Histogram Equalization

Thanya Ramanathan
Matric No. : G2303746K

1. Introduction to Histogram Equalization

A histogram is a diagram that shows how data are distributed. It is a frequently used in statistics and data analysis for visualising the frequency of certain values in a dataset. By segmenting the underlying data into discrete intervals or bins along the horizontal axis and displaying the frequency or count of data points falling into each bin along the vertical axis, histograms offer a visual representation of the distribution of the underlying data.

Histogram of an image is a visual depiction of the distribution of pixel intensities in a picture. It gives details on the incidence or frequency of various colours or intensity levels in the image. In Computer Vision, histograms are frequently used for a variety of tasks, such as segmentation, contrast adjustment, and picture enhancement.

In figure1, below, a random sample grayscale image is taken and its corresponding histogram is depicted. On the horizontal axis is the range of pixel values, generally from 0 to 256 and on the vertical axis is their corresponding frequency. Since the image is grayscale, a single histogram is enough to explain the intensity distribution. Whereas, in case of coloured images, three histograms are plotted to show the variation in red, green and blue spectrums.

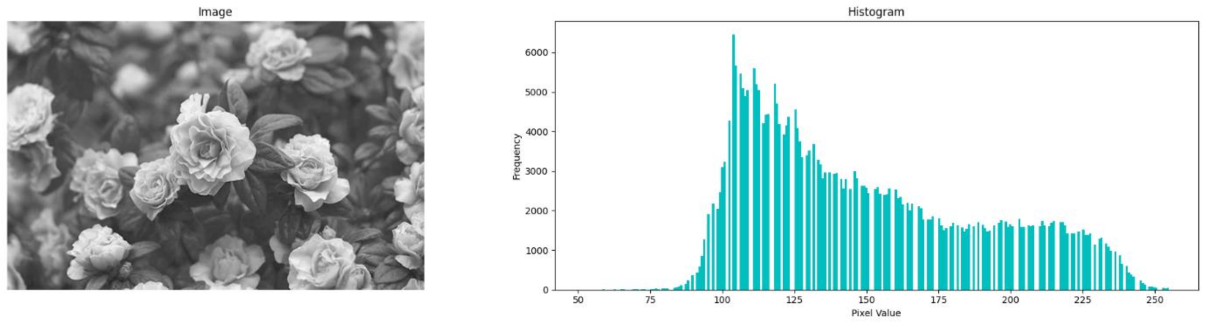

Figure 1 Sample Grayscale Image and Histogram

Histogram equalization is a technique used in image processing to enhance the contrast and improve the visibility of details in an image by redistributing the pixel intensities in a way that spreads them across the entire intensity range. It achieves this by transforming the histogram of an image so that it becomes approximately flat or uniform. Figure 2 illustrates the function of histogram equalization on the same sample image. The intensity distribution appears more spread out and the contrasts in the image have been highlighted.

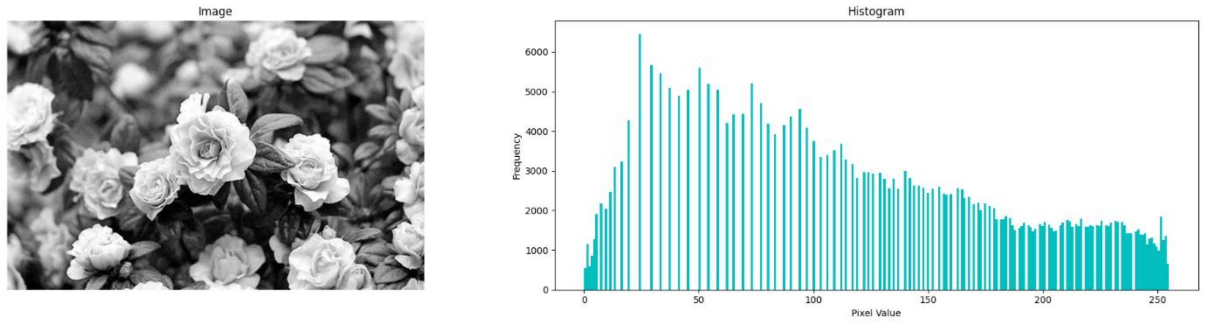

Figure 2 Sample Grayscale Image after Histogram Equalization

2. Mathematical Formulation

Taking the example of the image from figure 1 and splitting the image into rows and columns with the number of rows being M and columns N. The maximum intensity level, L = 255, the general value for 8-bit images.

To compute the histogram for the respective image,

$$H(k) = \sum_{i=1}^{M}\sum_{j=1}^{N} \delta(f(i,j) - k)$$

Where f(i,j) refers to the pixel intensity at co-ordinates (i,j), and δ is the Dirac delta function (which is 1 if its argument is zero, and 0 otherwise). That is, for a specific k-value, the Dirac delta function returns 1 every time f(i,j) = k, and count of these δ = 1 are stored as the frequency for that k value.

Once, the histogram is calculated, the cumulative distribution function (CDF) is computed. t represents the cumulative sum of probabilities associated with pixel intensities up to a given intensity level k.

$$C(k) = \sum_{l=0}^{k} P(f(i,j) = l)$$

Where P(f(i, j) = l) is the probability distribution function that denotes the that a pixel in the image has an intensity level l. In practice, P(f(i,j)=l) is estimated as the relative frequency of intensity level l in the image. It is computed by dividing the count of pixels with intensity l (i.e., H(l)) by the total number of pixels in the image (MN):

$$C(k) = \sum_{l=0}^{k} \frac{H(l)}{MN}$$

After calculating the CDF, it's normalized to ensure that it spans the entire intensity range. This step scales the CDF values to the desired range, typically from 0 to the maximum intensity level (L in this case).

$$S(k) = round(C(k).L)$$

These normalized values now replace the original pixel values in the image and an equalized new image is formed.

3. Implementation

Link to Google Colab Notebook:
https://colab.research.google.com/drive/1NRITKmPlng5wjZAJIIyb_tbnnTFhEty5?usp=sharing

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

This function equalizes the histogram by computing the probability distribution function and subsequently the cumulative distribution function for each channel and the new channel values are undated and returned

```
def hist_eq(channel):
    channel= np.asarray(channel) flat = channel.flatten()

    height, width = img.shape[:2] num_pixels = np.zeros(256, dtype=int) for pixel in flat:
    num_pixels[pixel] += 1

    total_pixels = sum(num_pixels) pdf = []
    pdf = num_pixels/total_pixels cdf = []
    cdf = [pdf[0]]
    for i in range(1, len(pdf)):
    cdf.append(cdf[i-1] + pdf[i]) cdf = np.array(cdf)

    n_cdf = cdf*255 #(L-1)=256-1=255 hel = np.array(n_cdf)
    hel = hel.astype('uint8')
    channel_new = hel[flat]

    return channel_new
```

This is a visualisation function used to plot the histograms of the various channels (3 channels in this project)

```
def hist_plot(b,g,r, titles):
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))
    axes[0].hist(b, bins=256, color='blue', alpha=0.5, histtype='stepfilled')
    axes[0].set_xlim([0, 260])
    axes[0].set_title(titles[0])

    axes[1].hist(g, bins=256, color='green', alpha=0.5, histtype='stepfilled')
    axes[1].set_xlim([0, 260])
    axes[1].set_title(titles[1])
    axes[2].hist(r,    bins=256,    color='red',    alpha=0.5,    histtype='stepfilled')
    axes[2].set_xlim([0, 260])
    axes[2].set_title(titles[2])

    plt.tight_layout() plt.show()
```

```
path = "/content/sample06.jpg"
img = cv2.imread(path)
cv2_imshow(img)
cv2.waitKey(0) cv2.destroyAllWindows()
```

First the histogram equalisation is performed on the RGB channels

```
img_b, img_g, img_r = cv2.split(img)
img_b= np.asarray(img_b)
img_g= np.asarray(img_g)
img_r= np.asarray(img_r)

flat_b = img_b.flatten()
flat_g = img_g.flatten()
flat_r = img_r.flatten()
t = ['Blue','Green','Red']
hist_plot(flat_b, flat_g, flat_r,t)
```

```
blue = hist_eq(img_b)
green = hist_eq(img_g)
red = hist_eq(img_r)
hist_plot(blue,green,red,t)
```

```
blue = np.reshape(blue, img_b.shape)
green = np.reshape(green, img_g.shape)
red = np.reshape(red, img_r.shape)
equalized_image = cv2.merge([blue, green, red])

cv2_imshow(equalized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Using HSV channels to perform histogram equalisation. In HSV spectrum we only equalize the V(value) channel because it holds the information about the intensity or luminance of the image. By equalizing the Value channel, you can achieve the desired contrast enhancement effect while preserving the image's overall color appearance.

```
hsv_image = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
img_h, img_s, img_v = cv2.split(hsv_image)

img_h= np.asarray(img_h)
img_s= np.asarray(img_s)
img_v= np.asarray(img_v)

flat_h = img_h.flatten()
flat_s = img_s.flatten()
flat_v = img_v.flatten()
titles = ['Hue','Saturation','Value']
hist_plot(flat_h, flat_s, flat_v, titles)
```

```
value = hist_eq(img_v)
hist_plot(flat_h,flat_s,value,titles)
value = np.reshape(value, img_v.shape)
equalized_image_hsv = cv2.merge([img_h, img_s, value])
bgr_image = cv2.cvtColor(equalized_image_hsv, cv2.COLOR_HSV2BGR)
cv2_imshow(bgr_image)
cv2.waitKey(0)
```

The original and the equalized images are displayed side-by-side to highlight the improvement after the equalization has been performed.

