NANYANG TECHNOLOGICAL UNIVERSITY

AI6121 Assignment 1 Histogram Equalization

YU YUE, G2202151A
JU XILAI, G2202544B
LUO HAO, G2202279H

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2022

Contents

List of Figures

Chapter 1

Histogram Equalization

1.1 Introduction to HE

Histogram equalization [1] (HE) is a fundamental and classic method to increase the global contrast of the images, which widely used in medical imaging and other fields. For image contrast enhancement, a simple understanding is to find a mapping function, which can map the relatively concentrated gray distribution of the original image to another relatively dispersed gray space, so as to enhance image contrast. The idea of histogram equalization algorithm is to artificially set a scattered gray distribution range, and use the cumulative distribution function as the mapping standard (show as 1.1), so as to achieve global contrast enhancement. Experimental results show that this algorithm can effectively enhance image contrast with less computational cost, especially for those represented by a narrow range of intensity values.

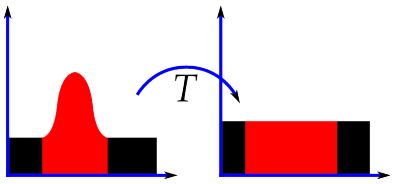


Figure 1.1: Histogram Equalization [1]

To map a narrow distribution to a wider distribution range, let's consider the cumulative distribution functions (CDF) of the grayscale distribution. For the grayscale level i, the probability of the occurrence is

$$p_x(i) = P(x = i) = \frac{n_i}{n}$$

The cumulative distribution function defined corresponding to the grayscale i is

$$CDF_x(i) = \sum_{j=0}^{i} p_x(j)$$

According to the algorithm of histogram equalization, we map the grayscale i to i' so that

$$i' = CDF_x(i) * (max\{y\} - min\{y\}) + min\{y\}$$

Here, x are those pixels in the result. Typically, the range of the grayscale is $[0, 255]$. Then, $max\{y\} = 255$ and $min\{y\} = 0$. Because grayscale level are integers, so we just take the integer part. Therefore, in our experiments, we map grayscale i as

$$i' = \lfloor 255 * CDF_x(i) \rfloor$$

The algorithm 1 is the pseudo code of HE.

Algorithm 1 image processing of Histogram Equalization

Input: The original image X
Output: The image Y after histogram equalization

```
cdf ← []
for i = 0 to 255 do
    cdf[i] ← Σ_{j=0}^{i} p_x(j)
end for
for each pixel i in X do
    pixel i' ← cdf[i'] in Y
end for
return Y
```

1.2 Color Spaces

Histogram equalization algorithm is used to enhance the contrast of grayscale images. However, if we apply it to RGB three-channel color maps, the performance would be unsatisfactory if we just apply HE on each of the channel. The Figure 1.2 shows the results on the testing images. As we can see from the figure, because we do histogram equalization on the RGB three channels respectively, the color of the original image has been drastically altered. Theoretically speaking, what we do is not contrast enhancement, but distribution equalization in the three-color channel, so that the relative proportion of RGB three-channel is changed to different degrees, thus changing the color of the original image. To solve the problem, we turned to the color space with the channel of lightness contrast (or luminance).

In fact, a lot of existing color space can satisfy our demand for lightness of the images. HSV [3] (for hue, saturation and value) is an alternative representation with brightness value. The LAB color space [4] (L for perceptual lightness, and a and b for the four unique colors of human vision: red, green, blue, and yellow) also has lightness parameter. Similarly, YUV model [5] defines one luminance component (Y) and two chrominance components(U for blue projection and V for red projection). In our experiments, we tried these three color space models (later CLAHE), and HSV color space is our main focus for basic histogram equalization.

We convert RGB images to HSV images first, and then we apply histogram equalization on the channel V(value). The range of channel V is $[0,1]$ and it's continuous, but the lightness level for RGB images is discrete. Therefore,the more lightness levels, the better performance. For better performance, we tried to keep the luminance distribution as close to continuous as possible. In our experiments, we set the number of lightness levels as 1000.

