INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DIGITAL IMAGE PROCESSING LABORATORY

A REPORT ON

EXPERIMENT 02

Histogram equalization and Matching

Name :	Parakh Agarwal	Anand Jhunjhunwala
Roll. No.:	17EC35016	17EC35032

01.02.2021

Group No. 2

DEPT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

VISUAL INFORMATION AND EMBEDDED SYSTEMS

Table of Contents

Objective:

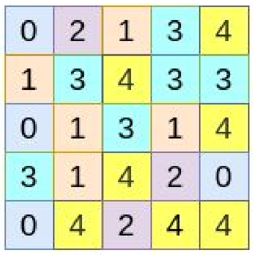
Write C/C++ modular functions to:
1) Perform histogram equalization on the given set of (gray and colored both) images.
2) Perform histogram matching of the input image with respect to Target image.

Introduction:

Histogram:

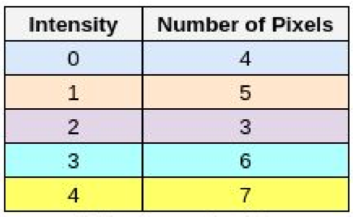
A histogram of an image is the graphical interpretation of the image's pixel intensity values. It can be interpreted as the data structure that stores the frequencies of all the pixel intensity levels in the image.

For example,
Suppose we have a sample 5*5 image with pixel diversities from 0 to 4. In the first step for generating the histogram, we create the Histogram Table, by counting the number of each pixel intensities. Then we can easily generate the histogram by creating a bar chart based on the histogram table. Se figure below



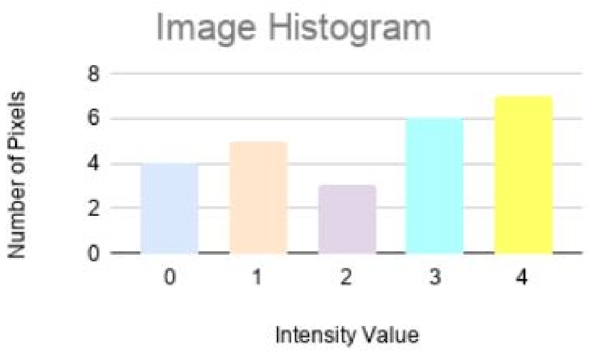
0	2	1	3	4
1	3	4	3	3
0	1	3	1	4
3	1	4	2	0
0	4	2	4	4

Image



Intensity	Number of Pixels
0	4
1	5
2	3
3	6
4	7

Histogram Table



The intensity level usually ranges from 0 to 255. For a gray-scale image, there is only one histogram, whereas an RGB colored image will have three 2-D histograms, one for each color.

Histogram Equalization:

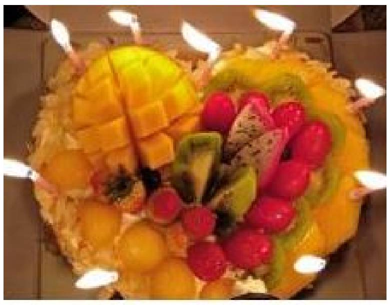
Histogram Equalization is an image processing technique that adjusts the contrast of an image by using its histogram. To enhance the image's contrast, it spreads out the most frequent pixel intensity values or stretches out the intensity range of the image. By accomplishing this, histogram equalization allows the image's areas with lower contrast to gain a higher contrast. In other words image histogram is transformed so that it matches with that of uniformly distributed histogram.

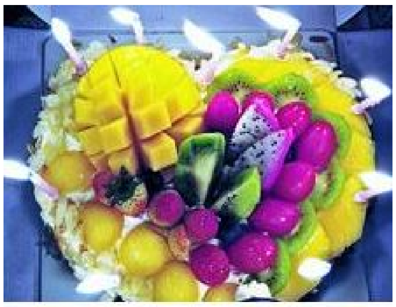
Histogram Equalization can be used when you have images that look washed out because they do not have sufficient contrast. In such photographs, the light and dark areas blend together creating a flatter image that lacks highlights and shadows.

See algorithm in algorithm section.

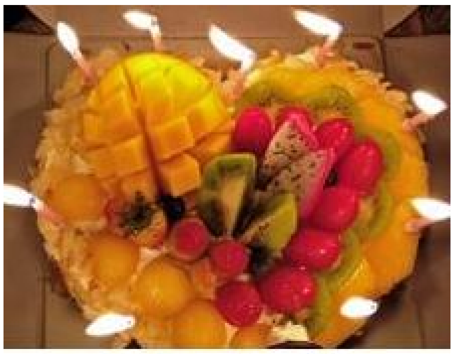
Note: For grayscale image, since we have only one channel (of intensity values), so we can apply histogram equalization w.r.t this and get our final histogram equalized image however in case of colour image we have 3 channels so one question may arise that which channel should be used to perform histogram equalization?. Several methods has been proposed, some of them are:

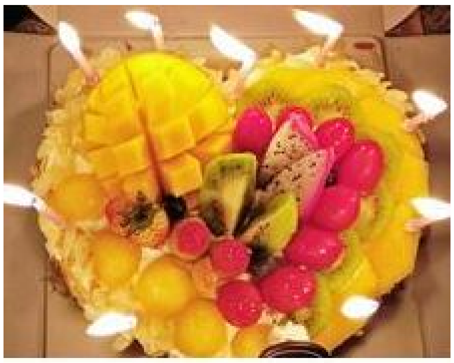
1) Apply histogram equalization separately to the Red, Green and Blue channels of the RGB color values of the image and rebuild an RGB image from the three processed channels. However, as we know that each colour has a different level of contribution in the intensity value so applying histogram equalization to each channel separately will create a heavy colour imbalance and will distort the image.


A: original rgb image


B: equalized by independent channel

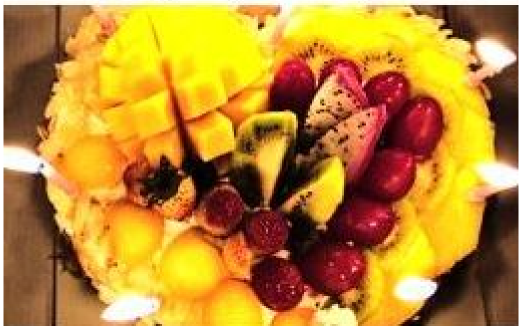
2) Use an average value of 3 channels and use that for histogram equalization. This method works better than the previous one however it still uses equal contribution of colour channels and results in spread of 3 channels while keeping their relative distribution intact.


