Assignment I: Histogram Equalization

Li Hantao, G2101725H, MSAI, hli038@e.ntu.edu.sg

Histogram Equalization **(HE) is an image processing technique that has been widely adopted to improve the contrast in images. It accomplishes this by spreading out the most frequent intensity values, i.e., stretching out the intensity range of the image. It usually increases the global contrast of images when the image pixels fall within a narrow range of intensity values. This allows for areas of lower local contrast to gain a higher contrast.**

This assignment consists of the following tasks:

1) Implement the HE algorithm in Matlab or Python or other computer programming languages and apply your implemented HE algorithm to the 8 sample images. The submission of your solution should include your source-code algorithm implementation as well as the enhanced sample images by your implemented algorithm.

2) Discuss the pro and con of histogram equalization algorithm according to the enhanced sample images by your implemented HE algorithm. Discuss possible causes of some unsatisfactory contrast enhancement.

3) Discuss possible improvements of the basic HE algorithm. Implement and verify your ideas over the provided test images. This subtask is optional, and there will be bonus marks for good addressing of this subtask.

I. Introduction

Histogram equalization(HE) is an algorithm that adjusts the grayscale distribution of an image to make the pixel distribution within [0, 255] more balanced. It can improve the contrast of the image and the subjective visual effect.

Next, we will deduce the formula of histogram equalization. Assuming that the image to be processed is a grayscale image, where r represents its grayscale, with the value range [0, L−1]. Then the HE processes correspond to a transformation T:

$$s = T(r),\ 0 \le r \le L-1 \tag{1}$$

Assuming that $p(r)$ and $p(s)$ represent the probability density function of random variables r and s, respectively, where $p(r)$ is continuously differentiable and T are known. The probability density function of s can be obtained by the following formula:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \tag{2}$$

Through the geometric meaning of the HE algorithm on the histogram, we can construct the following formula:

$$s = T(r) = (L-1)\int_0^r p_r(w)dx \tag{3}$$

Then, substitute (3) into (2) to obtain:

$$\frac{ds}{dr} = \frac{dT(r)}{ds} = (L-1)\frac{d}{dr}\int_0^r p_r(w)dw = (L-1)p_r(r) \tag{4}$$

For the discrete form, we use probability histogram instead of a probability density function, summation instead of an integral operation, respectively. Here, we can obtain the discrete form of the formula:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \tag{5}$$

where MN is the total number of image pixels, n_k is the number of pixels with gray level r_k, L is the number of gray levels within the image. (L = 256 for an 8-bit image)

According to (5), the gray value of the pixel in the output image can be obtained by mapping r_k to s_k. The transformation (mapping) in (5) is called HE transformation. In this assignment, we built the HE function in Python through the above formula.

II. Procedures - Task I

For the deducing of the HE algorithm above, which is in the gray image domain, there are various choices to convert it to the color image. In this part, we will finish the implementation of HE with two steps. Firstly, we complete the source-code in Python, checking the correctness through the comparison with the cv2 library function '*cv2.equalizeHist()*'. Secondly, by analyzing the effect for different color models, such as RGB, HSV, YUV(YCbCr), we determine the specific implementation method of the HE algorithm for color images.

A. Histogram Equalization Functions

Before considering the color model, we first need to ascertain the accuracy of the implemented HE algorithm program. The source-code of the whole assignment will be given in the *Appendix.C* and the other file.

We chose the eighth picture in the sample set, with a relatively even distribution of light and dark pixels, as the reference for testing the accuracy. We test the HE algorithm on both the RGB color component channel and HSV brightness (Value) channel, respectively, to ensure comprehensiveness. The output images obtained through the HE function implemented and the cv2 library function are shown in Fig.1, with the histograms corresponding to each image.

From the results, we can get the following conclusions: the HE algorithm implemented in this assignment is the same as the HE function provided in the cv2 library, in terms of the averaging effect on the histogram and the visual outcome of the output image, indicating that the implemented HE function is valid and accurate.

