Digital Image Processing Report

Histogram Equalization, Histogram Matching

Fahrul Firmansyah 20081010099

Muhammad Sabili Nurilhaq 20081010101

1. Histogram Equalization

Low contrast images can be enhanced by histogram equalization method. Histogram equalization is a technique used in image processing especially to enhance the contrast of an image. Here's how to implement histogram equalization technique in matlab:

1. Read an Image

```
img = imread("greyscale_img2.jpg");
[rows, cols] = size(img);
MN = rows * cols;
```

First of all, we need to prepare a grayscale image, so that we can read the image as you can see in the code above. Don't forget to also take the row and column size from that image and multiply it into a size of that image. Images we have prepared can be seen on Figure 1.1.

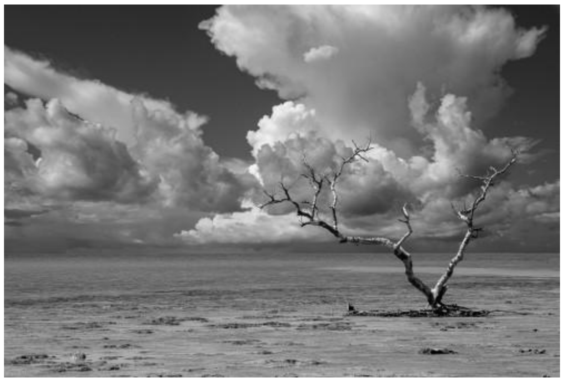

Figure 1.1. Input Image

2. Histogram Calculation

```
hist = imhist(img);
```

Second step of histogram equalization technique is histogram calculation. At this stage, we use the 'imhist' function to calculate the distribution of pixel intensity as you can see in the code above, so that it can generate the distribution value of pixel intensity easily.

3. Calculate PDF

```
PDF = hist / MN;
```

Next step, we calculate PDF (Probability Density Function) value using the code above. PDF value obtained from dividing the value of each histogram intensity by image size (MN). Division operations are performed between arrays (matrix) with scalar number (MN).

4. Calculate CDF

```
CDF = cumsum(PDF) * 256;
CDF = round(CDF);
```

The PDF value that has been obtained from the previous step will be used to get the CDF (Cumulative Distribution Function) value. CDF value obtained by summing each value cumulatively, which the nth value will be added with n-1 and n-1 will also be added with n-2 and so on until the value of n in the first index.The result of that calculation will be multiplied by the scalar value of 256 which is the gray level of the 8-bit image. In addition, we also round the CDF value obtained to get the value of the transformation result.

5. Creating the Enhanced Image

```
resultImg = uint8(rows:cols);
for i = 1 : rows
  for j = 1 : cols
    resultImg(i,j) = CDF(img(i,j) + 1);
  end
end
```

After getting the value of the transformation result, we try to transform each pixel value as you can see in the code above. Before transforming it, we need to create an image container with a width and height size according to the image that has been read, so we can put the transformed value into that container. After the image is created, we can try to display it to the console both the image results and the histogram of the image results. You can see a comparison of the results of the image enhancement and histogram respectively in Figure 1.2 and 1.3.