```
if img.shape[0] == equalized_image.shape[0] == bgr_image.shape[0]:
    concatenated_image = cv2.hconcat([img, equalized_image, bgr_image])
    cv2_imshow(concatenated_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    else:
    print("Images have different heights. They should have the same height to concatenate horizontally.")
```

4. Output Study

a. Sample Image 1

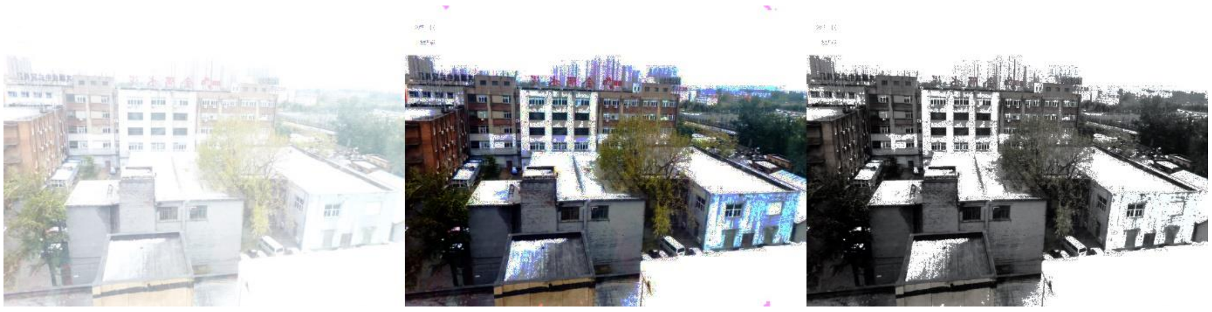

Figure 3 Original sample image and image after histogram equalization using both RGB and HSV spectrums

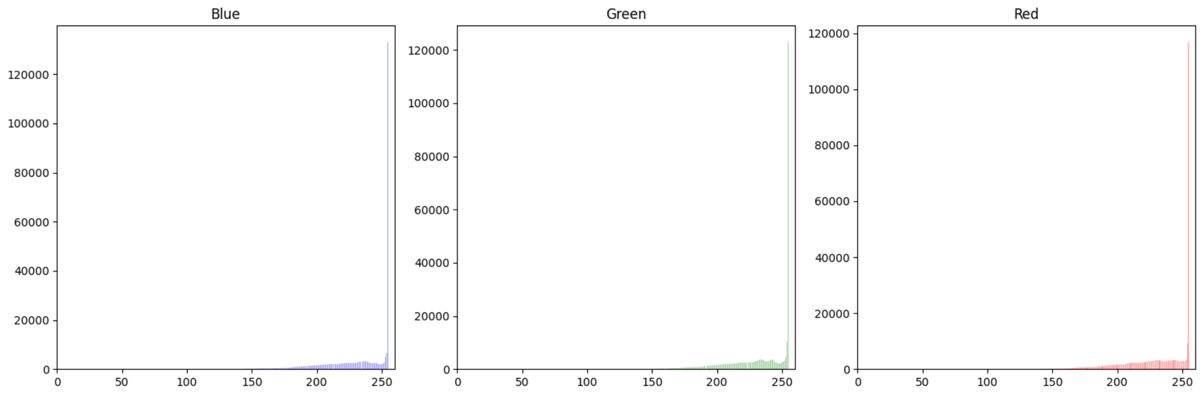

Figure 4 Original Image Histogram in RGB Spectrum

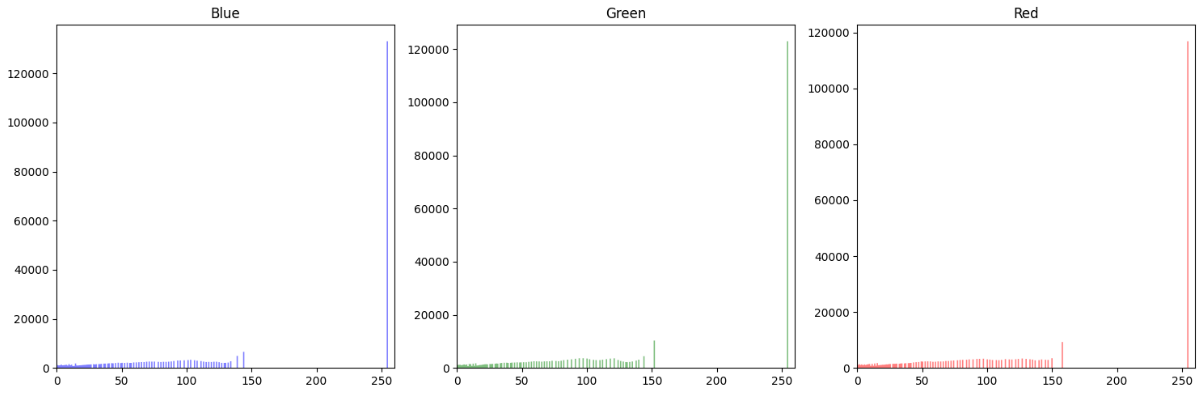

Figure 5 After Histogram Equalization using RGB Spectrum

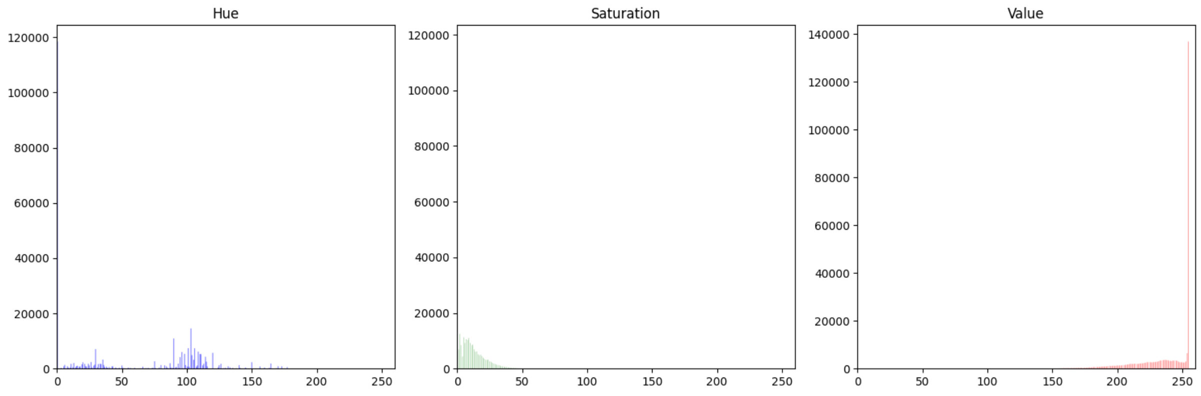

Figure 6 Original Image Histogram in HSV spectrum

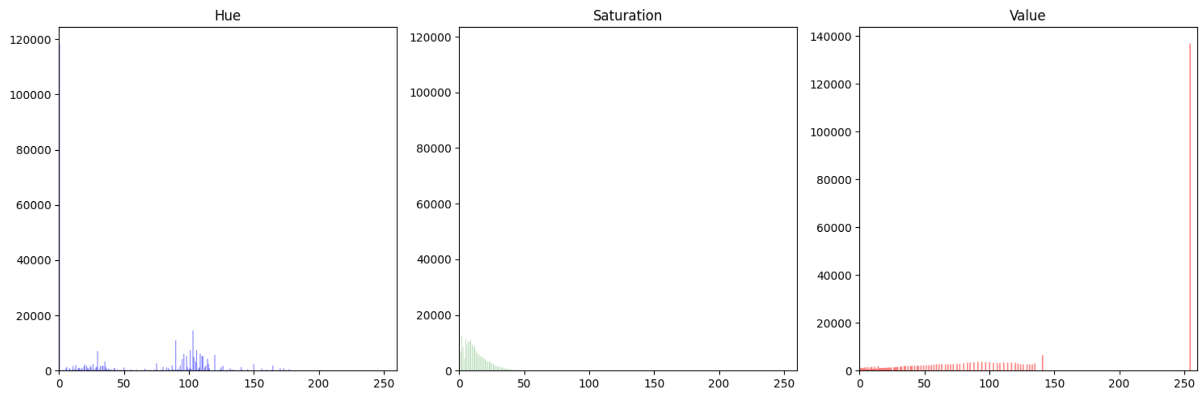

Figure 7 After Histogram Equalization in HSV Spectrum

b. Sample Image 2

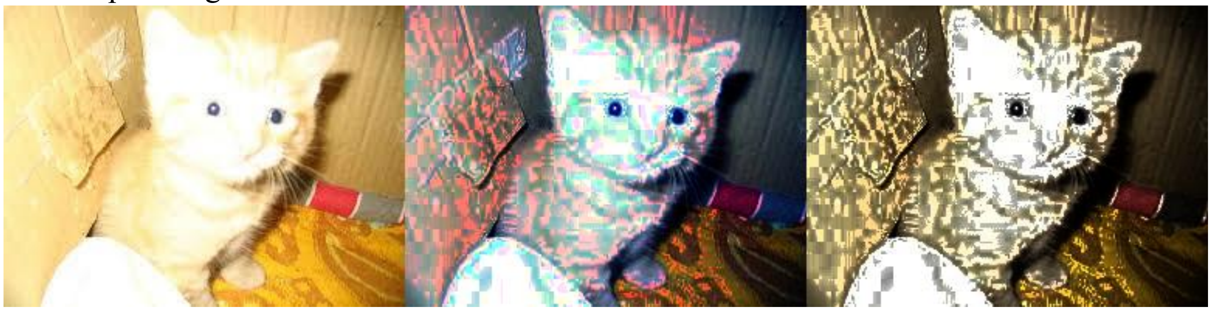

Figure 8 Original sample image and image after histogram equalization using both RGB and HSV spectrums

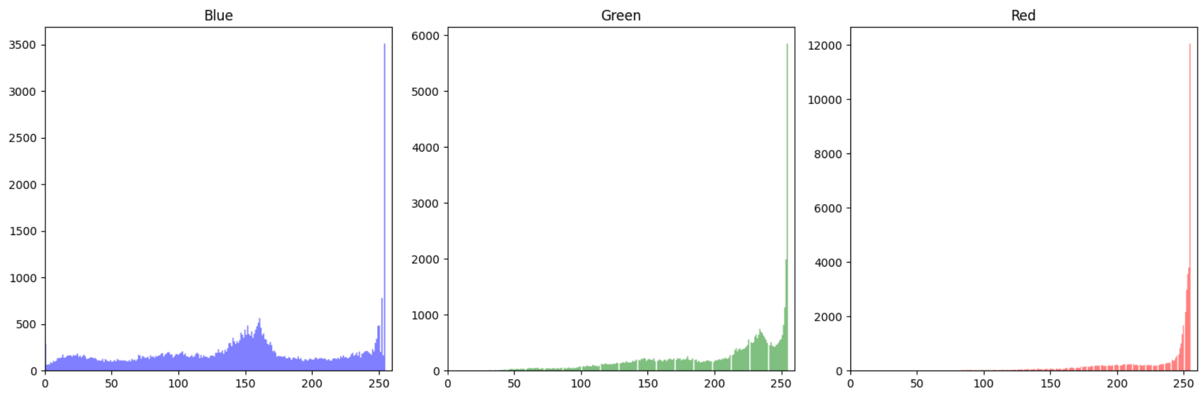

Figure 9 Original Image Histogram in RGB Spectrum

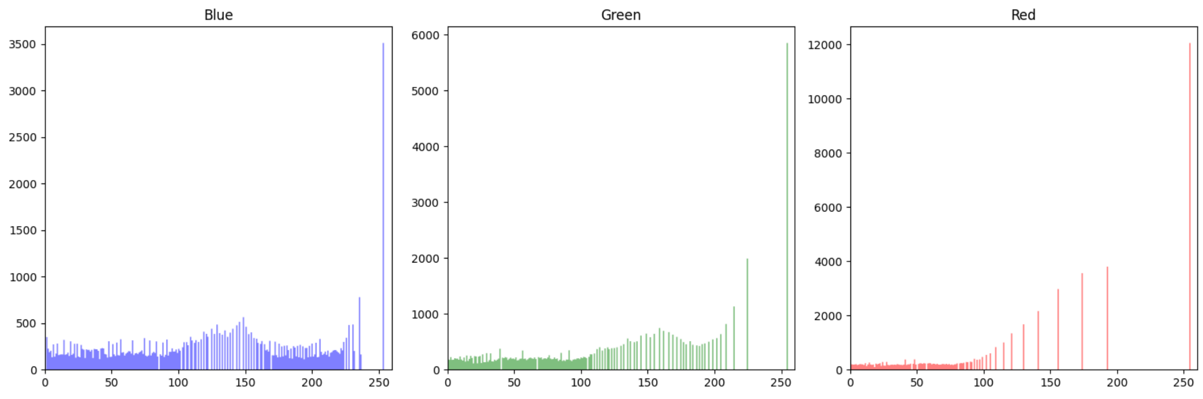