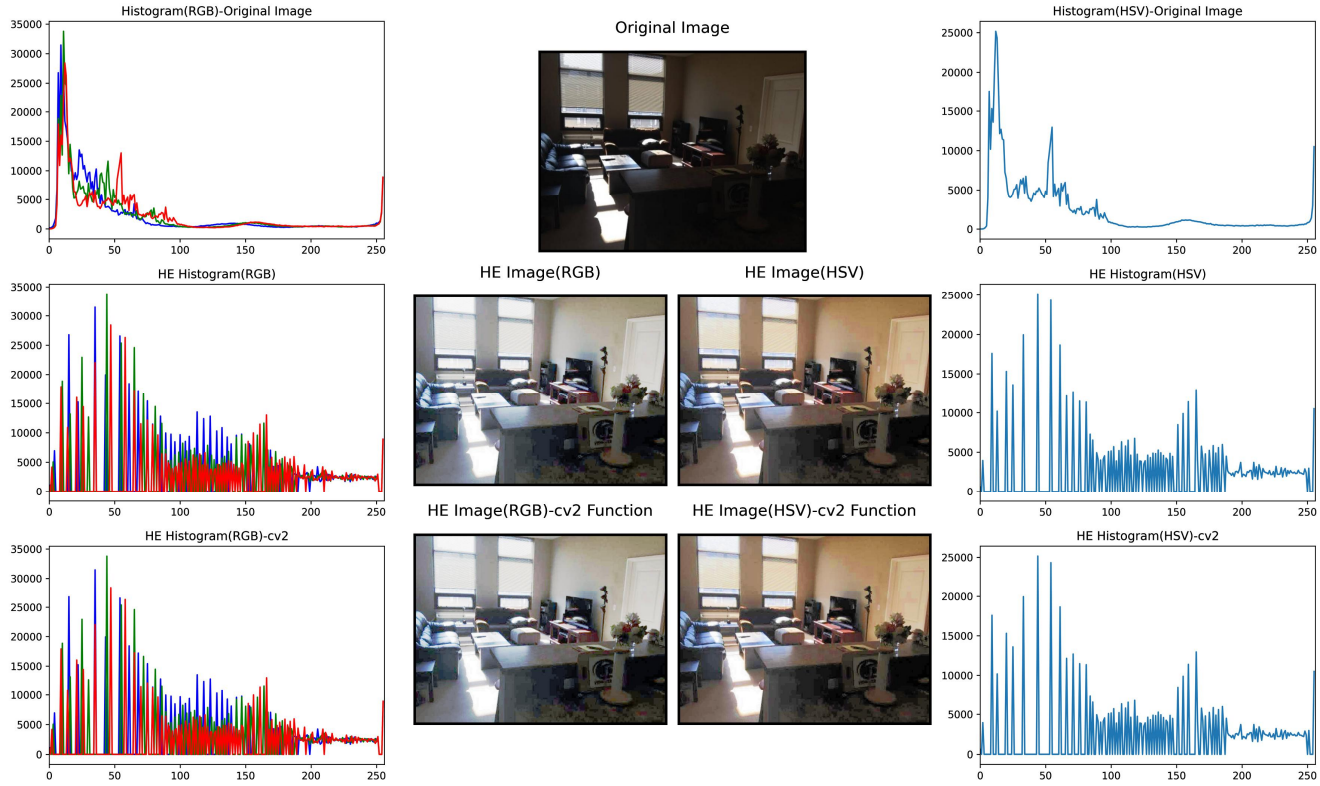


Fig. 1. Comparison of the implemented HE function and the standard cv2 HE function.

B. *Color Models*

It can be seen from the above comparison process that various color models for images (for above, it is RGB&HSV) can be used for their corresponding HE procedure. Nevertheless, which model can gather the best effect still requires specific comparative analysis. This section parallels the output images after HE under different color models and elects the most suitable color model for HE implementation.

We chose RGB, HSV, and YUV, where YCbCr of YUV was taken, three color models to compare their output results. RGB color model has an apparent meaning on the physical light combination, which is suitable for color tube working. However, each component of RGB participates in the calculation of image brightness, so it is not rational for the method of HE. Therefore, we continue to choose HSV and YCbCr, which obtaining the luminance channel separately: the V component in HSV is *max(R, G, B)*; and the calculation of the Y component in YCbCr is more intricate, which is specifically designed to integrate black-white and color representation by digital devices.

Three different color models of HE were performed on the eight photos given in the sample set in turn, and the outcomes are listed and compared in the *Appendix.A* for an entire display. We only extract a part of the result in this part for comparative analysis in Fig. 2.

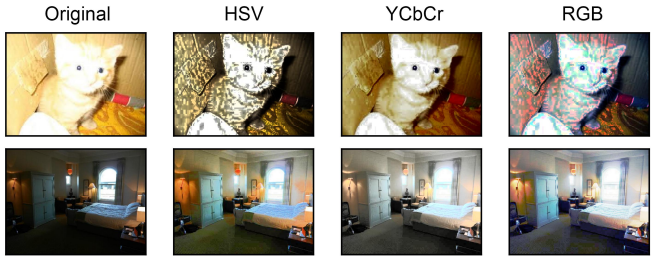


Fig. 2. Color Model Comparison (Partly).