A: original rgb image

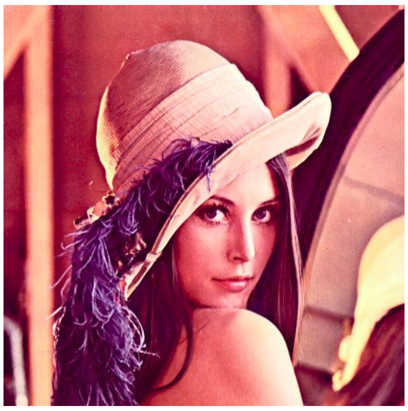

C: equalized by average histogram

3) Use of Intensity(I) channel in HSI color space after converting RGB to HSI space. In this method we first convert the RGB image to HSI space then perform histogram equalization only on the I channel keeping the values of H and S channel intact. This method performs better than the above two generally however applying equalization to the luminance increases contrast but not adjusting the hue and saturation weaks color in this case.

A: original rgb image↵

D: Intensity component equalization↵


4) Use of Y channel in YCbCr color space after converting RGB to YCbCr space. In this method we first convert the RGB image to YCbCr space then perform histogram equalization only on the Y channel keeping the values of Cb and Cr channel intact.



Histogram Matching:

In image processing, histogram matching or histogram specification is the transformation of an image so that its histogram matches a specified histogram. The well-known histogram equalization method is a special case in which the specified histogram is uniformly distributed. In simple words we modify one image based on the contrast of another one.

Histogram matching is useful when we want to unify the contrast level of a group of images. In fact, Histogram equalization can also be taken as histogram matching. For performing histogram matching we first first equalize the histogram of both images. Then, we need to map each pixel of the input image to the target image using the equalized histograms. Detailed algorithm is shown in the algorithm section.

Note: Similar technique can be used for colour image matching as was used in histogram equalization.

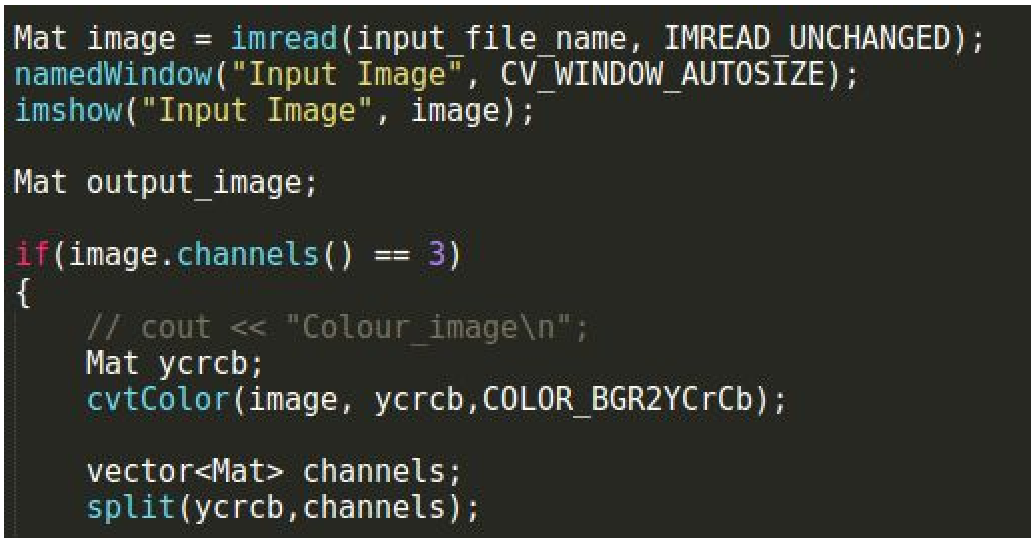
Algorithm:

In this section we tried to explain our implementation of the different algorithms used in this experiment.

Note: We have used Opencv only for reading and writing of images and for converting it from one space to another and both algorithms are implemented from scratch.

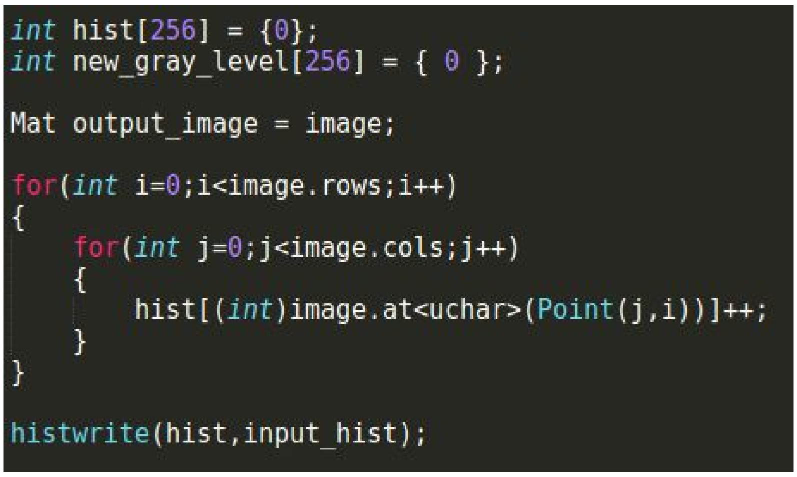
1. **Histogram Equalisation:**

   Step1: We first read the image and for colour image, we first convert it to YCbCr colour space and then use Y channel to perform histogram equalization, however for gray scale image we use the image as it is.



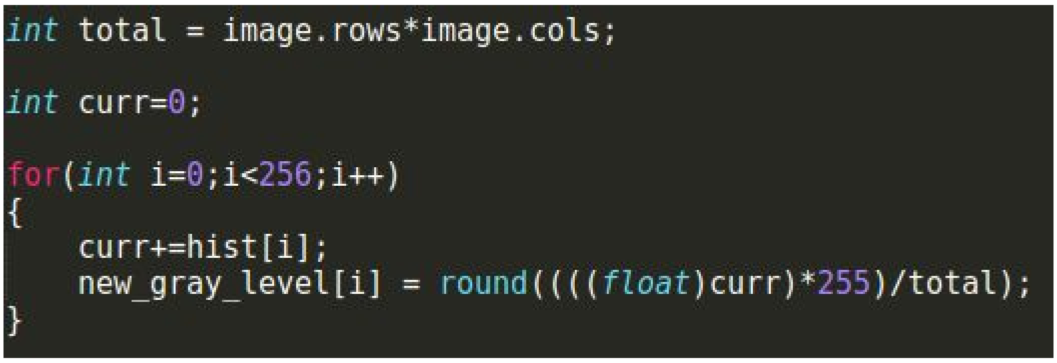
```
Mat image = imread(input_file_name, IMREAD_UNCHANGED);
namedWindow("Input Image", CV_WINDOW_AUTOSIZE);
imshow("Input Image", image);

Mat output_image;

if(image.channels() == 3)
{
    // cout << "Colour_image\n";
    Mat ycrcb;
    cvtColor(image, ycrcb,COLOR_BGR2YCrCb);

    vector<Mat> channels;
    split(ycrcb,channels);
```

Step2: We then find the histogram of the given image and save it, here histwrite function form the histogram image and saves it.