Figure 10 After Histogram Equalization using RGB Spectrum

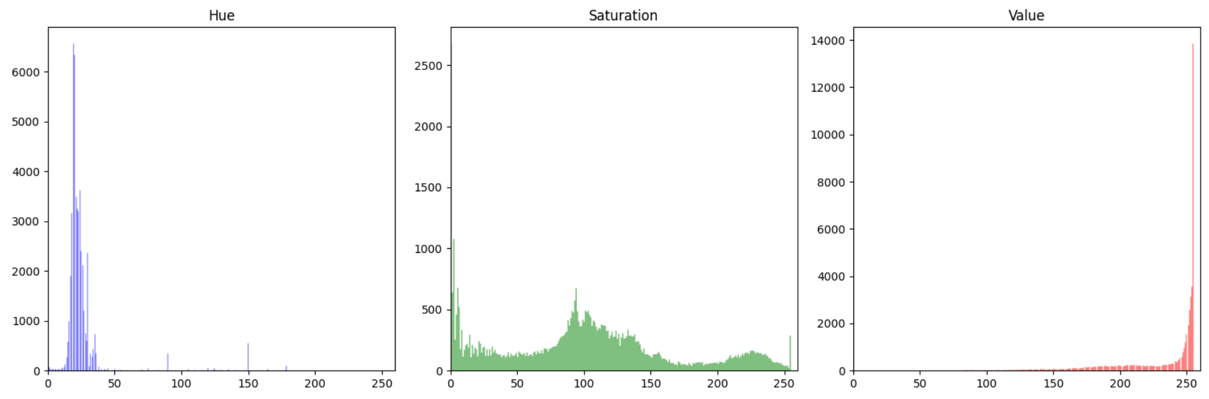

Figure 11 Original Image Histogram in HSV spectrum

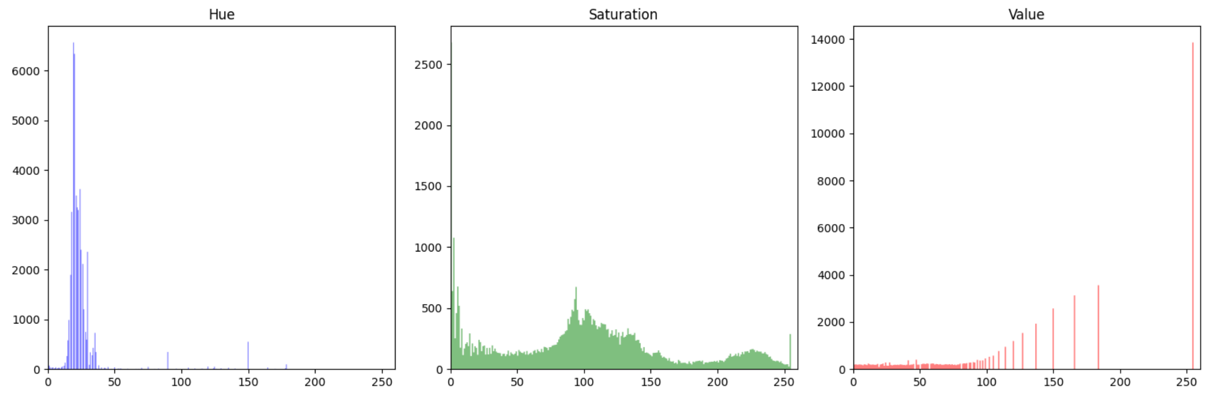

Figure 12 After Histogram Equalization in HSV Spectrum

c. Sample Image 3

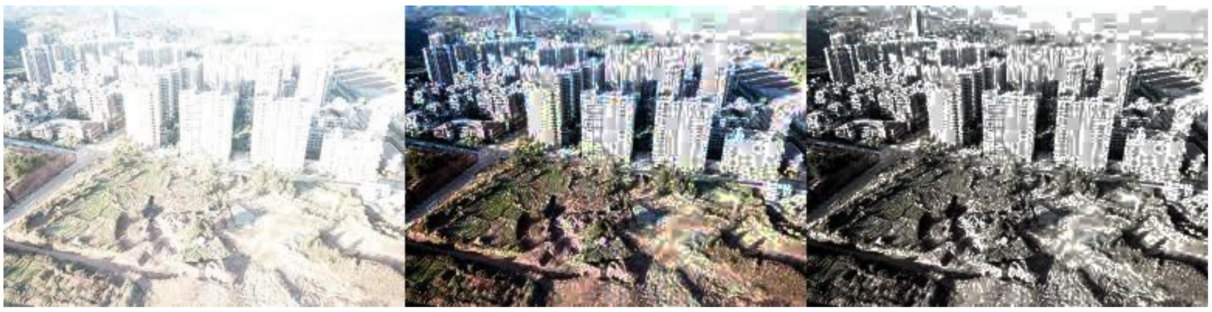

Figure 13 Original sample image and image after histogram equalization using both RGB and HSV spectrums

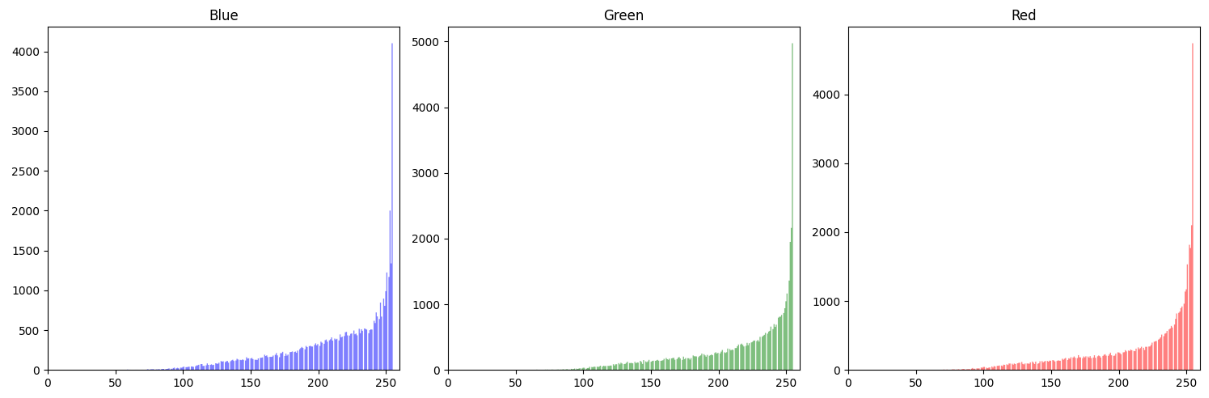

Figure 14 Original Image Histogram in RGB Spectrum

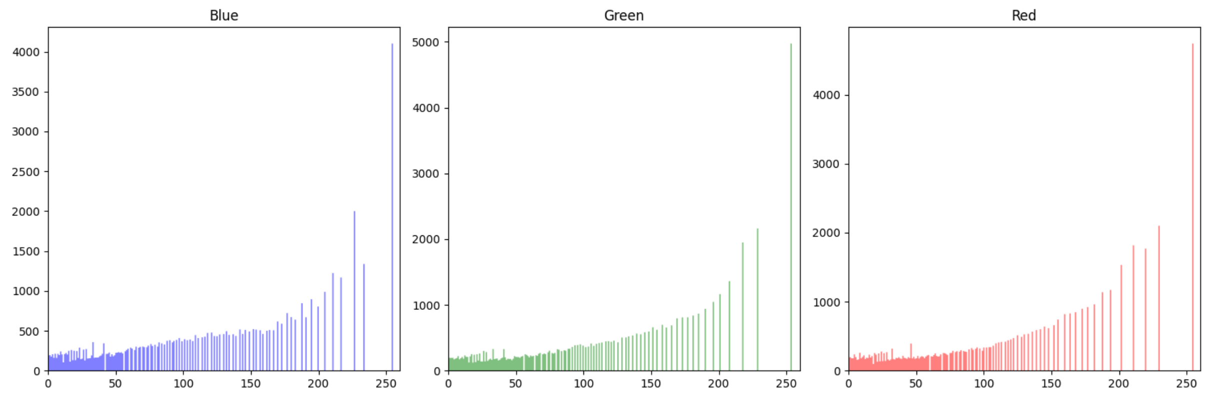

Figure 15 After Histogram Equalization using RGB Spectrum

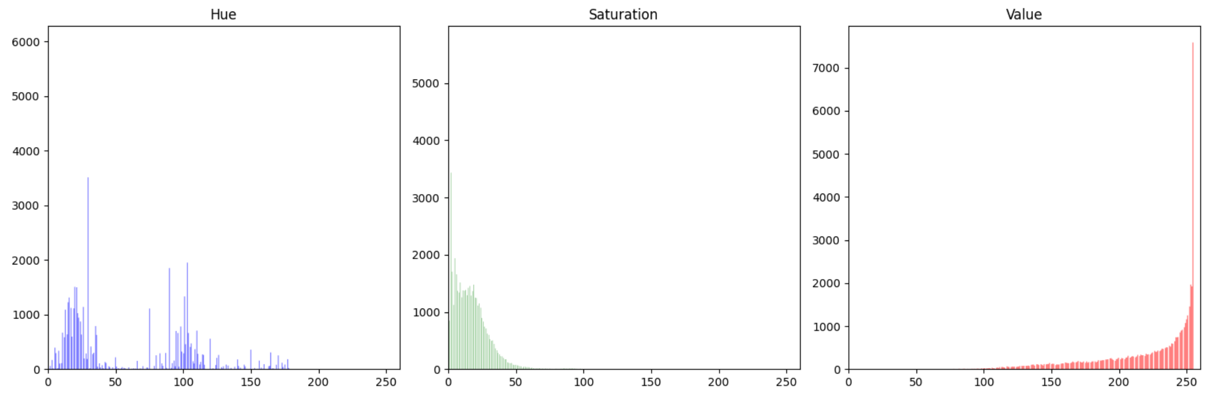

Figure 16 Original Image Histogram in HSV spectrum

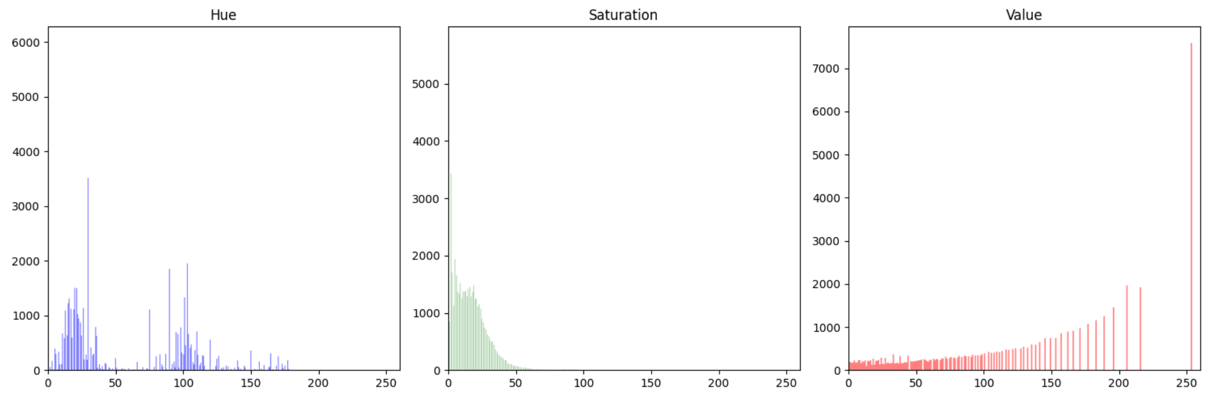

Figure 17 After Histogram Equalization in HSV Spectrum

d. Sample Image 4

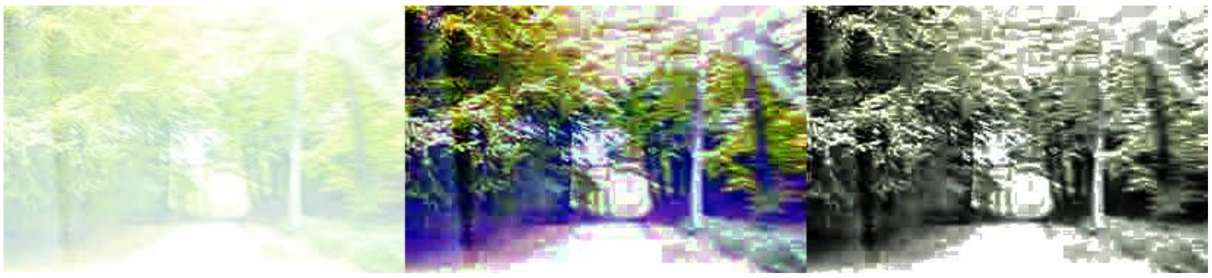

Figure 18 Original sample image and image after histogram equalization using both RGB and HSV spectrums

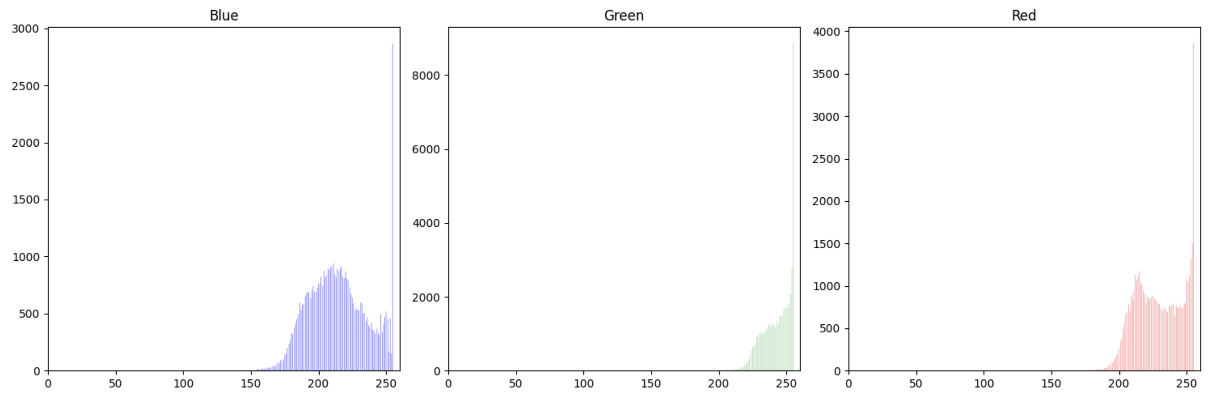

Figure 19 Original Image Histogram in RGB Spectrum

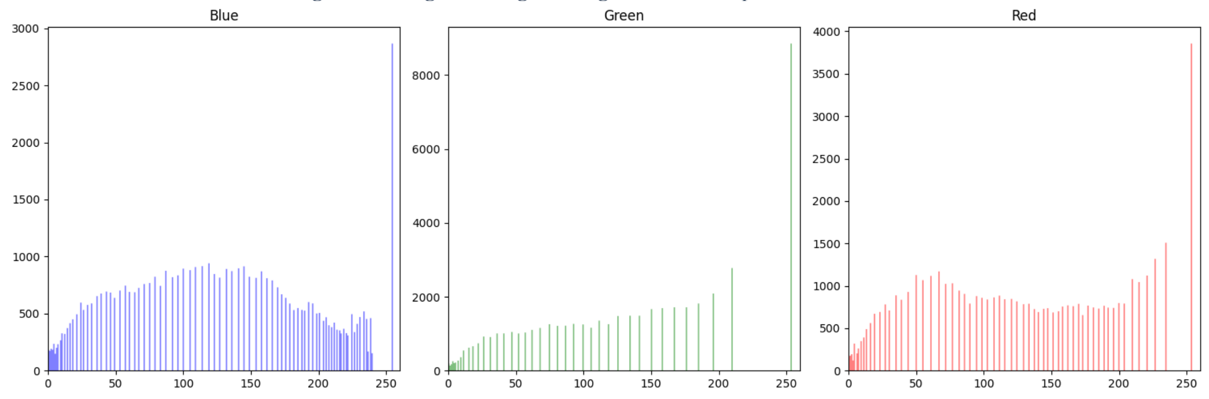

Figure 20 After Histogram Equalization using RGB Spectrum

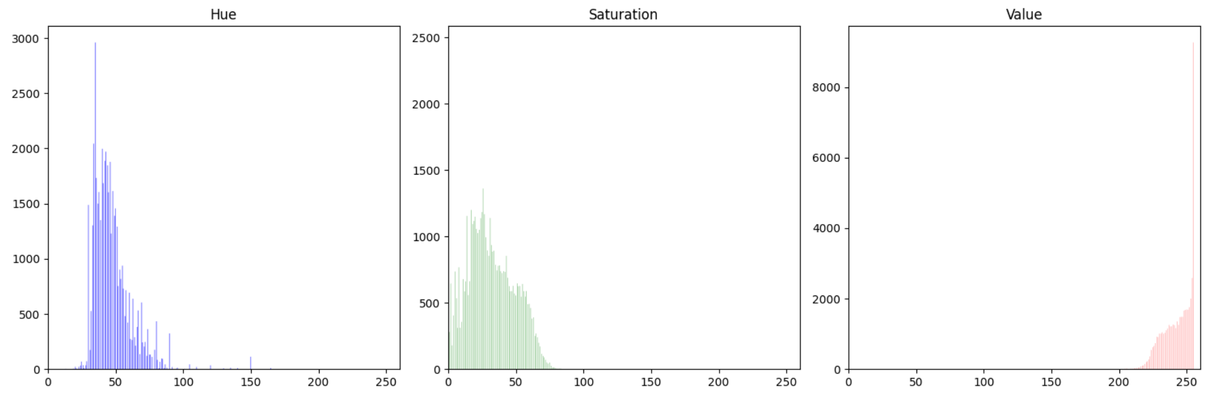