We select an image with uneven light and dark distribution, that is, a cat picture with most of part being overexposed; and an image with relatively uniform light and dark distribution, that is, an indoor picture with both highlights and shadows, as the example, which carries out the corresponding analysis:

Firstly, we can find there is an evident and non-negligible problem when observing the RGB image: the hue distribution in the output image is incorrect in that them does not appear in the original one. The reason can be analyzed: three channels of RGB image jointly contribute to the coding of color information and brightness information. When we apply nonlinear mapping, HE, to the three channels respectively, we change the image color while adjusting the brightness. Modifying the color of an image is certainly not the goal of the HE algorithm; thus, we

will abandon the RGB color model.

Secondly, by inspecting the HSV output for the cat image, we can find obvious brightness cracks in the picture. We conjecture that the brightness information obtained simply by *max(R, G, B)* is discontinuous. The equalization of an image with a narrow histogram distribution will intensify this discontinuity by stretching the histogram, resulting in a poor output image effect. On the other hand, although there is no hue deviation, the output image's color saturation is not satisfactory. That is, for the images with narrow lightness distribution, the saturation in the output is significantly reduced, losing a lot of color information; for the picture with more uniform brightness, the middle part's saturation is improperly raised, such as the bed and wall in the second image, which makes the picture appear an unnatural state. After comparison, we conclude that the HE output of the YCbCr is desirable, especially there are no problems caused by RGB / HSV image in the above analysis. Consequently, we choose YCbCr as the image color coding model. The comparison between the results of HE and the original image is shown in Fig. 3.

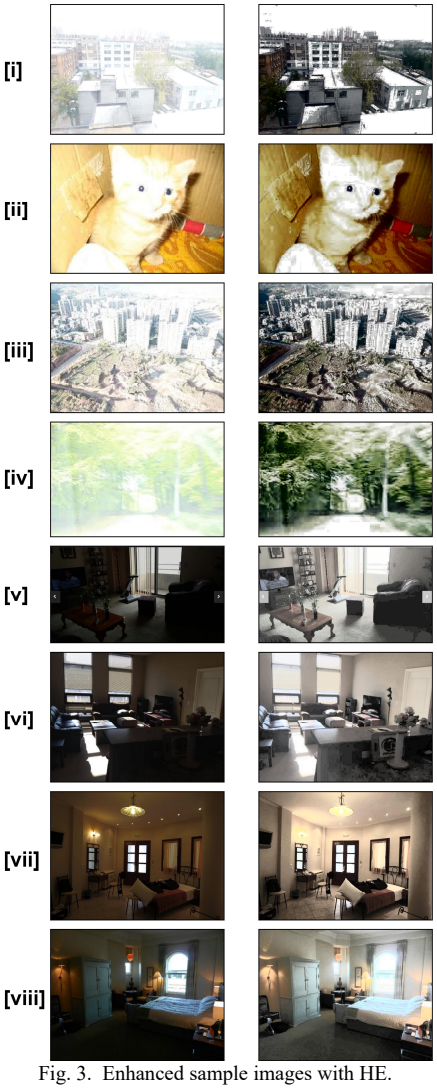


Fig. 3. Enhanced sample images with HE.

III. Procedures - Task II

A. Advantages

1) The algorithm is uncomplicated, and the mapping function of the equalization histogram is not changed facing distinct specific images, so the processing speed is fast.
2) HE is a reversible operation. If the equalization function is identified, the original histogram can be restored, and the complexity of the calculation is small.
3) For all kinds of images, the application significantly enhances the image contrast. Moreover, in the case that the original image is primarily bright, as shown in **[iii]**, or mostly dark, as presented in **[v]**, the outcome of the image will be greatly improved after the implementation of HE.

B. Disadvantages

1) This limitation corresponds to the first advantage: the mapping function does not have specificity, resulting in the algorithm incorrectly increasing the contrast of noise and eliminating the valuable information. For example, in **[vi]**, the noise is significantly enhanced while the data outside the window is lost.
2) HE algorithm directly stretches the histogram of all pixels, which will change the light-dark distribution of the image. Ideally, HE will equalize the average brightness of pixels to ***L/2***. In some circumstances, it will significantly modify the mean brightness. For instance, the gloomy room in **[vii]** becomes a bright room after HE. This difference will affect the perception of people and the further recognition by some algorithms.
3) After HE, the original details may disappear instead of being enhanced, which means the bright part will be brighter, while the dark part is going to be darker. For example, the information outside the window in **[viii]** becomes completely overexposed after HE, for most of the pixels in the original image are stacked in the dark part, and only a tiny portion of it is in the highlight. When stretching the pixels stacked in the left side to the middle and right side during HE, the highlight will be further compressed, lose the original data, and become entirely overexposed.