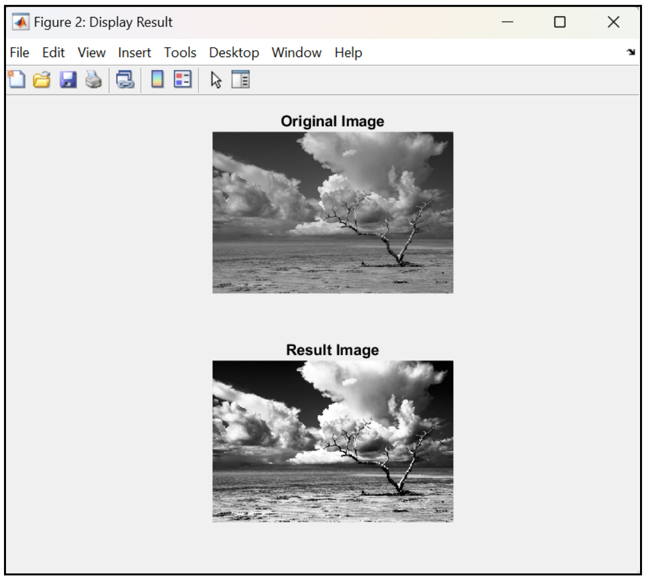

Figure 1.2. Image Comparison Results

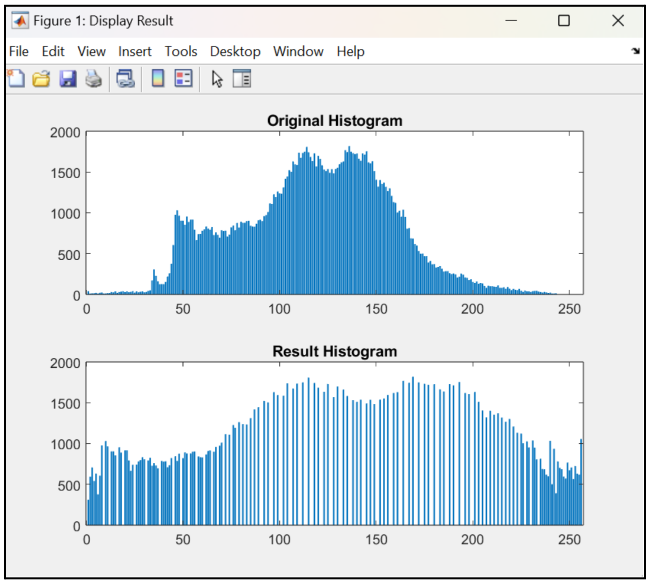

Figure 1.3. Histogram Comparison Results

Based on the results of the image and histogram that has been obtained as can be seen in Figure 1.2 and 1.3, it can be concluded that the goal of enhancing the image from low contrast to high contrast has been achieved. This is based on the results of the intensity distribution of pixel values on the histogram that looks flatter than the histogram of the original image whose distribution is focused on the middle.

2. Histogram Matching

Histogram is one of data visualization that is commonly used to visualize numerical data distribution. Using histogram we can know the intensity value distribution in our gray image. Through histogram we can also observe the contracity of an image, whereas high contrast images give us more information about the image. Image that can be called high contrast when the range of lowest intensity value and highest intensity value is big. We can also enhance our image contrast using another image that has higher contrast. These are the step to visualize pixel value distribution in matlab:

a. Read the Low Contrast Image

```
low=imread('low.jpg');
gray1=low(:,:,1);
```

The code above works as an image reader and saves the image as an array with 3 dimensional shape. After we read the image, we convert it to a gray image by taking only the red layer.

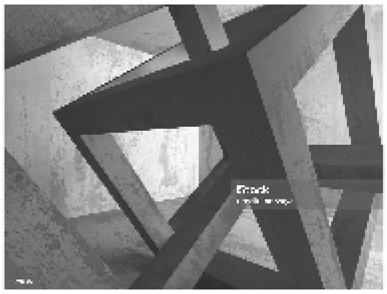
Figure 2.1. Low Contrast Image

b. Read the High Contrast Image

```
high=imread('high.jpg');
gray2=high(:,:,1);
```

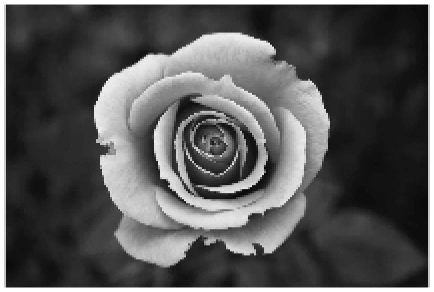

Figure 2.2. High Contrast Image

The code above works as an image reader and saves the image as an array with 3 dimensional shape. After we read the image, we convert it to a gray image by taking only the red layer.

c. Save the Intensity Distribution

```
hist1 = imhist(gray1);
hist2 = imhist(gray2);
```

The code above works as a container of the histogram. The histogram container will be used to calculate Cumulative Distribution Function. Cumulative Distribution Function (CDF) is the fraction of pixels in which a pixel value is less than or equal to the specified pixel value.

d. Calculate Cumulative Distribution

```
cdf1 = cumsum(hist1) / numel(gray1);
cdf2 = cumsum(hist2) / numel(gray2);
```