Figure 21 Original Image Histogram in HSV spectrum

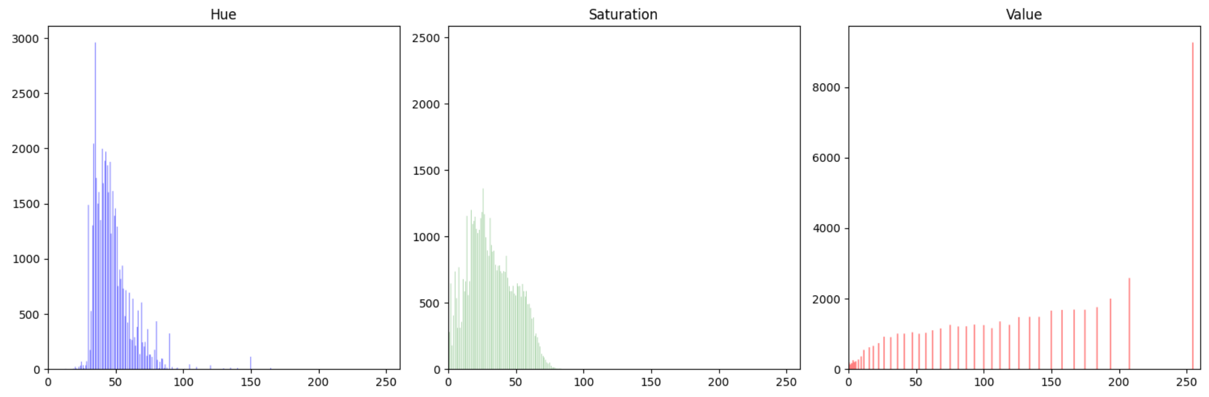

Figure 22 After Histogram Equalization in HSV Spectrum

e. Sample Image 5

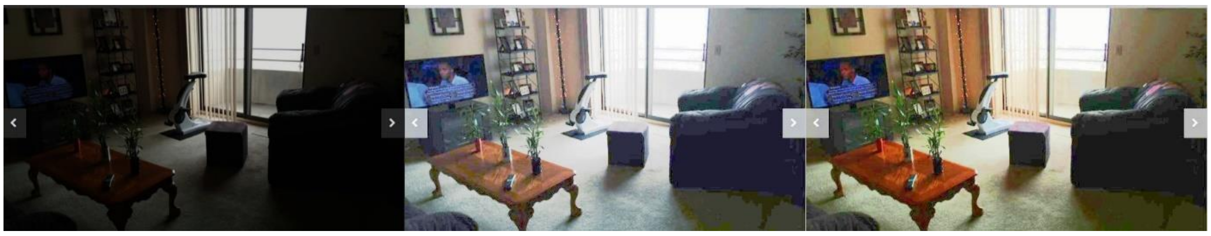
Figure 23 Original sample image and image after histogram equalization using both RGB and HSV spectrums

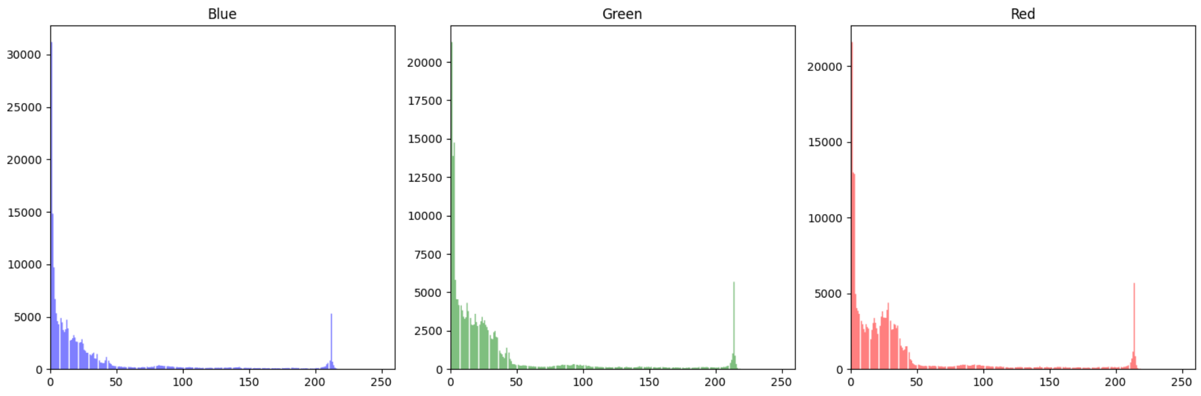

Figure 24 Original Image Histogram in RGB Spectrum

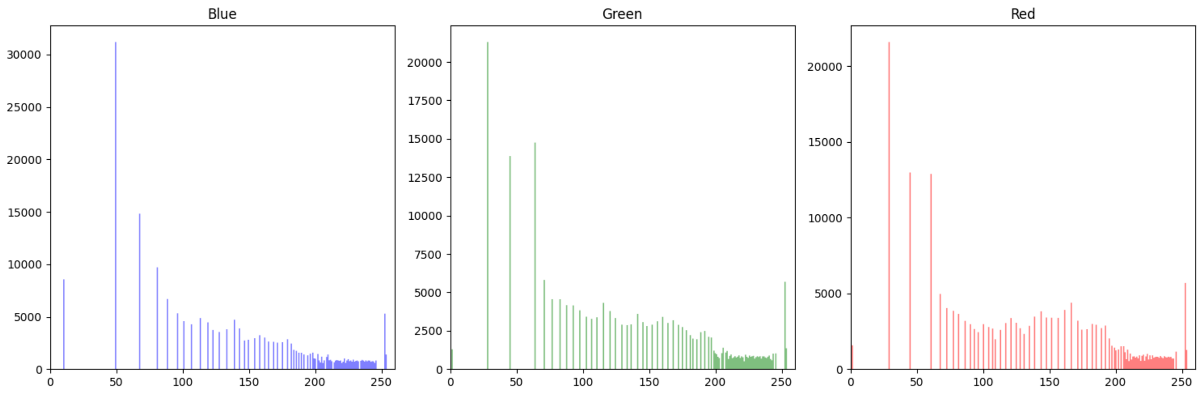

Figure 25 After Histogram Equalization using RGB Spectrum

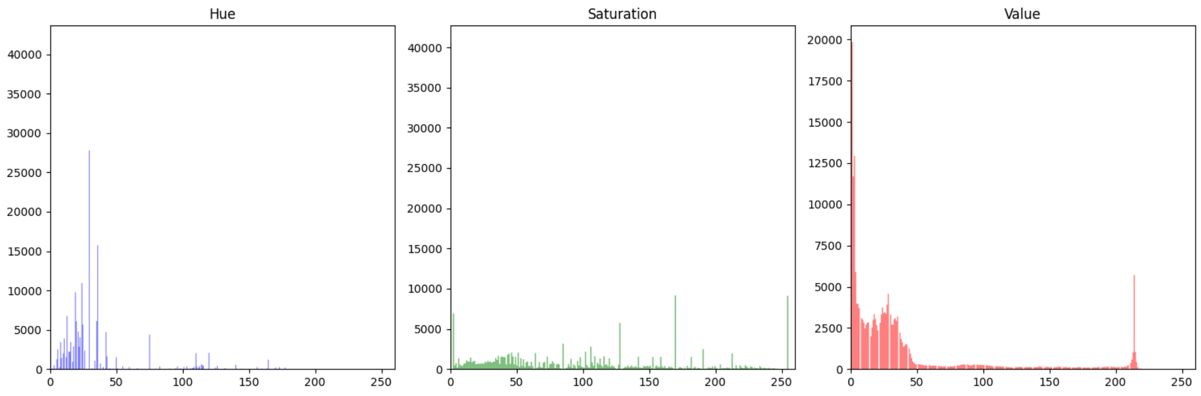

Figure 26 Original Image Histogram in HSV spectrum

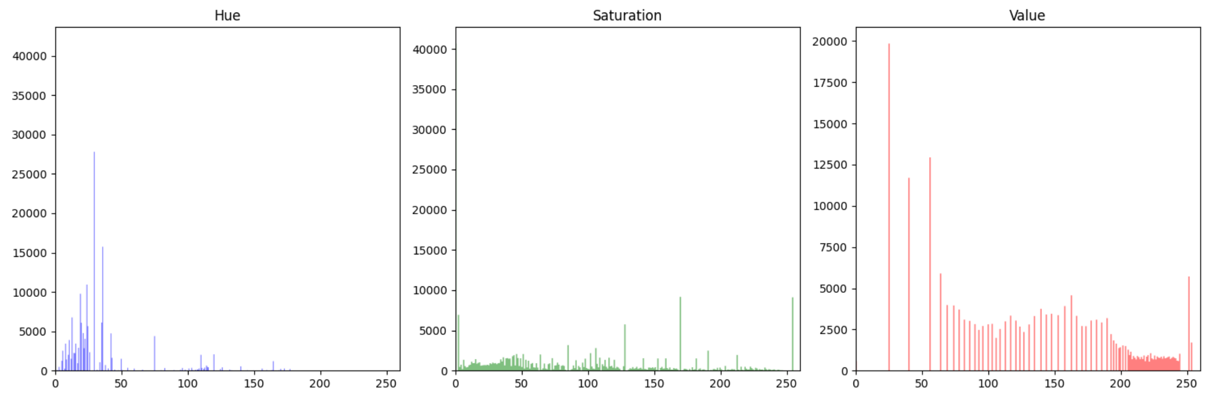

Figure 27 After Histogram Equalization in HSV Spectrum

f. Sample Image 6

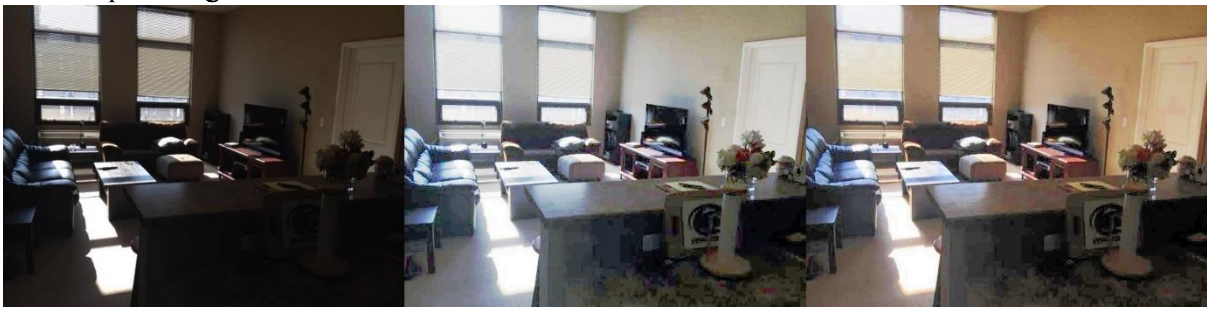
Figure 28 Original sample image and image after histogram equalization using both RGB and HSV spectrums

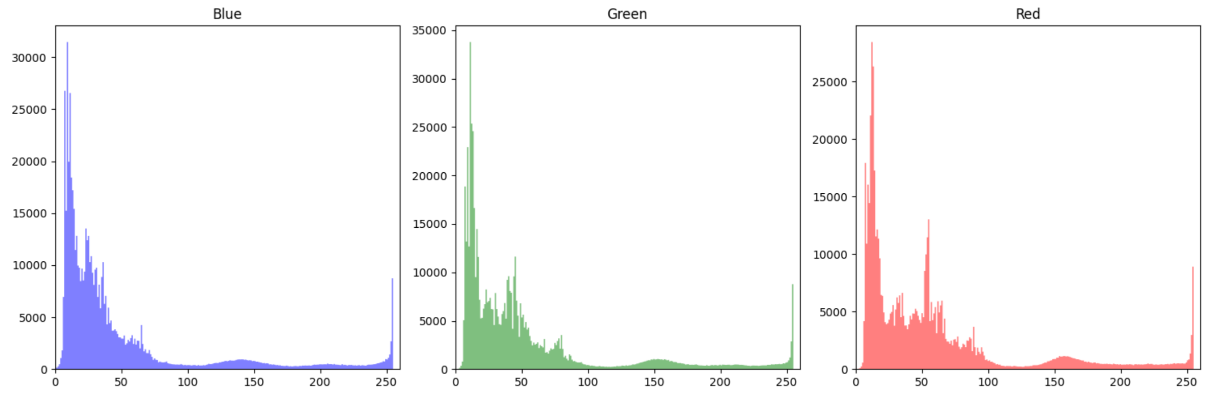

Figure 29 Original Image Histogram in RGB Spectrum

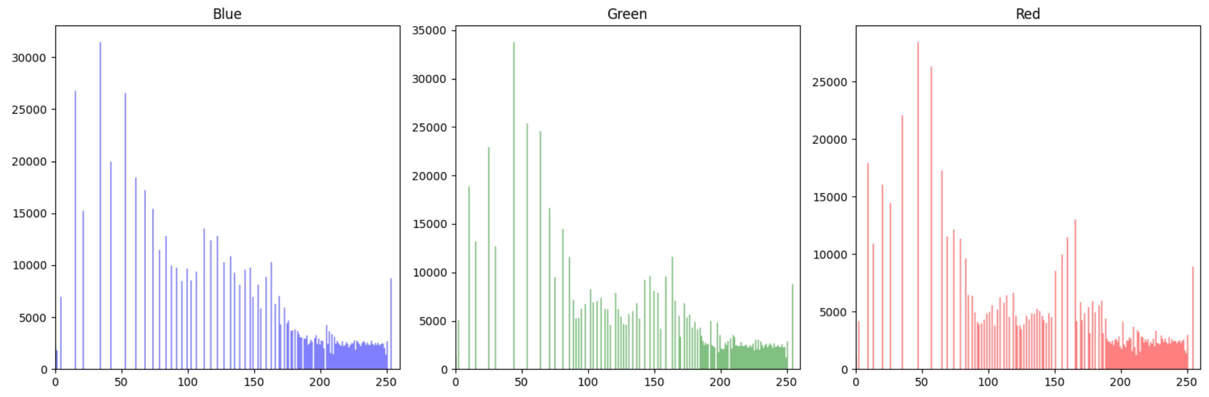

Figure 30 After Histogram Equalization using RGB Spectrum

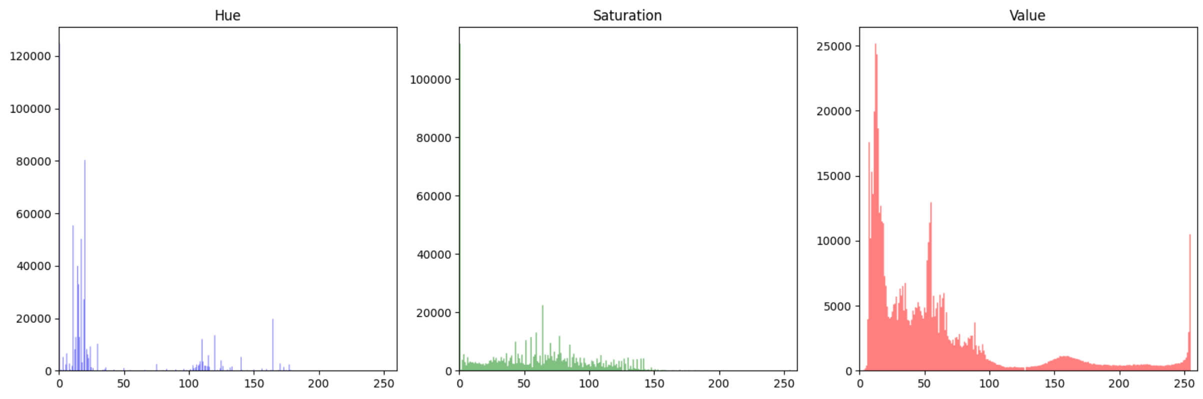

Figure 31 Original Image Histogram in HSV spectrum

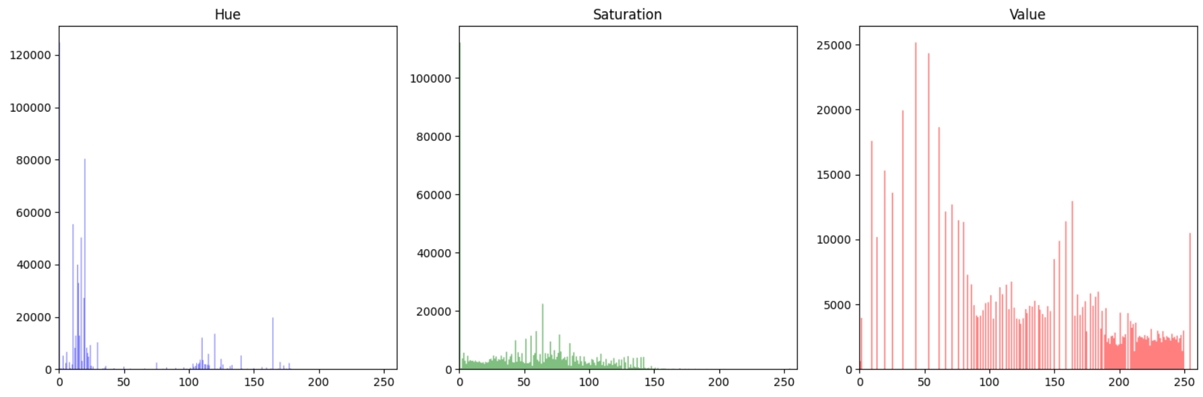