Fig. 4. Image cracks after HE.

4) In the original image, the adjacent areas with a similar luminance value may be stretched into spaces with huge brightness differences after HE. Such as the wall in **[i]** and the forehead in **[ii]**, which is pointed in Fig. 4, showing obvious image cracks. The gray level is a discrete value, here *L=256*, so the histogram obtained via HE in the actual

operation is not uniformly distributed but approximately uniformly distributed. If the original histogram has numerous pixels accumulated on a small scale, they may be relocated to the bin with a significant brightness difference during equalization.

IV. Procedures - Task III

A. Re-approach Average Luminance

For the weaknesses mentioned in the above analysis, we intuitively believe that most of them can be bypassed if the average luminance value is not considerably varied during HE. Firstly, it can directly solve the second disadvantage. Secondly, ensuring the HE does not transfer multiple pixels can also avoid the third and the fourth cons because their origin cause is the accumulation of a large number of pixels within a small scale on the histogram of the input image.

To maintain the average brightness, we primarily try to adjust the image curve to achieve the purpose. As shown in Fig. 5, the horizontal axis of the image curve represents the input brightness, while the vertical represents it after applying the new mapping. The initial curve is a proportional straight line (the black line). Assuming that the average luminance of the image is $\boldsymbol{m}$ before HE is implemented, we apply (a) or (b) diagram to the output of HE, which elect according to the relationship between m and $\boldsymbol{L/2}$. Mapping the blue curve in, the pixel with value $\boldsymbol{x_0}$ becomes $\boldsymbol{y_0}$.

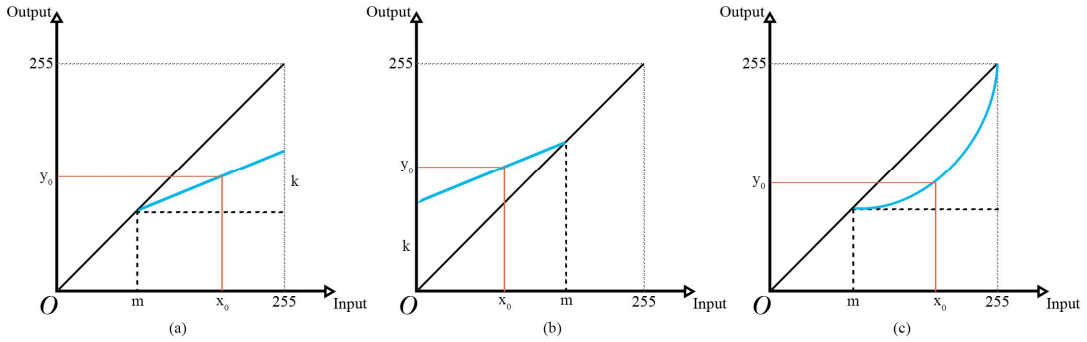


Fig. 5. Sketch curve of re-approach average luminance

Through simplistic geometric calculation, we can gain the expression of $\boldsymbol{k}$ and the blue curve, which can restore the average value to $\boldsymbol{m}$:

$$(a): \quad k = \frac{m^2}{m(L-1)}$$

$$y_0 = \frac{k(x_0-m)}{(L-m-1)} + m \tag{6}$$

$$(b): \quad k = 2L - \frac{L^2}{m}$$

$$y_0 = k + \frac{x_0(m-k)}{m} \tag{7}$$

Notwithstanding, after the application of the above algorithm, it is observed that this approach is erroneous. The resulting images are shown in Fig. 6. Although this algorithm restores the image's luminance, its negative impact is more significant than the positive side. It cannot improve other disadvantages, but the two curves with different slopes cause the separation in the image contrast, resulting in the loss of the contrast of part of pictures.

We still can further promote this method, such as applying the curve as Fig. 5(c), to restore brightness on the premise of maintaining the original contrast and brightness range; however, we think this idea is not worth time. The reasons are as follows: firstly, this method cannot resolve the contrast splitting between two image parts. Secondly, to enhance the effect of HE, we should consider the HE algorithm itself instead of starts with the output after HE is applied, similarly to this method.

Fig. 6. Results of re-approach average luminance.