1.3 Overall Performance

The overall performance of the histogram equalization algorithm is shown in Figure 1.3. The code can be found on github

1.4 Analysis of the Performance

Obviously, by looking at these images, we can generally observe a significant increase in contrast. One of the main advantages is that it is a fairly straightforward technique and is reversible. If the equalization function is known, the original histogram can be recovered without much computational effort.

However, as we can see from the results, after the transformation, the lightness level of the image decreases and some details disappear, especially for those images with much pixels that have closer lightness levels(b, h in figure 1.3). That's because those pixels with closer lightness levels may be mapped to one lightness level, and details decreases.

Another disadvantage is that it is not selective about the data it processes. It may increase the contrast of background noise and decrease the contrast of useful signals. As we can see from the images b, d, f and h in the figure 1.3, the noise in the image is also amplified by HE.

Besides, When the image contains areas that are significantly brighter or darker than most of the image, the contrast in these areas will not be sufficiently enhanced.

In a word, the advantages and disadvantages of basic histogram equalization on HSV three channels are as follows:

Advantages:

- Simple and easy to implement. Relatively efficient contrast enhancement can be achieved with low complexity
- Invertible operation. The original histogram can be recovered if the equalization function is known

Disadvantages:

- Not selective while processing. It may increase the contrast of background noise and decrease the contrast of useful signals
- Lightness level of the image and details may disappear.
- Performance really depends on the original images. The contrast in those significantly brighter or darker areas will not be sufficiently enhanced. [2]