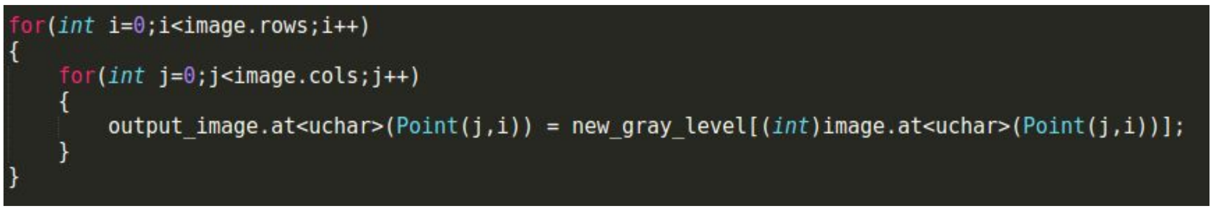
```
int hist[256] = {0};
int new_gray_level[256] = { 0 };

Mat output_image = image;

for(int i=0;i<image.rows;i++)
{
    for(int j=0;j<image.cols;j++)
    {
        hist[(int)image.at<uchar>(Point(j,i))]++;
    }
}

histwrite(hist,input_hist);
```

Step3: We then find the cumulative normalized histogram for each gray level (This is our transfer function) and then use this cumulative histogram and find the new gray value that each previous gray value has to map to by multiplying its corresponding cumulative value by 255. Here new_gray_level stores new gray value for each previous gray value.



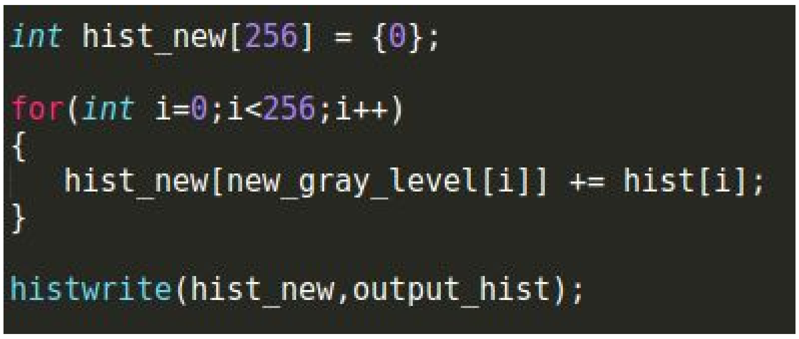
```
int total = image.rows*image.cols;

int curr=0;

for(int i=0;i<256;i++)
{
    curr+=hist[i];
    new_gray_level[i] = round((((float)curr)*255)/total);
}
```

Step4: We replace each previous gray value with its new gray level value to form the final output image.



```
for(int i=0;i<image.rows;i++)
{
    for(int j=0;j<image.cols;j++)
    {
        output_image.at<uchar>(Point(j,i)) = new_gray_level[(int)image.at<uchar>(Point(j,i))];
    }
}
```

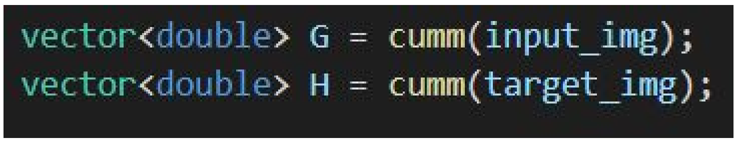
Step5: At last we calculate the new equalized histogram and save it to the system.



```
int hist_new[256] = {0};

for(int i=0;i<256;i++)
{
    hist_new[new_gray_level[i]] += hist[i];
}

histwrite(hist_new,output_hist);
```

2. Histogram Matching:

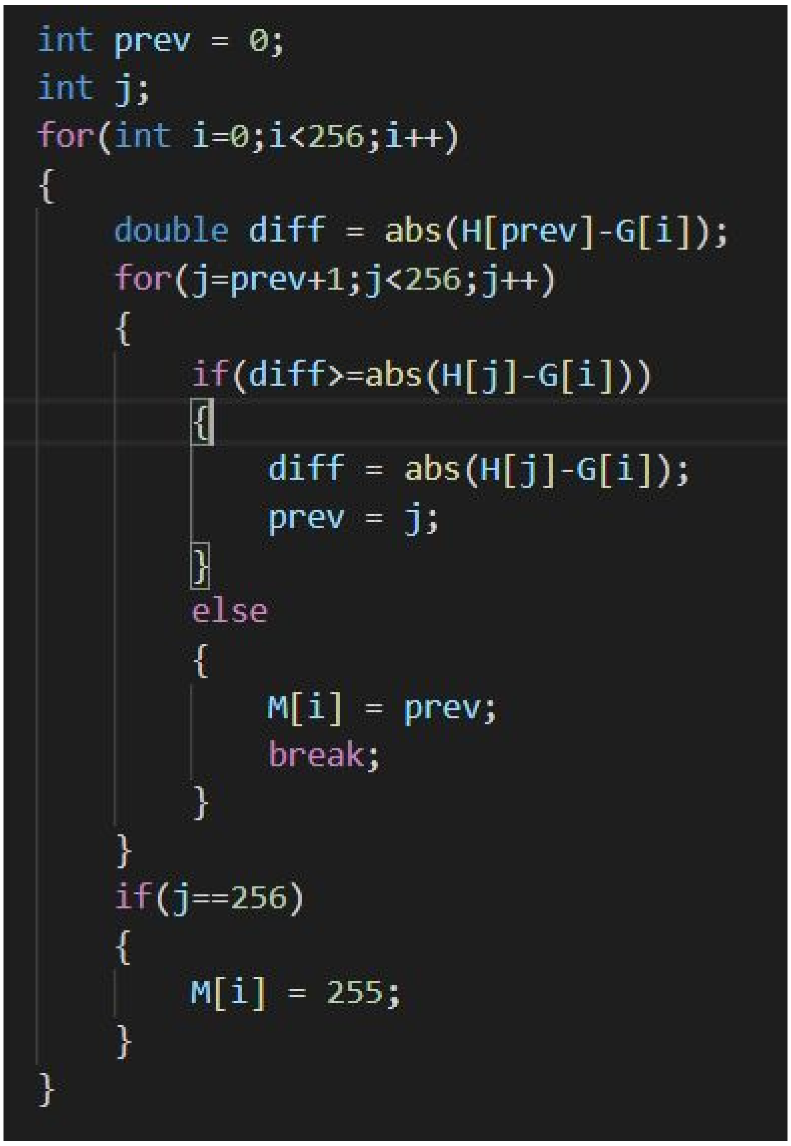
1. The initial steps for histogram matching involves a similar process as histogram equalisation. We first have a normalised cumulative histogram using the above mentioned steps for each target and reference image, which we mention as H and G respectively.



```
vector<double> G = cumm(input_img);
vector<double> H = cumm(target_img);
```

