COMP 408/508 Programming Assignment
Histogram Equalization

Cakmak, Serike
Department of Computer Engineering
Koc University
Istanbul, TURKEY
scakmak13@ku.edu.tr

***Abstract*—This document is a report for the assignment of Computer Vision course of the Computer Engineering Dept., Koc University. Aim of this assignment was to be familiar with histogram equalization operations using OpenCv.**

***Keywords*— histogram, histogram equalization, gamma correction, intensity, luminance**

I. Introduction

In this paper, the use of OpenCv operations related to histograms and also the algorithm implementations are explained.

II. YCbCr Color Space

YCbCr or Y′ CbCr, is a family of color spaces. Y′ is the luma component and Cb and Cr are the blue-difference and red-difference chroma components.

In the implementation image is specified by the user, since it is a color image there is a need to be converted for that image to be able to perform some basic operations related with histograms. RGB channels in the image are converted to YCbCr so that its intensity channel Y can easily be used to find the histogram and to equalize the histogram.

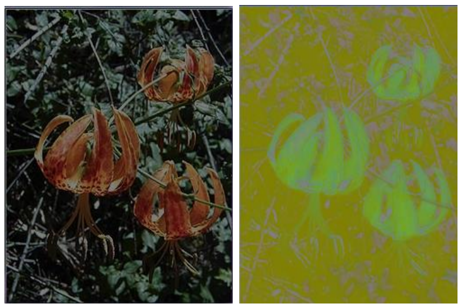
Figure 1. Left: Original Image Right: YCbCr converted image

Conversion from BGR (default color format in OpenCV) to YCbCr is handled by OpenCv:

```
cvtColor(img, ycc_img, CV_BGR2YCrCb);
```

Figure 1 (right) shows the intensity values of the original image. But as seen in the right image of Figure 1, color information cannot be understood by looking to the image.

III. Intensity Histogram

The histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels among those grayscale values. Since our image is a color image the luminance channel which can be found from the previous step is used for the calculation of histogram.

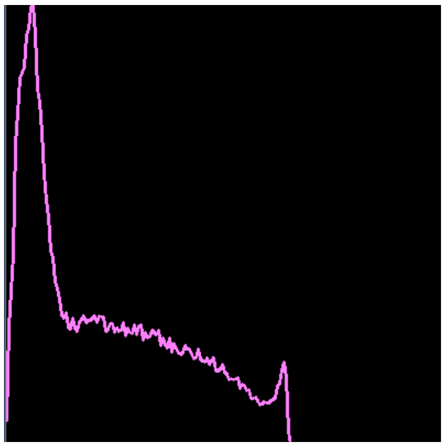
Figure2. Intensity Histogram of image which is calculated by OpenCv's calcHist function.

The intensity histogram obtained in Figure 2 is drawn one more time by using my own implementation to compare these with each other.

In my implementation, I have counted the number of each intensity value. After that I have found the maximum intensity value to use in the normalization. By the use of drawing funcitons in OpenCv the result has appeared as in Figure 3.

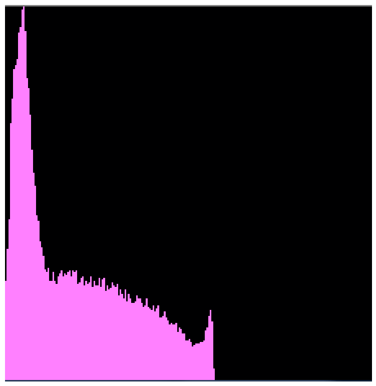

Figure 3. Intensity histogram with my implementation

By comparing Figure 2 and Figure 3 it is possible to say that OpenCv also uses the same algorithm while calculating the histogram since these two figures are almost exactly the same.

IV. GAMMA CORRECTION

Gamma correction is a nonlinear operation that is used to code the luminance. To apply gamma correction below formula is used where T is the mapping function:

$$T(x) = x^{1/\gamma}, 0 \le x \le 1$$

Here, a gamma value is needed and its range depends on the application. Figure 4 is an illustration of gamma correction with gamma value 2.0.

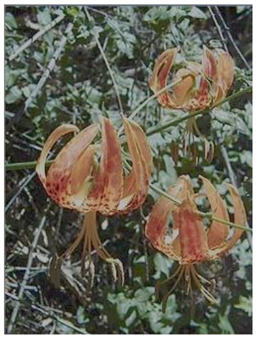

Figure 4. Gamma Correction on the original image

V. HISTOGRAM EQUALIZATION

The goal of histogram equalization is to produce an output image with a flat (uniform) histogram. The steps of histogram equalization algorithm described in lecture notes involve computing the normalized histogram, computing the cumulative histogram and compute the value for all pixels [1]. Histogram equalization was also performed on the luminance channel (Y). Figure 5 represents the result of this algorithm applied on the image.