Cumsum function is used to calculate the sum of all the elements before it. And numel function is the total number of elements (pixels). By dividing the result of cumsum() by numel(), we obtain the cumulative distribution of pixel intensities in the image.

e. Pixel Intensity Distribution Matching

```
mapping = zeros(256,1,'uint8');
for idx = 1 : 256
    [~,ind] = min(abs(cdf1(idx) - cdf2));
    mapping(idx) = ind-1;
end
```

To enhance the contracity of the low contrast image, we have to map the distribution of the high contrast image intensity value with the distribution of the low contrast image intensity value. To perform that, we need to create a container for the result mapping that has shape 256, and we need to calculate the difference between CDF low contrast image and CDF high contrast image for each intensity, and store that value into the mapping results container variable.

f. Perform the Histogram Matching Transformation

```
[height, width] = size(gray1);
result=zeros(height,width,'uint8');
for x = 1: height
    for y = 1:width
         result(x,y) = mapping(gray1(x,y)+1);
    end
end
```

After we get the matching result distribution we can perform the histogram matching contrast enhancement. To do that, we only need to get the value in (x, y) position of the low contrast image, and search the new value in the mapping results container variable.

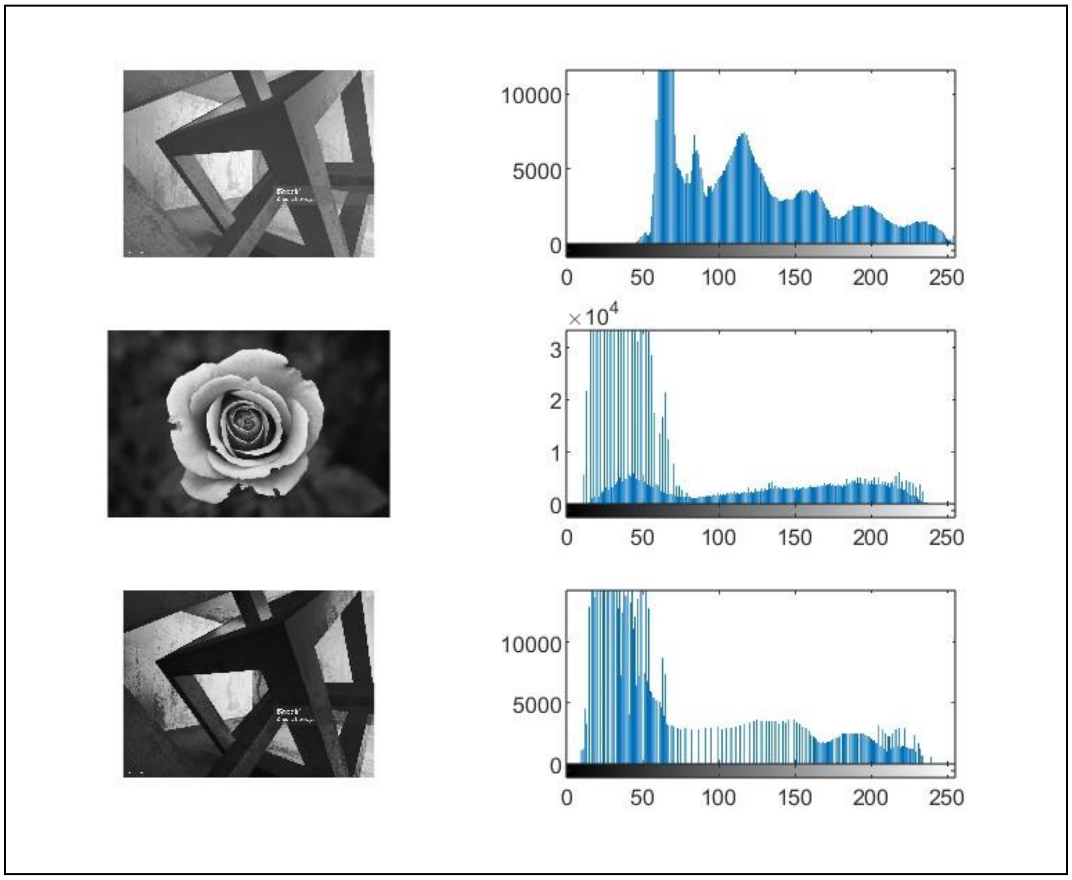

Figure 2.3. Result