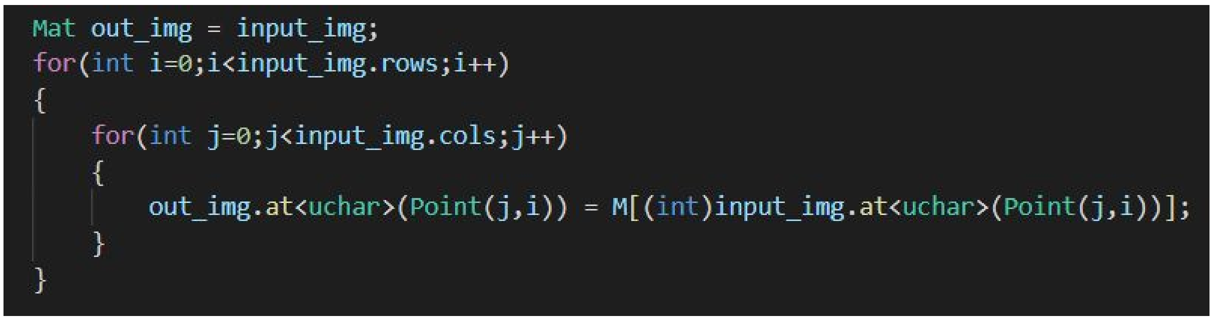
2. The objective is to find a function M to match G and H i.e., G(x1) = H(x2) such that M(x1) = x2. Then the function M is applied on every pixel of the target image. Below shown is the code to find the function M that gives the relation between normalised

value of cumulative distribution. We try to find the nearest values in the normalised distribution for each, and M stores the matched indexes.



```
int prev = 0;
int j;
for(int i=0;i<256;i++)
{
    double diff = abs(H[prev]-G[i]);
    for(j=prev+1;j<256;j++)
    {
        if(diff>=abs(H[j]-G[i]))
        {
            diff = abs(H[j]-G[i]);
            prev = j;
        }
        else
        {
            M[i] = prev;
            break;
        }
    }
    if(j==256)
    {
        M[i] = 255;
    }
}
```

3. Finally we combine the channels to obtain the final image where the original intensity of the target image is replaced by that of the reference image.

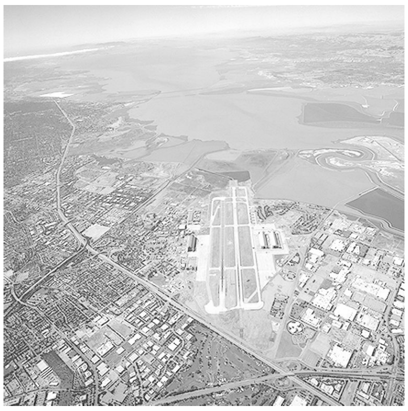


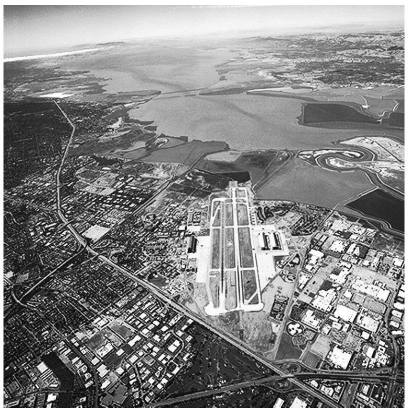
```
Mat out_img = input_img;
for(int i=0;i<input_img.rows;i++)
{
    for(int j=0;j<input_img.cols;j++)
    {
        out_img.at<uchar>(Point(j,i)) = M[(int)input_img.at<uchar>(Point(j,i))];
    }
}
```

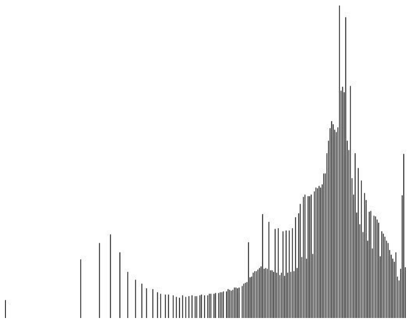
4. The histwrite function too is similar to the above and is used for the same purpose.

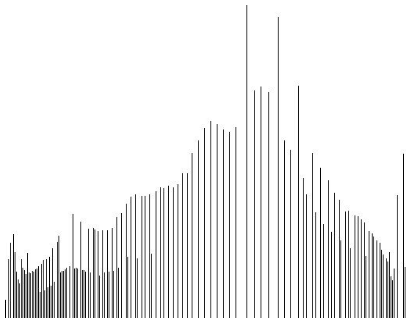
Results:

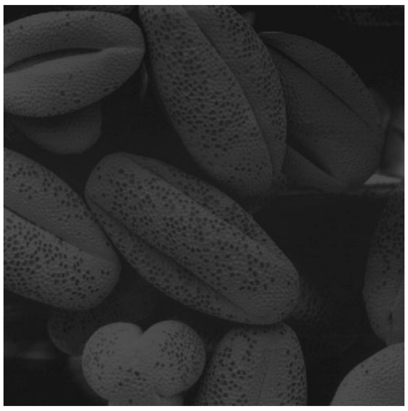
For Histogram Equalization:

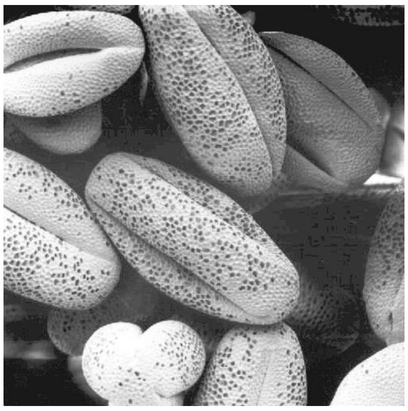

Input Image

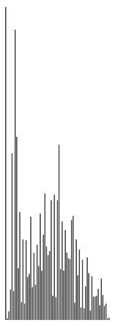

Histogram Equalized image

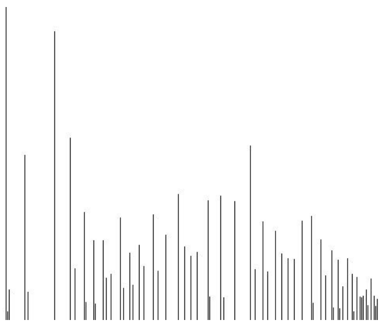

Input histogram

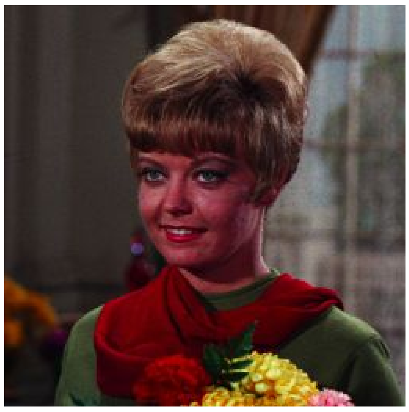

Equalized histogram

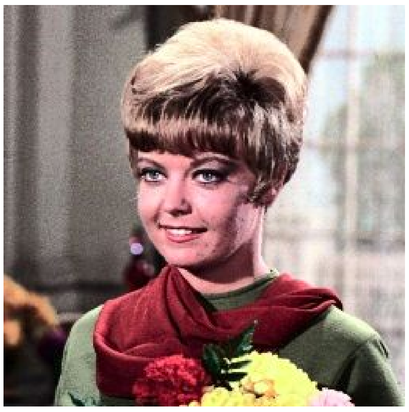

Input Image

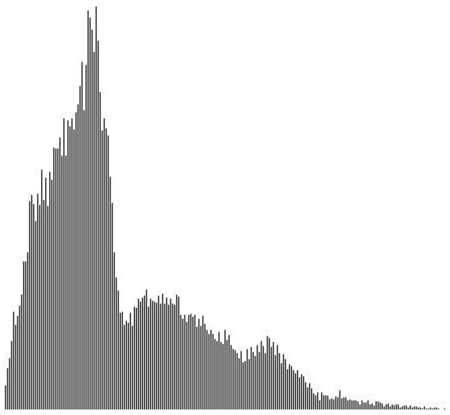

Histogram Equalized image

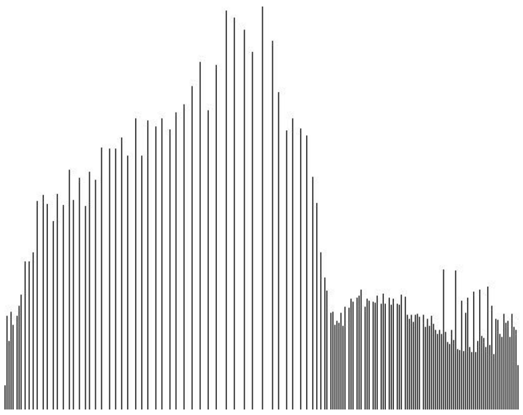

Input histogram

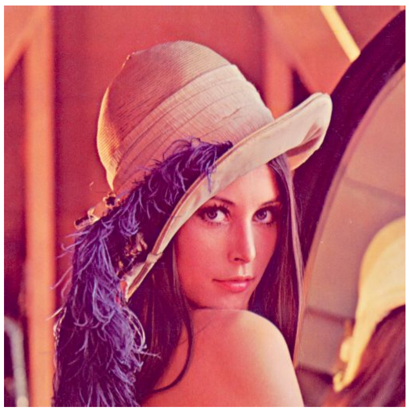

Equalized histogram

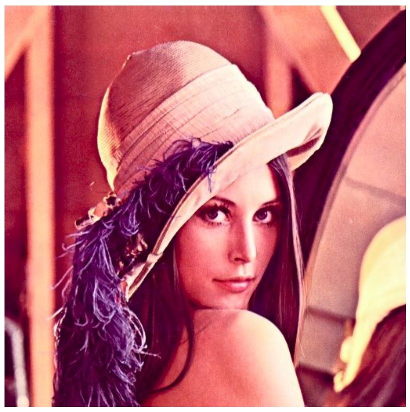

Input Image

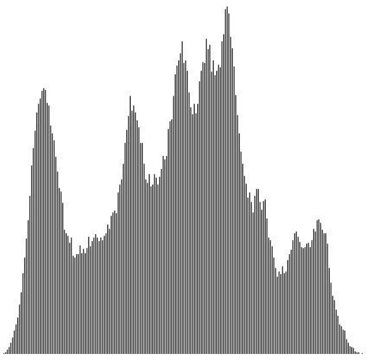

Histogram Equalized image

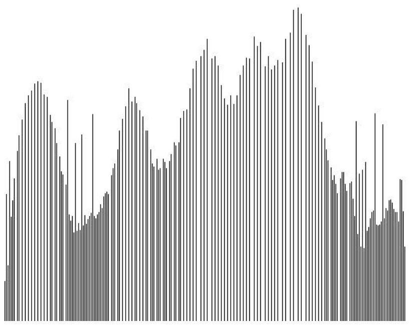

Input histogram


Equalized histogram

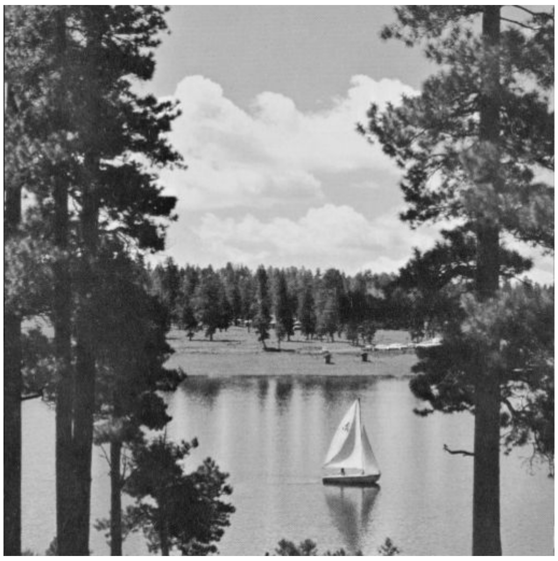

Input Image

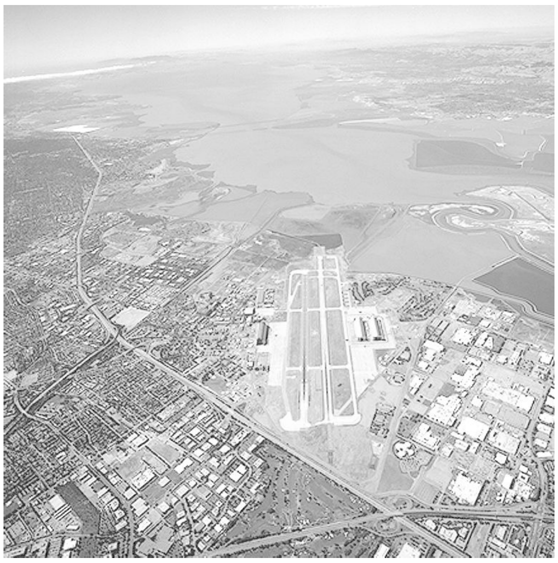

Histogram Equalized image

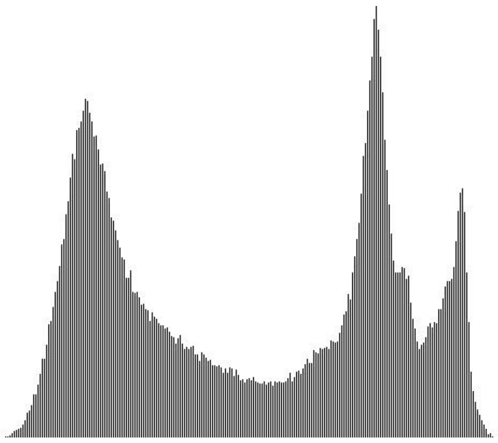

Input histogram

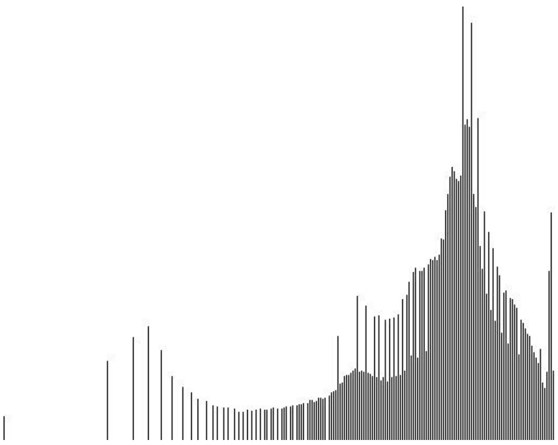

Equalized histogram

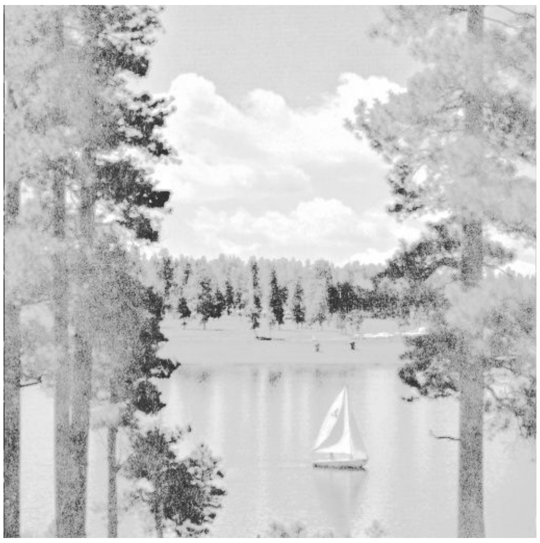
For Histogram Matching:

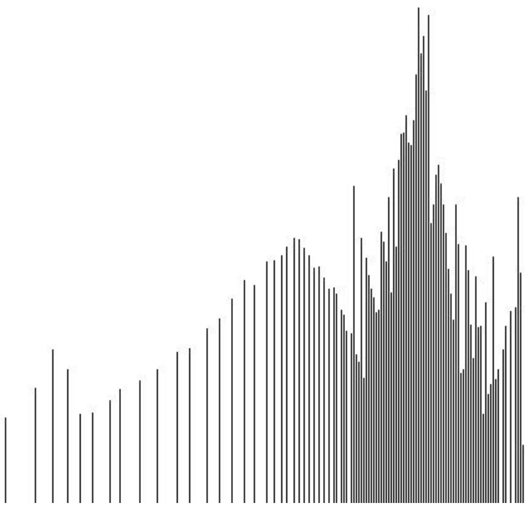

Input Image

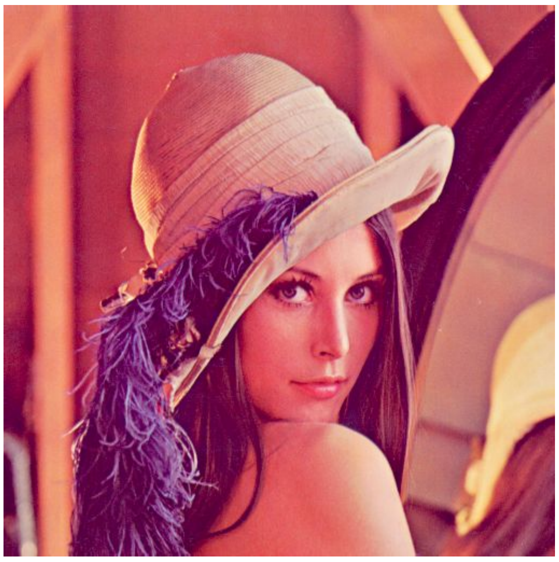

Target Image

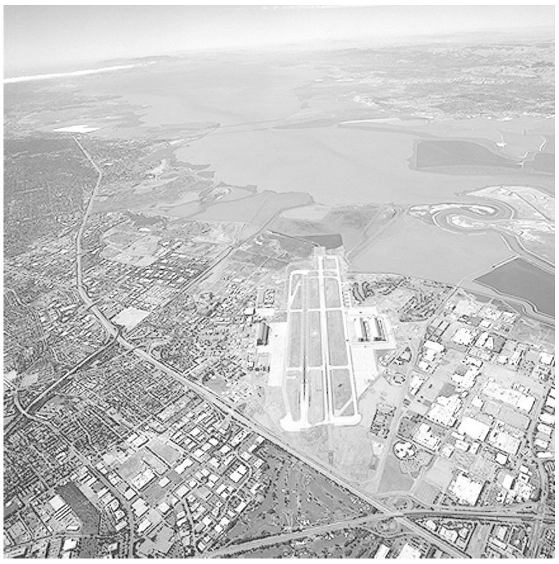

Input Histogram

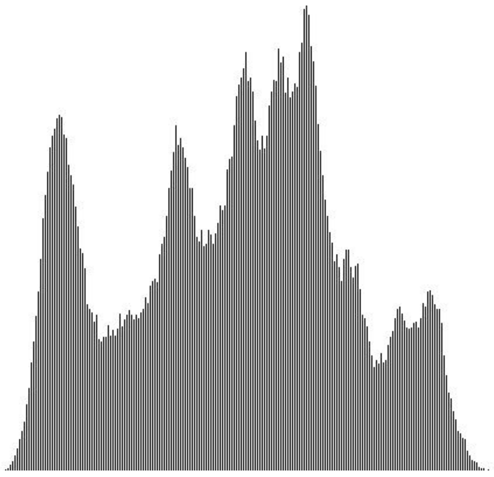

Target Histogram

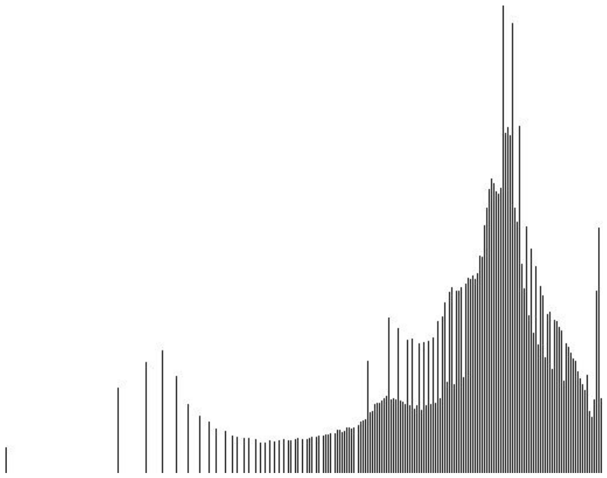

Histogram matched image

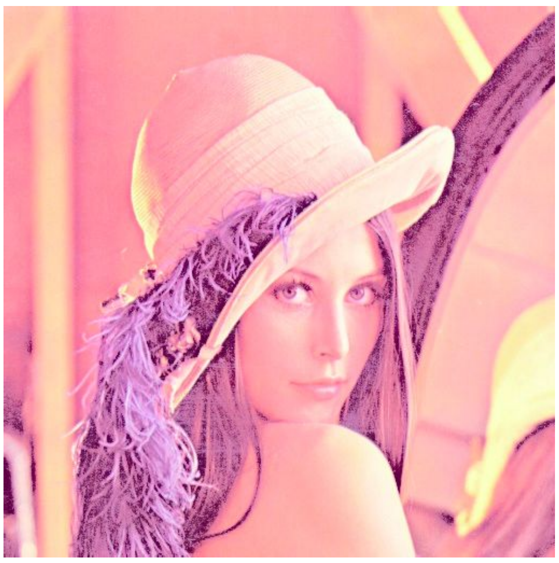

Matched Histogram

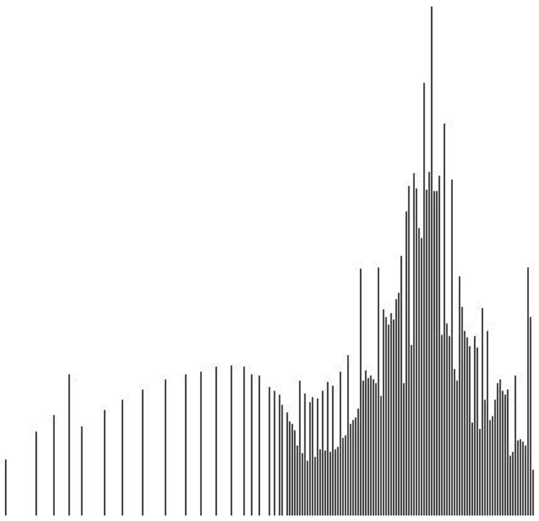

Input Image

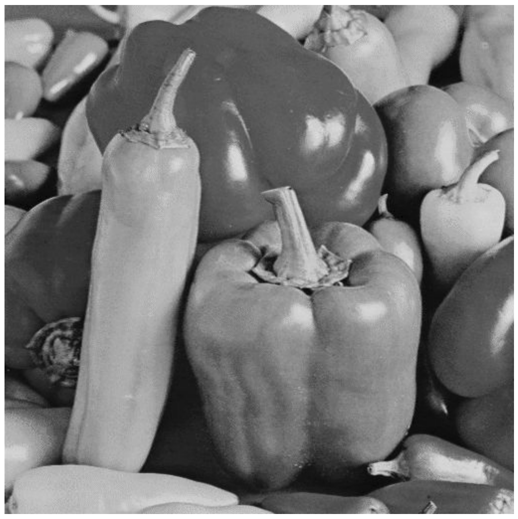

Target Image


Input Histogram

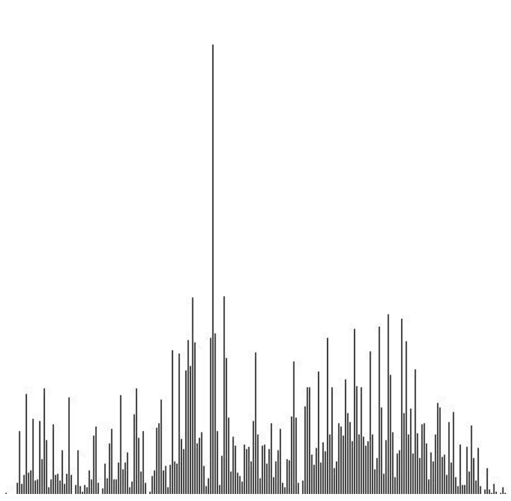

Target Histogram

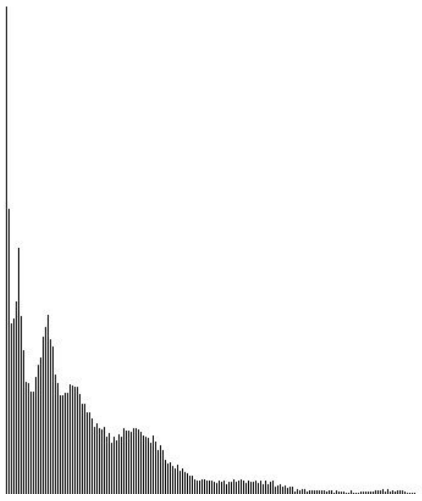

Histogram matched image

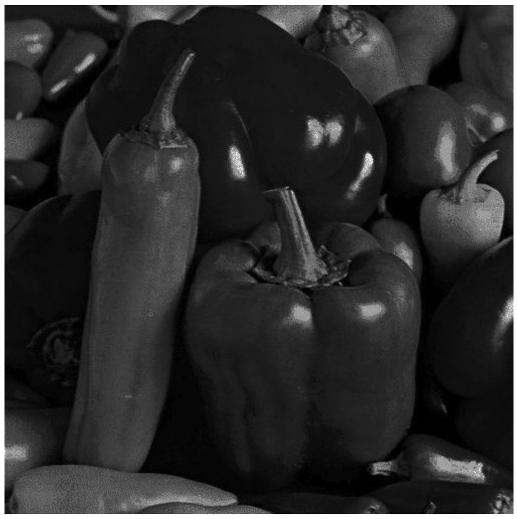

Matched Histogram

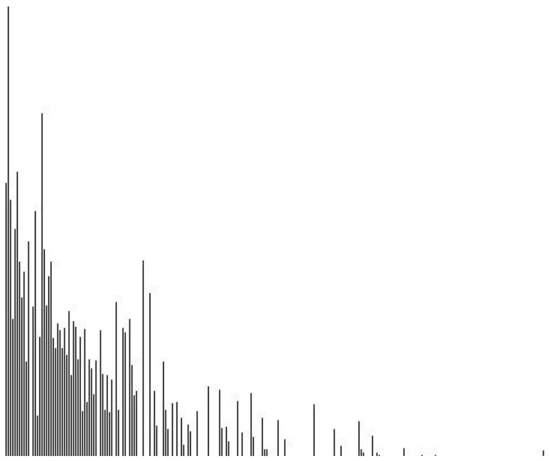

Input Image

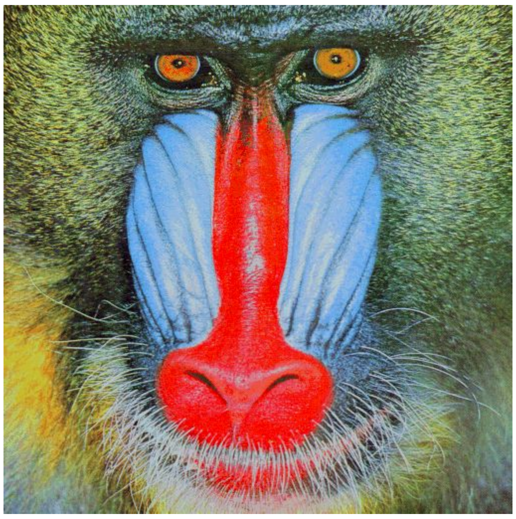
Target Image

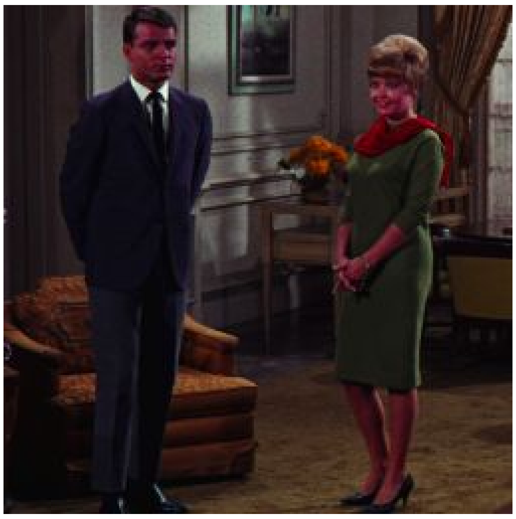

Input Histogram

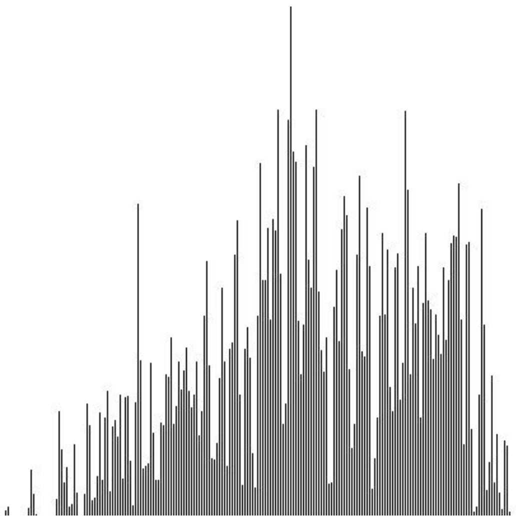

Target Histogram

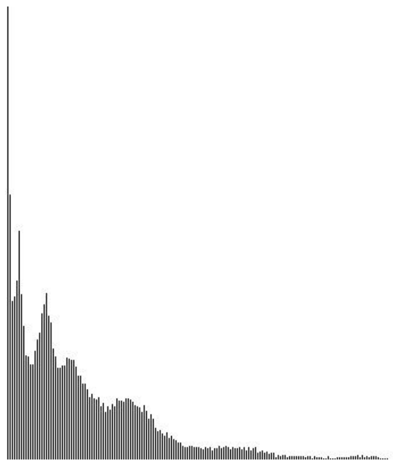

Histogram matched image