B. Left-right HE Algorithms

Considering that approaching average luminance after HE implementation is not satisfying, we try to take steps before HE to preserve the luminance: the histogram of the original image can be divided into left and right parts with the segmentation line, the mean value m, and the left and right histograms can be equalized respectively. In this way, the calculus formula of HE will become as follows:

$$s_k = \begin{cases} \frac{(m-1)}{M_l N_l} \sum_{j=0}^{k} n_j, & k < m \\ \frac{(L-m-1)}{M_r N_r} \sum_{j=m}^{k} n_j, & k \geq m \end{cases} \tag{8}$$

where subscripts l and r represent the left and right sub-histograms, respectively.

In the program implementation, we only necessitate to slightly modify the original function by counting the probability distribution respectively for the pixels on both sides of m to obtain the new HE algorithm.

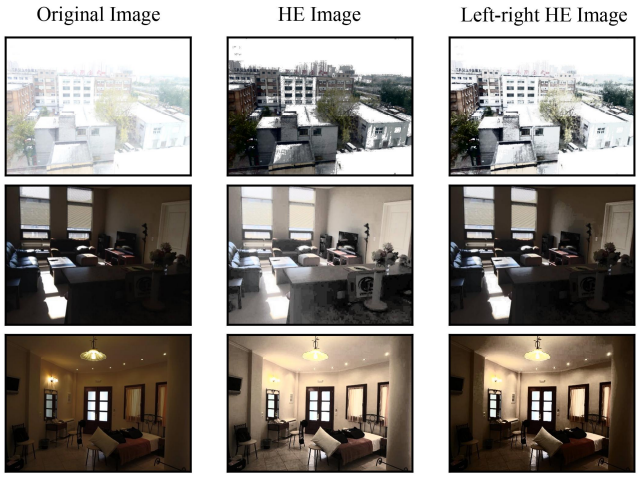


Fig. 7. Results of left-right HE (Partly).

All the images processed by the left-right HE are given in *Appendix.B*, and only some, shown in Fig. 7, are selected for analysis in this section. We can notice that the processed image retains the overall luminance, which is better than the original HE version. In addition, the third and fourth shortcomings have also been greatly enhanced. In the first picture, the original 'broken' wall is not torn in the new one; the highlight information outside the window initially lost in the second picture is also retained. Moreover, the new image also reduces noise that was previously heightened incorrectly by the HE.

It can be concluded that left-right HE has comprehensively solved the problems caused by HE; however, the image generated by the new HE is not as good as the original one in enhancing the contrast. For example, in the dim room, the room's contents cannot be clearly observed after left-right HE. We choose to maintain the overall brightness while reducing the scale where the original pixels can be equalized, or the histogram is stretched. The dark pixels that can be balanced to [0, L] can only be stretched in the left half histogram [0, ***m***], which is the inevitable drawback of this algorithm.

Consequently, we expect to improve the contrast of only the bright part (or shadow part) while maintaining the effect of the existing algorithm. The answer is also apparent: we cannot only consider the whole picture or a particular field. We should divide the image into multiple sub-images with the neighbor-area of each pixel.

C. CLAHE

For the image segmented, HE may be a suitable method, but dealing with the discontinuous gap between each sub-image is still an obstacle.

Through online literature viewing, we notice that some scholars have proposed the solution, namely contrast limited adaptive histogram equalization (CLAHE) [1].

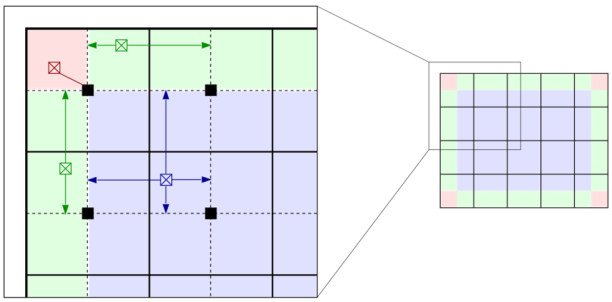


Fig. 8. Sketch of the CLAHE.

The image is partitioned into equally sized rectangular tiles, as shown in the right part of Fig. 8. The HE function applied in each sub-image is the same as the original HE. Pixels in the bulk of the image (shaded blue) are bilinearly interpolated, pixels close to the boundary (shaded green) are linearly interpolated, and pixels near corners (shaded red) are transformed with the transformation function of the corner tile. The interpolation coefficients reflect the location of pixels between the closest tile center pixels so that the result is continuous as the pixel approaches a tile center. Because the implementation of the CALHE function is relatively complex, we choose to call the function '*cv2.createCALHE()*' directly. The obtained results can be browsed in *Appendix.B*, while only part of the results are shown in Fig. 9 in this section.