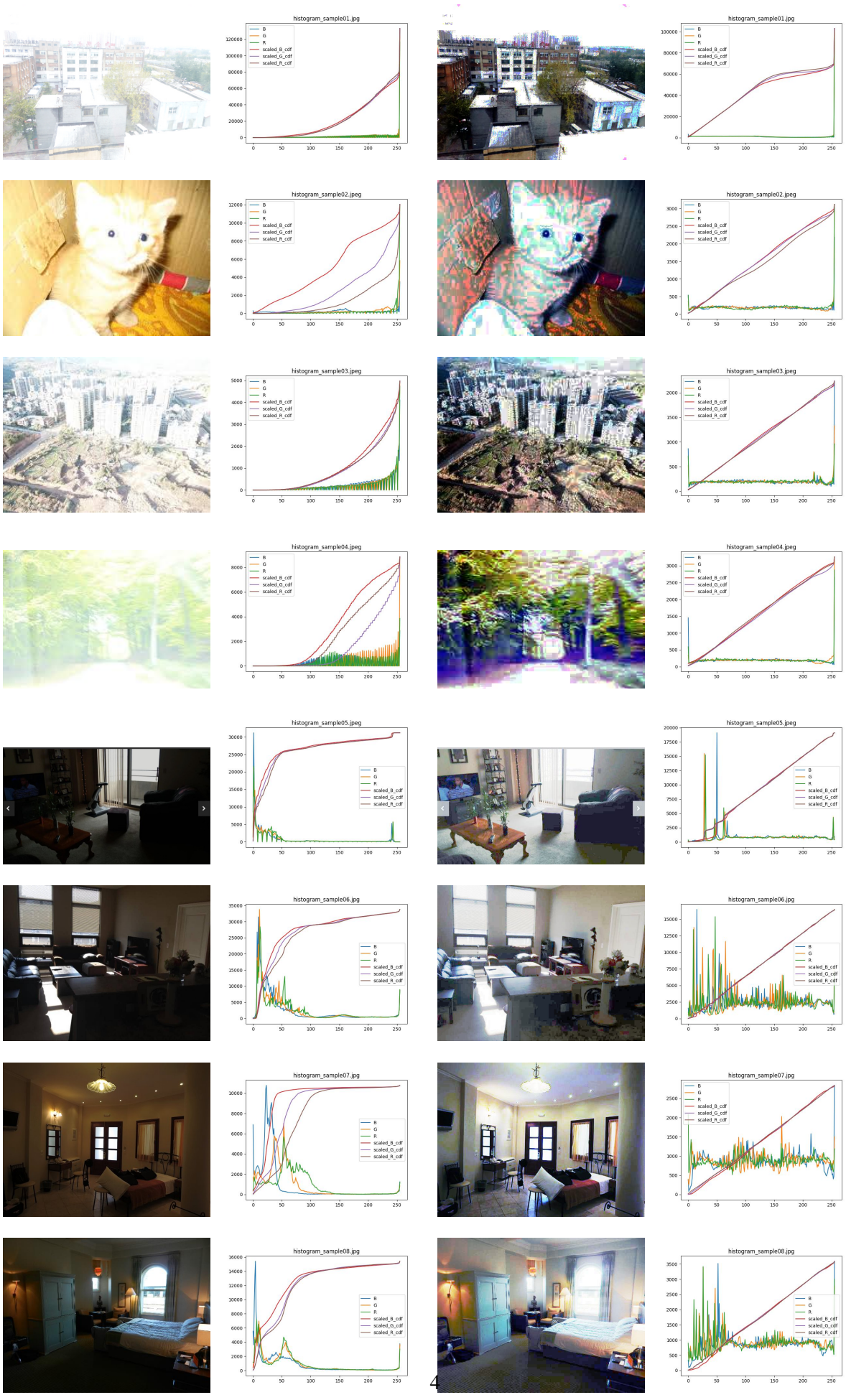


Figure 1.2: RGB distribution before/after HE on RGB

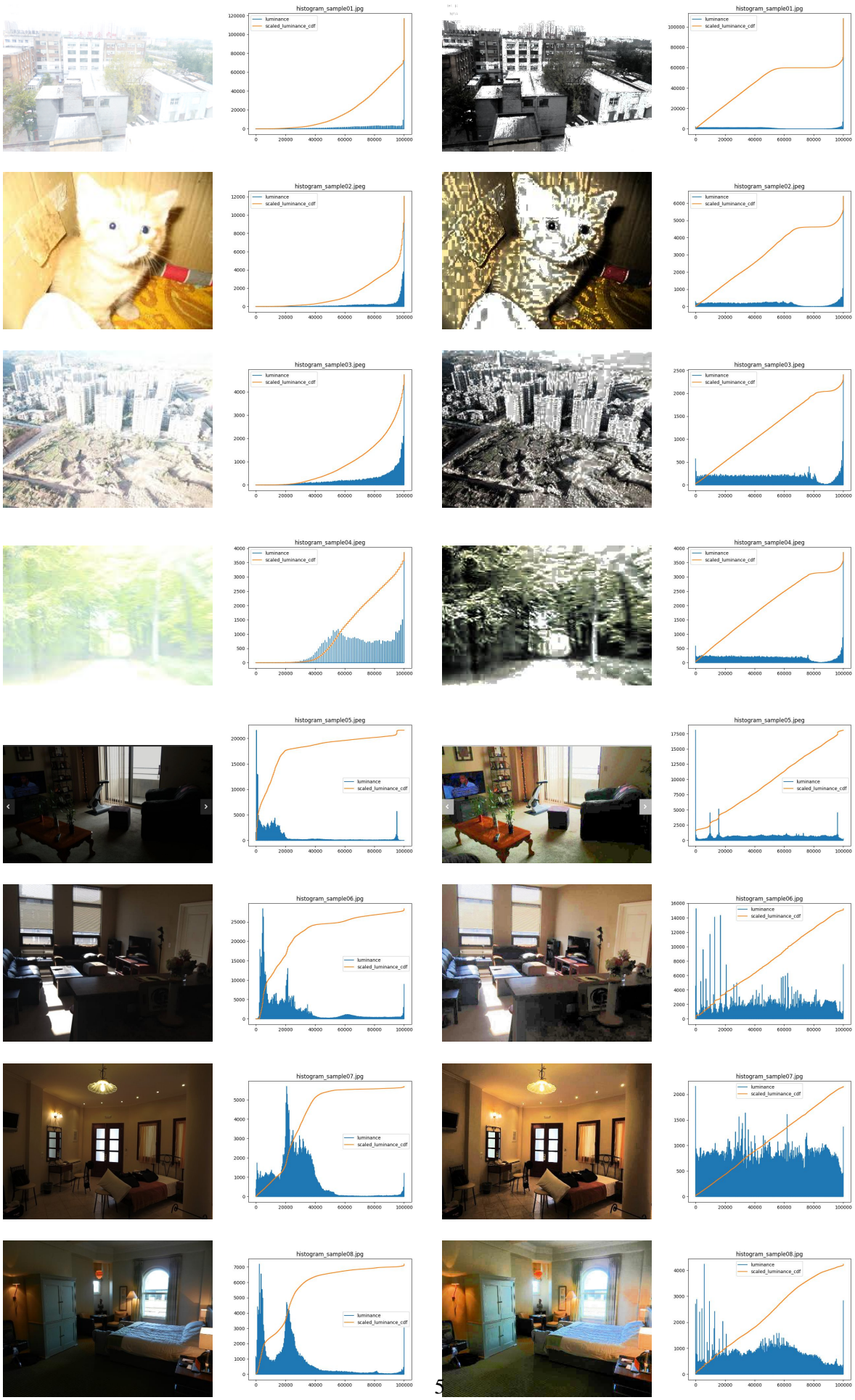


Figure 1.3: Luminance distribution before/after HE on HSV

Chapter 2

Adaptive Histogram Equalization

2.1 Introduction to AHE

Adaptive Histogram Equalization [2] (AHE) is an improved version of original Histogram equalization. As is mentioned, histogram equalization applies a transformation to the pixels of an image, which is derived from the histogram of the entire image. This method is simple, but ignores different areas of the image that vary greatly in light and dark. The contrast in these parts will not be significantly enhanced. The feature of AHE is that it selects a region of the image instead of the entire image for histogram equalization. For each pixel, the pixels in the neighbour square area are selected, and the histogram of these pixels is transformed to the value of the central pixel(show as 2.1).

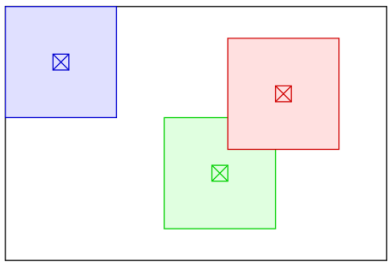


Figure 2.1: Adaptive Histogram Equalization [2]

Similar to HE, AHE just do HE for each pixel according to the distribution of a patch of the image. The algorithm 2 is the pseudo code of HE.

2.2 Overall Performance

The overall performance of the adaptive histogram equalization algorithm is shown in Figure 2.2. The code can be found on github

2.3 Analysis of the Performance

The testing was far from ideal. In the figure 2.1, we choose the patch size as $(8, 8)$. Obviously, the AHE algorithm increases the contrast in the image, but it is not difficult to see that the increased contrast is too strong, especially lines and noise in the images.As a result, the image is distorted and looks more like an image transformed into an oil painting. We analyzed the reason and we believe it may due to unreasonable patch size, so as a comparison, we also tried the influence of different patch sizes on contrast enhancement.