Figure 32 After Histogram Equalization in HSV Spectrum

g. Sample Image 7

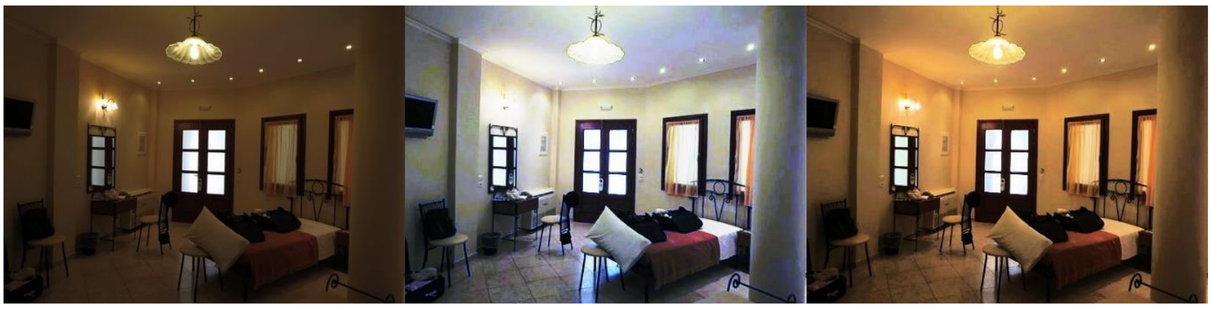

Figure 33 Original sample image and image after histogram equalization using both RGB and HSV spectrums

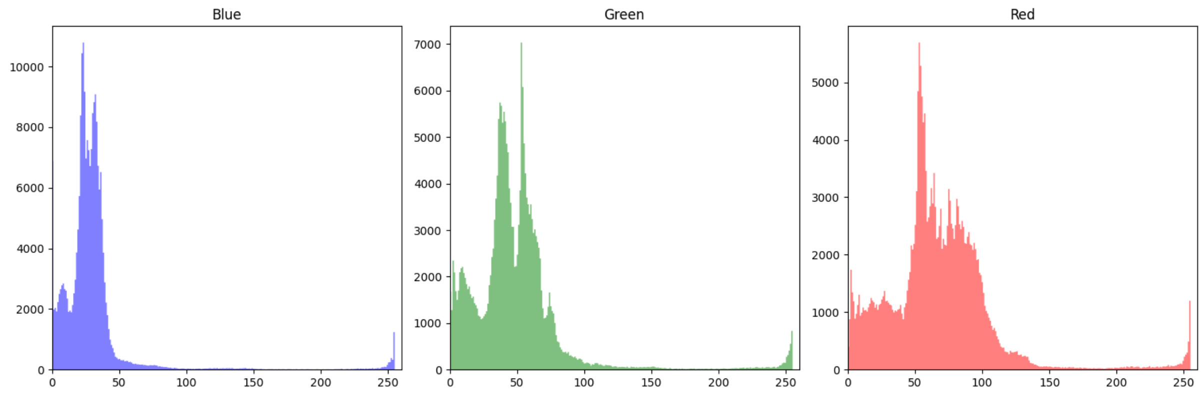

Figure 34 Original Image Histogram in RGB Spectrum

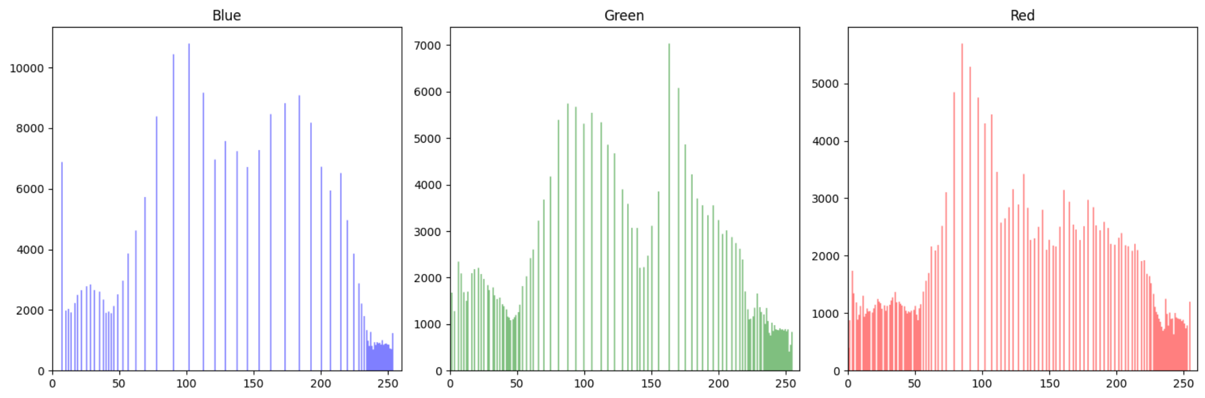

Figure 35 After Histogram Equalization using RGB Spectrum

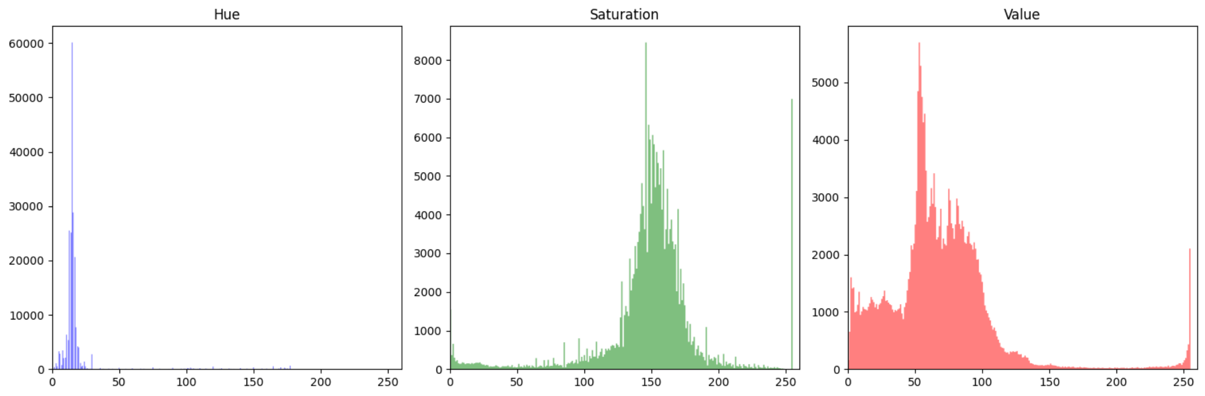

Figure 36 Original Image Histogram in HSV spectrum

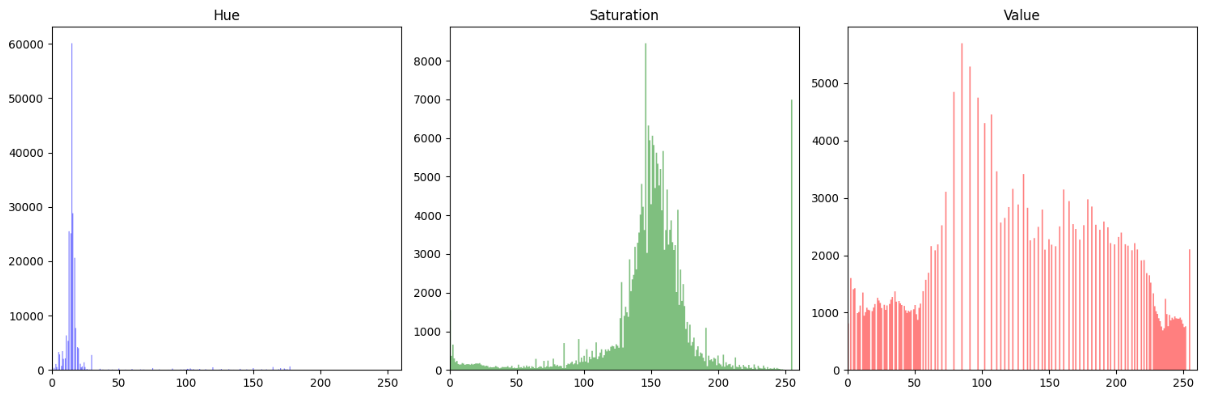

Figure 37 After Histogram Equalization in HSV Spectrum

h. Sample Image 8

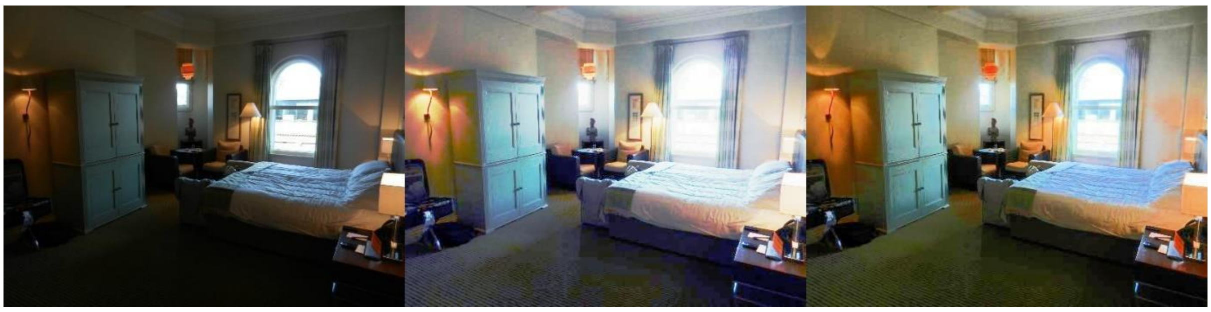

Figure 38 Original sample image and image after histogram equalization using both RGB and HSV spectrums

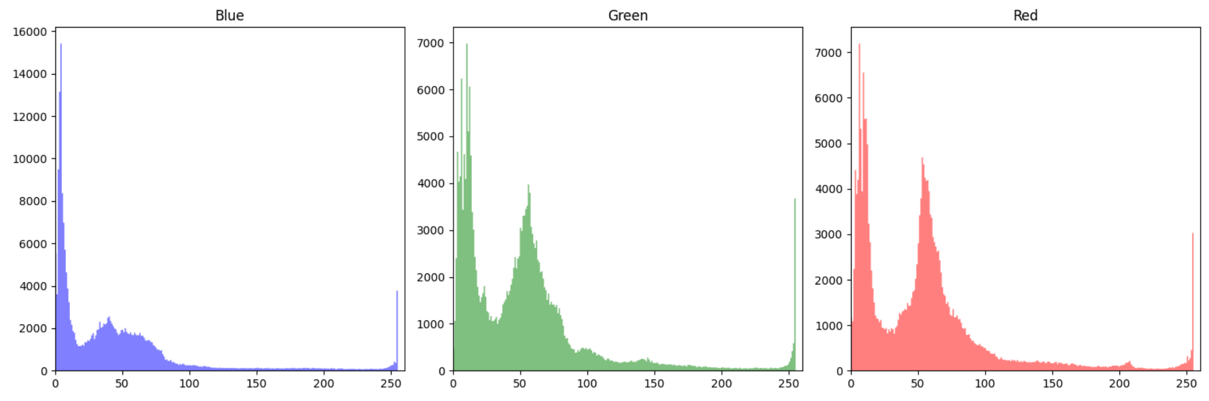

Figure 39 Original Image Histogram in RGB Spectrum

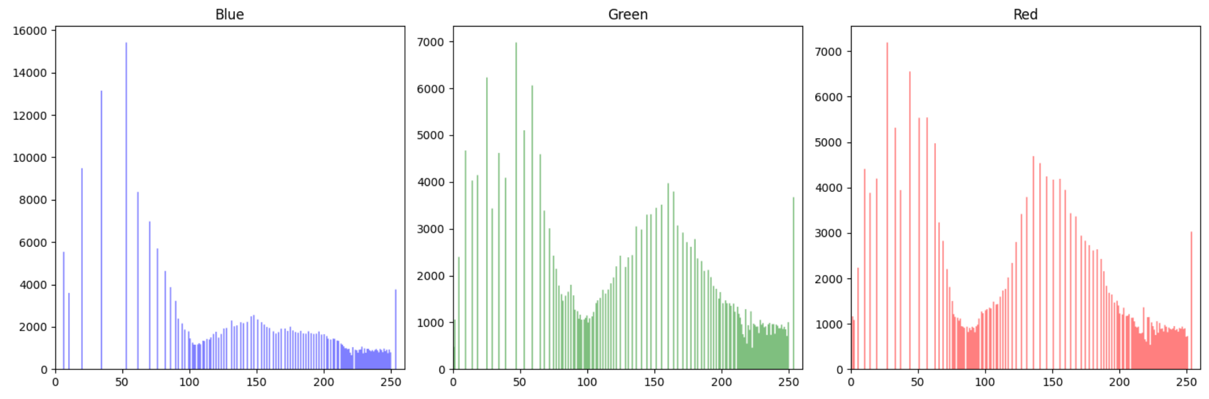

Figure 40 After Histogram Equalization using RGB Spectrum

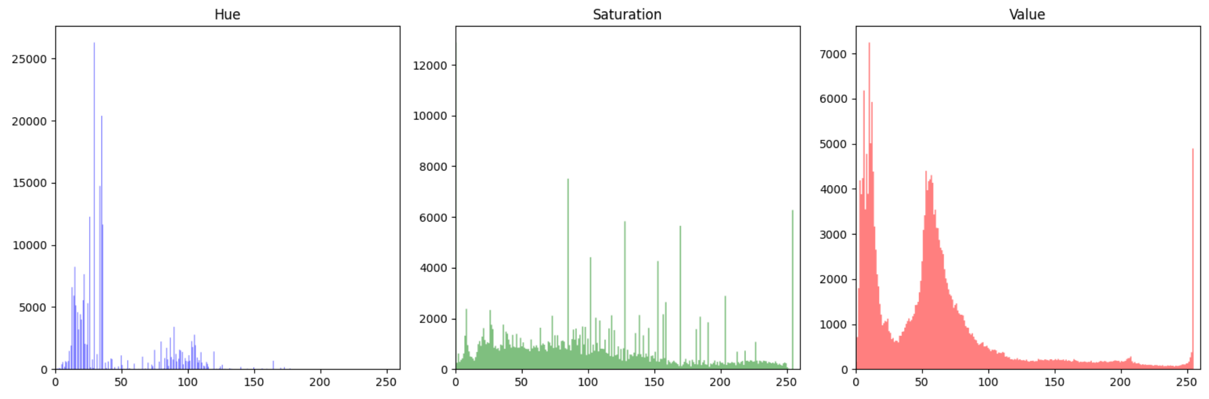

Figure 41 Original Image Histogram in HSV spectrum

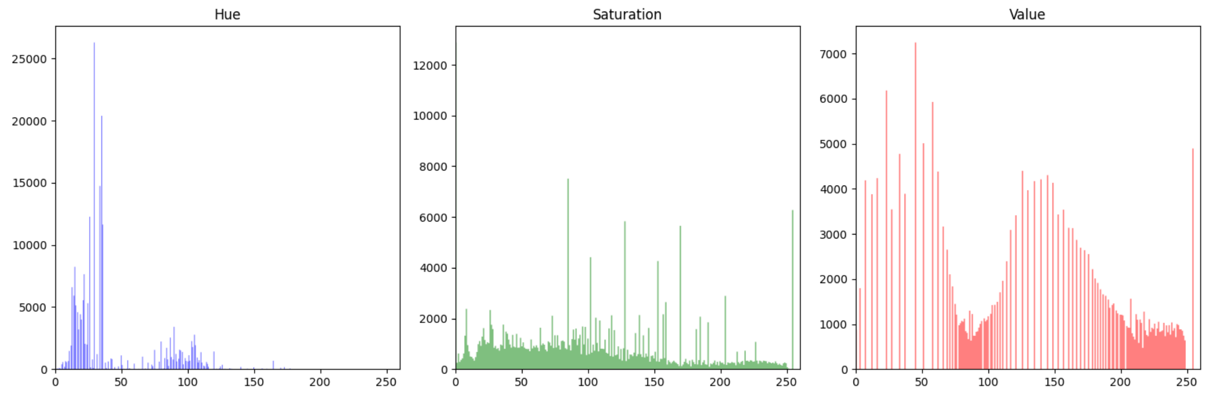

Figure 42 After Histogram Equalization in HSV Spectrum

5. Inferences