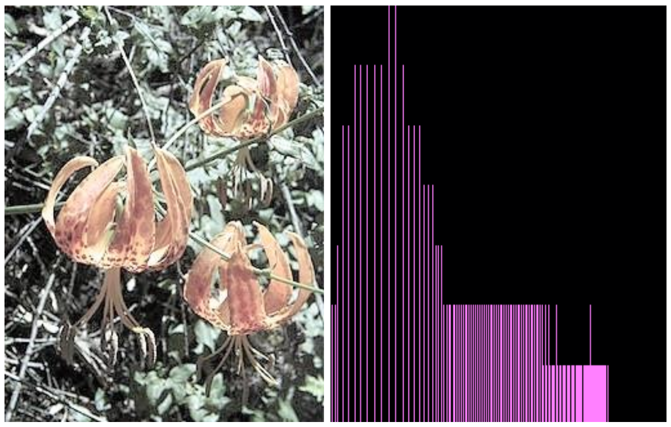

Figure 5. Equalized color image and its histogram

The same algorithm was implemented also by using OpenCv's functionality:

```
equalizeHist(ycc_planes[0], ycc_planes[0]);
//equalize histogram on the 1st channel (Y)
```

Equalized image which was obtained by using equalizeHist function of OpenCv is also shown in Figure 6 to emphasize the similarities or differences between using these two methods. However, there exists no difference between two methods and Figure 6 is highly consistent with the result of above algorithm.

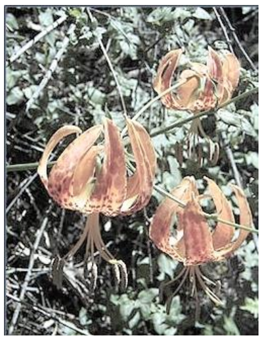

Figure 6. Equalized image with equalizeHist() function

VI. LOCALLY ADAPTIVE HISTOGRAM EQUALIZATION

While global histogram equalization can be useful, for some images it might be preferable to apply different kinds of equalization in different regions. In such cases, instead of computing a single curve, we subdivide the image into MXM pixel blocks and perform separate histogram equalization in each sub-block. Since some blocking artifacts may occur during this process, we use moving window [2].

To perform locally adaptive histogram on the image, I have used my own implantation for histogram equalization from previous part. I divided the whole image into MXM sub images and called my equalize function with these sub images. By sliding the window I have tried to avoid from the blocking artifacts. According to my observations window size is very important for the result in locally adaptive histogram equalization. With bigger windows comes better results. However smaller windows tends to reveal the details more. As the window size increases to the size of the image, you are left with histogram equalization.

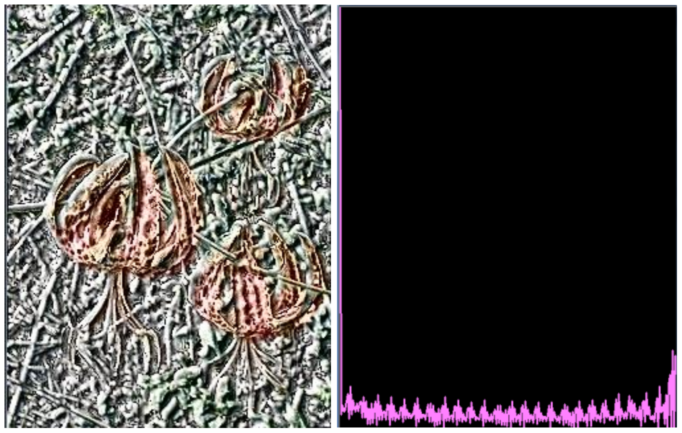

Figure 7. Locally Adaptive Histogram Equalization with window = 5 and its histogram

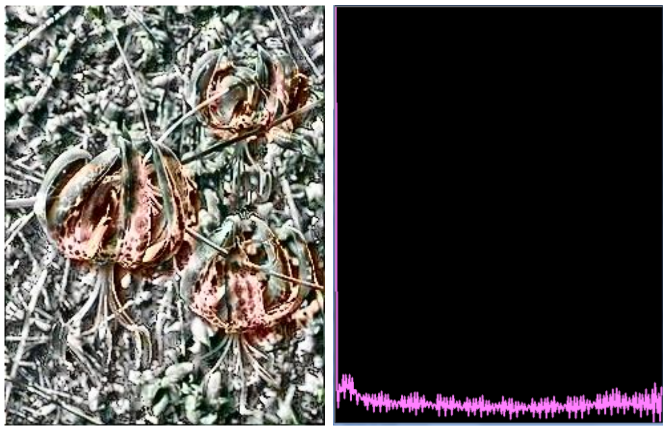

Figure 8. Locally Adaptive Histogram Equalization with window =10 and its histogram

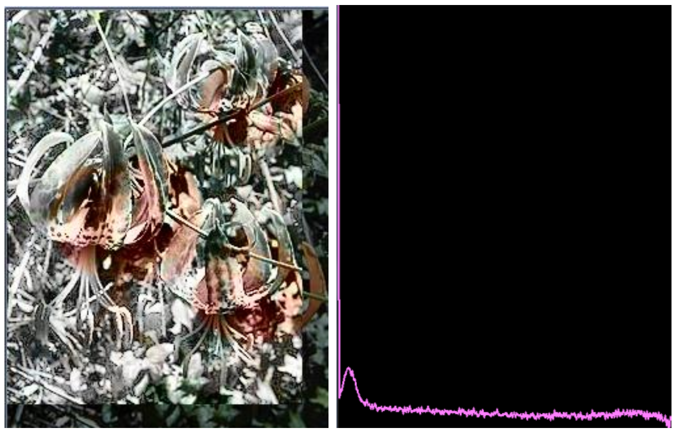

Figure 9. Locally Adaptive Histogram Equalization with window =25 and its histogram

REFERENCES

[1] "COMP 408/508 Computer Vision, Filtering and Image Enhancement " ,2013, http://home.ku.edu.tr/~yyemez/comp508/filtering.pdf

[2] Szeliski,Z. "Computer Vision: Algorithms and Applications," Springer, 2010, p.109.