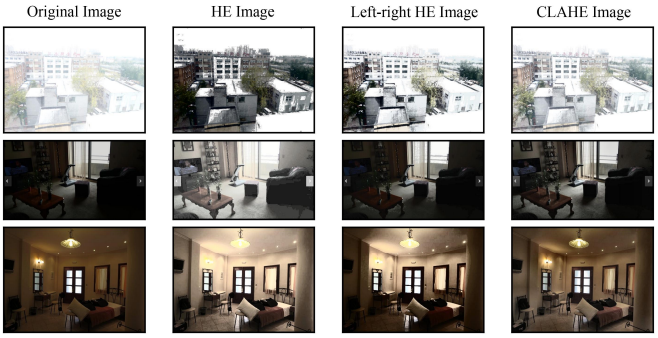


Fig. 9. Results of CLAHE (Partly).

It can be observed from the figure that the image output by the CLAHE is satisfying, which comprehensively solves the cons mentioned in Section 2. While maintaining the original detail information and average luminance, the output has no cracks of pixels with similar brightness, and the enhancement effect of image contrast is similar to the original HE algorithm.

V. Code Availability

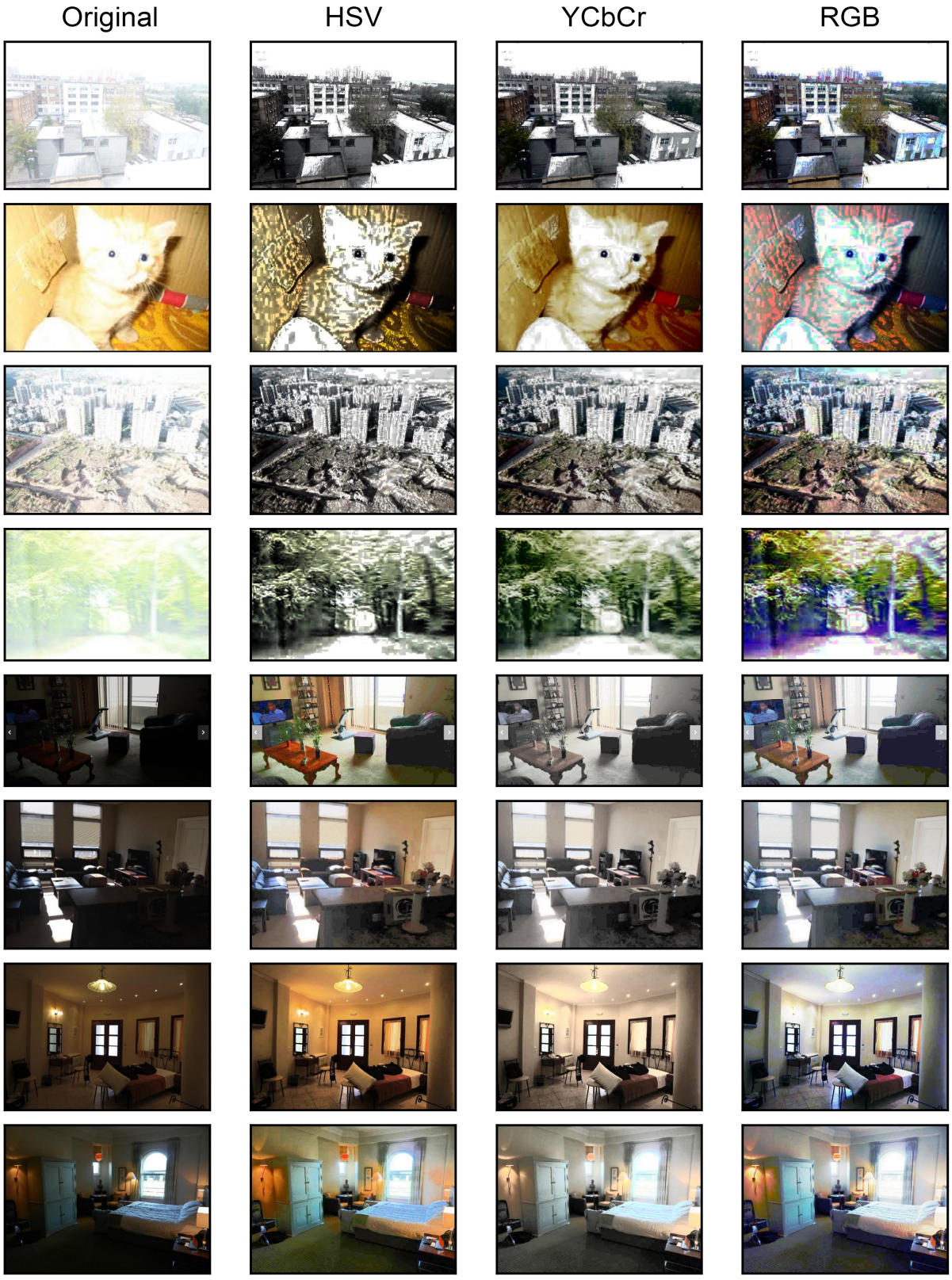
The main functions related to assignment are given in *Appendix.C*. For the file containing all the source-code, see another file in the compressed package.