From the images and their respective outputs, it is notable that the Histogram Equalisation algorithms have performed considerably well in fulfilling their objective, However, there are some positive and negative aspects inferred from the outputs that require attention.

a. Firstly, taking into perspective the images with higher brightness, sample images 1,2,3 and 4, both algorithms have underperformed. Histogram equalization primarily redistributes pixel intensities to achieve a more uniform distribution. In images that are already bright, there may be limited scope for further enhancing contrast without overemphasizing the brightest areas. As a result, the image may still lack the desired contrast improvement.
b. In case of the RGB algorithm, it does not inherently separate luminance (brightness) from chrominance (color information) hence equalizing RGB channels affects both their brightness and color simultaneously. Additionally, equalizing each colour channel independently can lead to color distortion.
c. On the other hand, because RGB separates color components, you can easily apply color adjustments after equalization by manipulating the individual channels. This allows for precise control over color balance.
d. In case of the HSV spectrum, since only the Value channel is equalized the control over colour enhancement is lost. This has a larger effect while dealing with bright images.
e. It can be observed that the HSV algorithm performs better in terms of the color distortion observed in the RGB algorithm. This point is clearly emphasized in sample image 4 where the colour distortion in the RGB output is visible while the HSV escapes it. This maybe because HSV separates color information from brightness, hence the equalization affects only the image's brightness, reducing the risk of significant distortion.
f. In case of the darker images, sample images 5 to 8, both the RGB and HSV algorithms have performed well. HSV stands on in bringing out the contrast in the image whereas RGB highlights the details.

It is evident that both methods have their strengths and weaknesses, hence it is essential to choose the algorithms based on the use-case. HSV could be preferred if maintaining the original colours is important, especially when working with natural photographs. However, RGB can be a more straightforward and efficient option if you want greater control