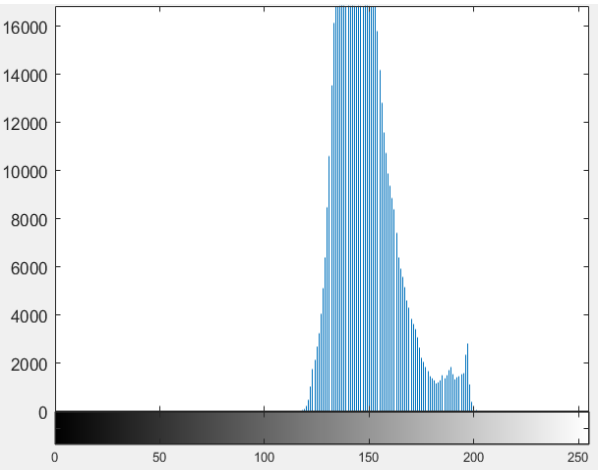

Figure 2: Original Histogram

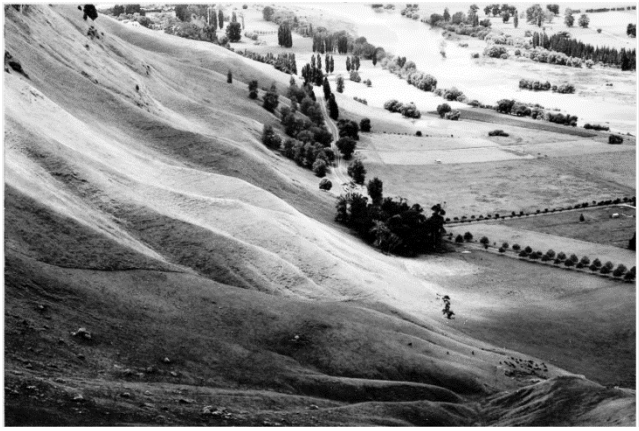

Figure 3: Edited Image after Histogram Equalization

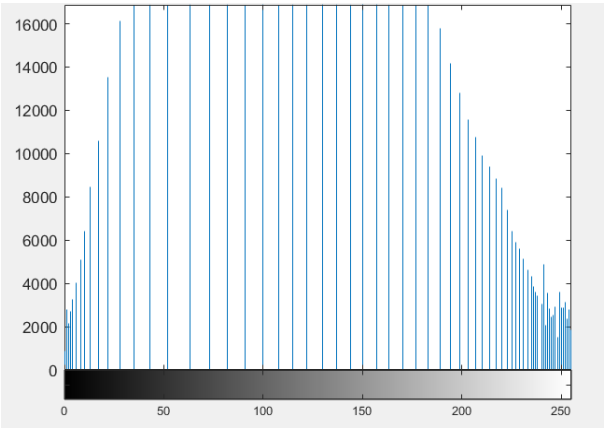

Figure 4: Histogram of edited Image

Averaging Filter(Box Filter)

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called averaging filters. By replacing the value of every pixel in an image by the average of the intensity levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in intensities. Because random noise typically consists of sharp transitions in intensity levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp intensity transitions so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number of intensity levels. A major use of averaging filters is in the reduction of "irrelevant" detail in an image. By "irrelevant" we mean pixel regions that are small with respect to the size of the filter mask. This is what a 3x3 average filter computes at each pixel.

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

1/9 x

1	1	1
1	1	1
1	1	1

Implementation code in matlab

```
a=imread('image1.jpg');

figure, imshow(a)

%an=imnoise(a,'gaussian');

[m,n]=size(a);

%figure, imshow(uint8(an))

b=input('Enter Averaging Mask size: ');

z=ones(b);

[p,q]=size(z);

w=1:p;

x=round(median(w));

anz=(zeros(m+2*(x-1),n+2*(x-1)));

for i=x:(m+(x-1))
   for j=x:(n+(x-1))
      anz(i,j)=a(i-(x-1),j-(x-1));
      %anz(i,j)=an(i-(x-1),j-(x-1));
```

```
    end
end
% figure, imshow(uint8(anz))
sum=0;
x=0;
y=0;
for i=1:m
    for j=1:n
        for k=1:p
            for l=1:q
                sum= sum+anz(i+x,j+y)*z(k,l);
                y=y+1;
            end
            y=0;
            x=x+1;
        end
        x=0;
        ans(i,j)=uint8((1/(p*q))*(sum));
        sum=0;
    end
end
figure, imshow(uint8(ans))
```

Figure 5: Original Image.

Figure 6: Result after applying Box Filter.

Laplacian Mask

Laplacian is a second order derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian simply by adding the Laplacian image to the original. Laplacian operator gives good result in finding the fine details of the image. It increases the sharpness of the image.

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y)$$

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

0	1	0
1	-4	1
0	1	0

When considering the 4 neighbors only.

Implementation code in matlab

```
clc

clear all

close all

image=imread('LaplacianImg.tif');

filter=[1 1 1 ; 1 -8 1 ; 1 1 1];

rows = size(image,1);

cols = size(image,2);

outputimage = zeros(rows,cols);


for row = 2:rows-1
   for col = 2:cols-1
      for i=1:3
      outputimage (row,col)= sum (sum(double(image(row-1: row+1, col-1: col+1)) .* filter ));
      end
   end
end
```

```
%image =uint8(image);
figure,imshow (outputimage)
image1=imsubtract(image,uint8(outputimage));
figure,imshow(image)
figure,imshow(image1)
```

Figure 7: Original Image.

Figure 8: After adding the mask with original image.

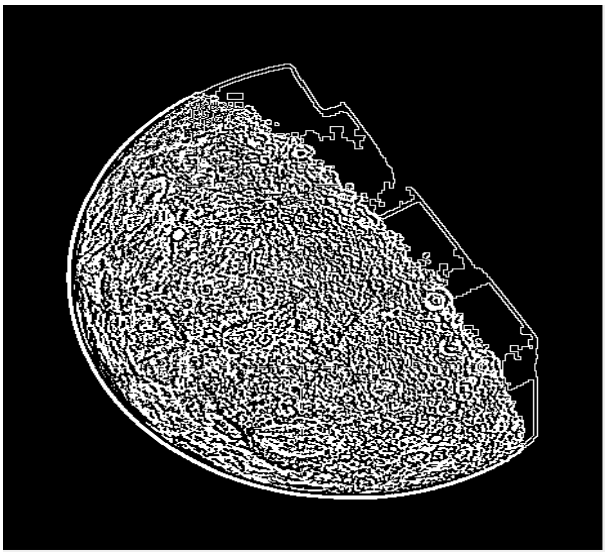

Figure 9: Laplacian mask.