References

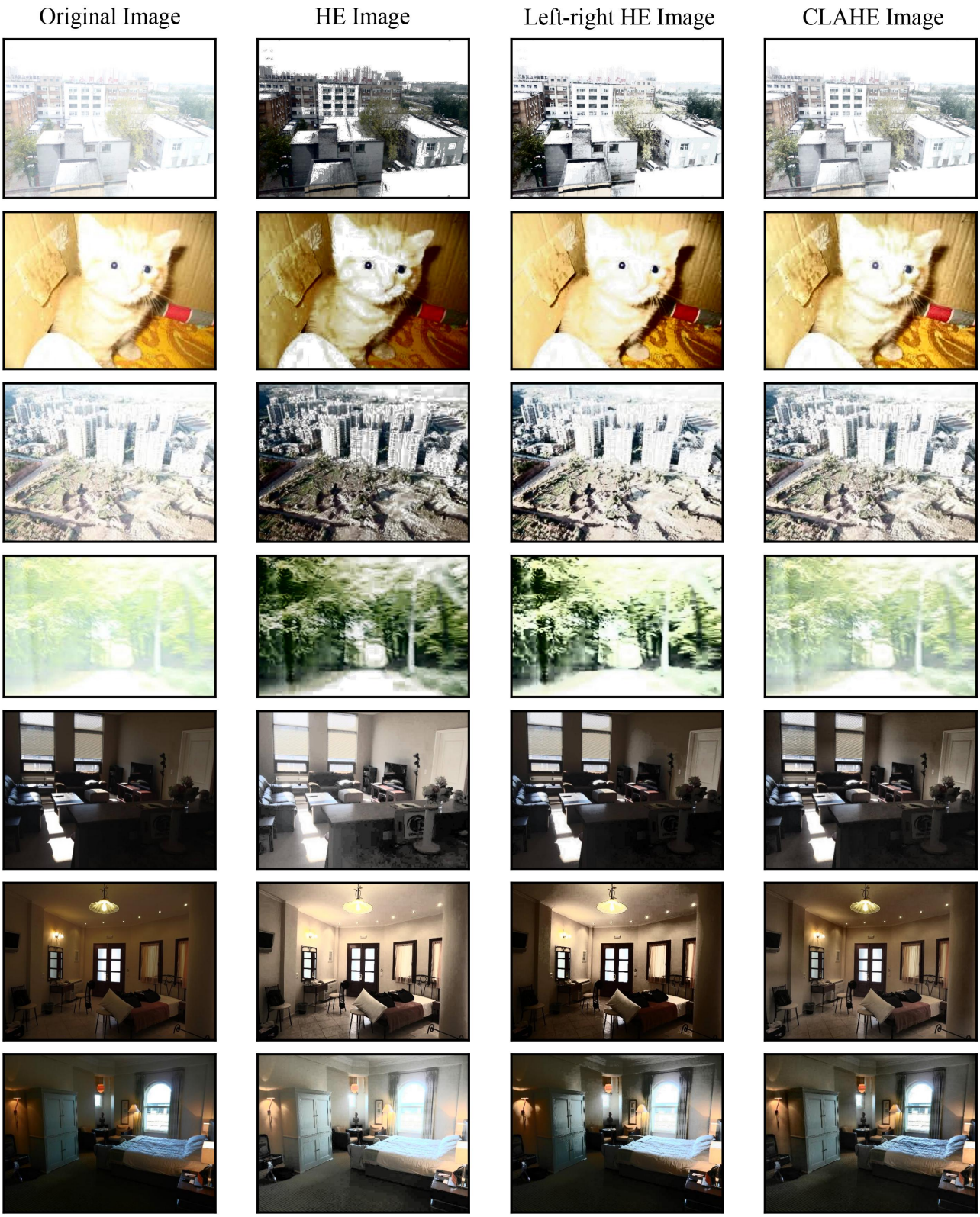
[1] Wikipedia, "Adaptive histogram equalization", [Online]. Available: https://en.wikipedia.org/wiki/Adaptive_histogram_equalization