over colour enhancement if your application doesn't contain colour information. For example, in case of detecting movements in the dark, the RGB algorithm may perform better due to its ability to highlight the details.

6. Improvements

a. Adaptive Equalization – To overcome the issue faced with bright images, adaptive equalization could be used. In this method, the contrast is adjusted locally based on the characteristics of each image region.

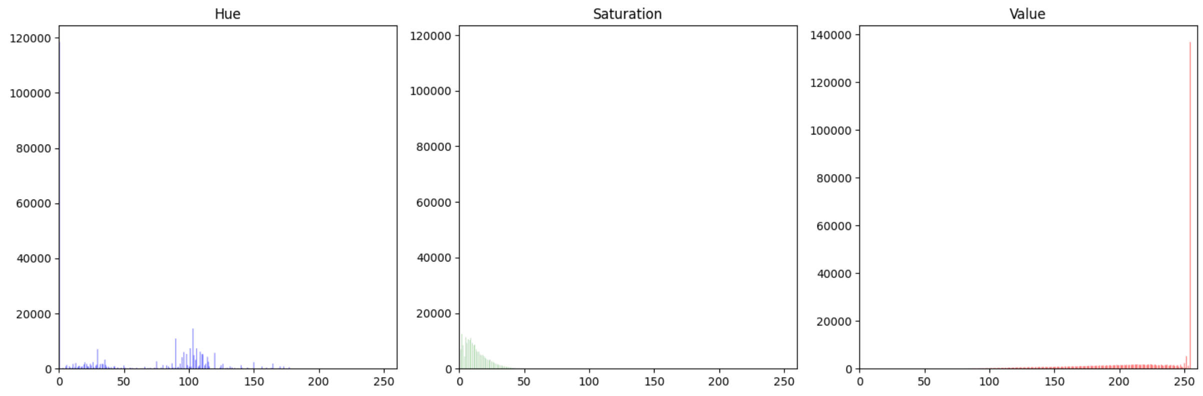

Figure 43 Histogram Output for Adaptive Equalisation

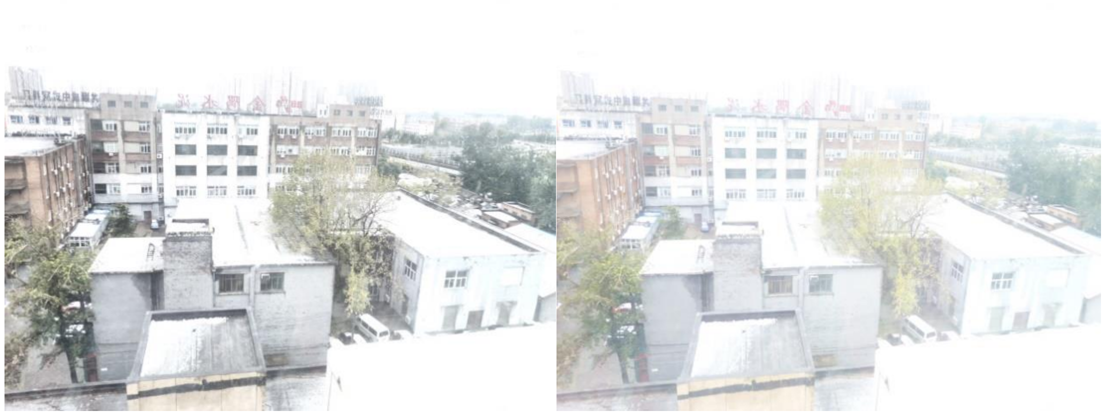

Figure 44 (a) Image output after Adaptive Equalisation (b) Original Sample Image

Applying adaptive equalisation to the HSV algorithm has helped improve the colour retention and the darker regions are better highlighted.

b. Lower or raise brightness before performing histogram equalization. The brightness of the sample image below was first adjusted before performing adaptive histogram equalisation and the resulting image overcomes the shortcomings of the original algorithm.

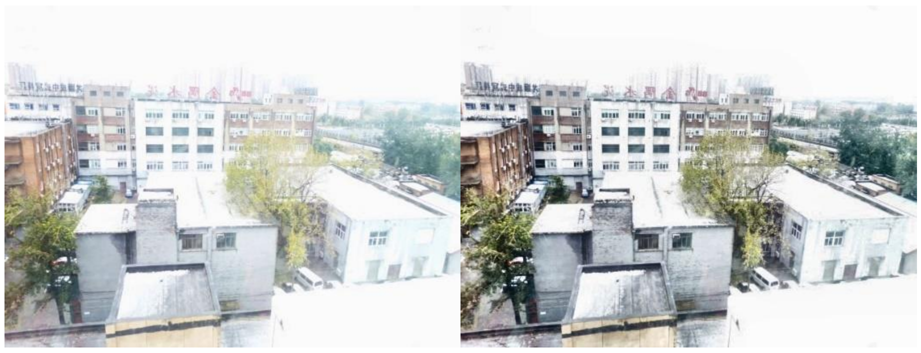

Figure 45 (a) Original Image after Brightness reduction (b) Image after Equalisation