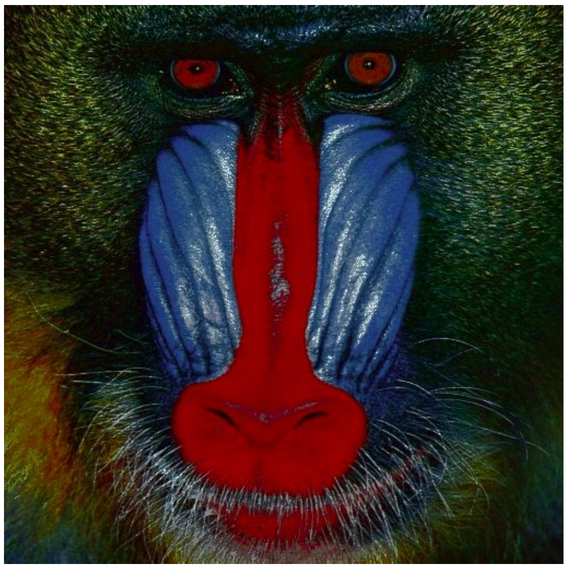

Matched Histogram

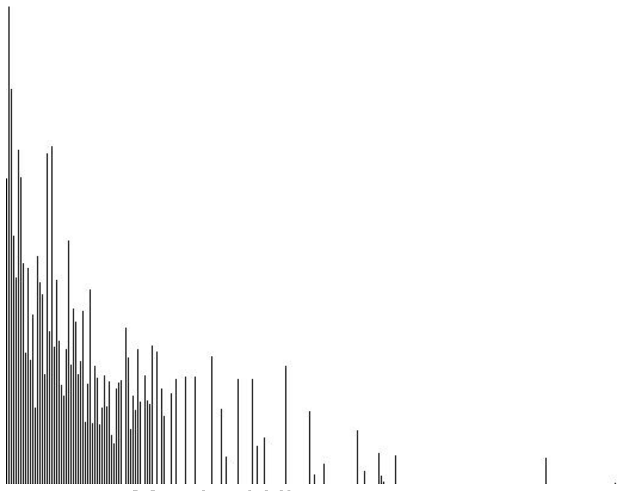

Input Image


Target Image


Input Histogram


Target Histogram


Histogram matched image


Matched Histogram

Analysis:

---

1. If we are using WSL or SSH, the imshow function will not work. So we will have to comment it out where the display is not working.
2. We could also have read all the images in IMREAD_COLOR mode to obtain the 3 channel image, and converted it into YCrCb directly from which the Y channel could be used for the operations.
3. Whereas we read the image in the IMREAD_UNCHANGED mode to compare the different color spaces and the output that we will get from each of them.
4. In case of histogram equalization we observe significant improvement in contrast of image for both color and grayscale image.
5. With histogram equalization smaller details are more easily observable as can be seen in example one.
6. In case of histogram matching we observe that if we provide a whitewashed image as a target image then the input image also shows the same effect after matching, as can be seen in result 1 and 2.
7. Similarly if we provide a dark image the resulting image turns out to be dark, as can be seen in result 3 and 4. This is applicable for both color and grayscale image.
8. We observe that even after histogram matching the final histogram looks similar to the target histogram but it prevails its original shape to some extent as can be seen in result 1 and 2.

References:

1. https://towardsdatascience.com/histogram-matching-ee3a67b4cbc1
2. https://en.wikipedia.org/wiki/Histogram_matching#:~:text=In%20image%20processing%2C%20histogram%20matching,specified%20histogram%20is%20uniformly%20distributed
3. https://www.tutorialspoint.com/dip/histogram_equalization.htm
4. https://medium.com/@kyawsawhtoon/a-tutorial-to-histogram-equalization-497600f270e2
5. https://hypjudy.github.io/2017/03/19/dip-histogram-equalization/.