VI. Appendix

A. Color Model Comparison



B. Improved Algorithm Comparison



C. Main function source-code

```
def CalHE(image, hist):
    r"""
    Calculate the Histogram Equalization for single input
image.
    Input are the single channel and its Histogram. Output is
image.
    """

    #Calculate the probability distribution function of the
histogram
    sum_hist = 0
    sum_p = [0] * 256
    for i in range(len(hist)):
        sum_hist = sum_hist + float(hist[i])
        sum_p[i] = sum_hist

    #Use the probability distribution to establish the Lookup
Table
    lut = np.zeros(256, dtype=image.dtype)
    for i, ele in enumerate(lut):
        lut[i] = int(255.0 * (sum_p[i] / sum_hist) + 0.5)

    #Use the Lookup Table to calculate the new image after HE
    image_result = lut[image]
  return image_result

def HistogramEqualization_RGB(image_b, image_g, image_r):
    r"""
    This function is for RGB image.
    Calculate the Histogram Equalization for R, G, B channel
image.
    Output are three images, in R, G, B channels.
    """

    hist_b, hist_g, hist_r = CalHist_RGB(image_b, image_g,
image_r)
    he_b = CalHE(image_b, hist_b)
    he_g = CalHE(image_g, hist_g)
    he_r = CalHE(image_r, hist_r)
    return he_b, he_g, he_r


def HistogramEqualization_HSV(image_v):
    r"""
    This function is for HSV/YUV image.
    Calculate the Histogram Equalization for V channel image.
    Output is the new V channel image.
    """

    hist_v = cv2.calcHist([image_v], [0], None, [256], [0,
256])
    he_v = CalHE(image_v, hist_v)
    return he_v

def HistogramEqualization_AdjustAvarage(image_v):
    r"""
    Re-adjust the output image with HE to the average
luminance
    """

    hist_v = cv2.calcHist([image_v], [0], None, [256], [0,
256])
    he_y = CalHE(image_v, hist_v)
    m = np.mean(image_v)
    new_y = he_y.copy()

    if m > 125.5:
        m = 125.5 + 0.5 * (m - 125.5)
        k = 510 - (255**2) / m
        for i, row in enumerate(he_y):
            for j, pixel in enumerate(row):
                if pixel < m:
                    new_y[i][j] = k + (((m - k) / k) * pixel)
    elif m < 125.5:
        m = m + 0.5 * (125.5 - m)
        for i, row in enumerate(he_y):
            for j, pixel in enumerate(row):
                if pixel > m:
                    new_y[i][j] = (((m / (255 - m))**2) *
(pixel - m)) + m
    return new_y

def LR_CalHE(image, hist):
    r"""
    Calculate the Histogram Equalization for single input
image.
    This function use the Left-right method to modify the
picture
    Input are the single channel and its Histogram. Output is
image.
    """

    #Calculate the probability distribution function of the
histogram
    m = int(np.mean(image) + 0.5)
    sum_hist_less = 0
    sum_hist_large = 0
    sum_p_less = [0] * 256
    sum_p_large = [0] * 256

    for i in range(len(hist)):
        if i <= m:
            sum_hist_less = sum_hist_less + float(hist[i])
            sum_p_less[i] = sum_hist_less
        else:
            sum_hist_large = sum_hist_large + float(hist[i])
            sum_p_large[i] = sum_hist_large

    #Use the probability distribution to establish the Lookup
Table
    lut = np.zeros(256, dtype=image.dtype)
    for i, ele in enumerate(lut):
        if i <= m:
            lut[i] = int((m-1) * (sum_p_less[i] /
sum_hist_less) + 0.5)
        else:
            lut[i] = int((255 - m) * (sum_p_large[i] /
sum_hist_large) + 0.5) + m
    image_result = lut[image]
    return image_result

def LR_HistogramEqualization_HSV(image_v):
    r"""
    This function is for HSV/YUV image.
    This function use the Left-right method to modify the
picture
    Calculate the Histogram Equalization for V/Y channel
image.
    Output is the new V channel image.
    """

    hist_v = cv2.calcHist([image_v], [0], None, [256], [0,
256])
    he_v = LR_CalHE(image_v, hist_v)
    return he_v
```