Algorithm 2 image processing of Adaptive Histogram Equalization

Input: The original image X, patch size $h * w$, output distribution range $[range_{min}, range_{max}]$
Output: The image Y after histogram equalization

for each pixel x_i in X **do**
    find the patch P with centre x_i
    $n \leftarrow 0$
    for p_i in P **do**
        if $p_i \leq x_i$ **then**
            $n+ = 1$
        end if
    end for
    for corresponding pixel y_i in Y **do**
        $y_i \leftarrow \frac{n}{h*w} * (range_{max} - range_{min}) + range_{min}$ in Y
    end for
end for
return Y

The figure 2.3 shows the different. As we can observe, appropriately increasing patch size can relatively improve the side effect of "Mosaic". However, increasing the patch size means increasing the algorithm overhead, because CDF in a larger area needs to be calculated for each pixel. On the other hand, for images of different sizes and scales, the optimal patch size is also different.

With a reasonable patch size, AHE is suitable for improving local contrast and enhancing edge sharpness in each area of the images. But on the other hand, AHE tends to over-amplify noise in relatively uniform areas of the images. When the image region containing the pixel neighborhood has relatively uniform intensity, its histogram will be strongly peakized, and the transformation function will map a narrow range of pixel values to the entire range of the resulting image. [2]

In a word, the advantages and disadvantages of adaptive histogram equalization on HSV three channels are as follows:

Advantages:

- Suitable for improving local contrast and enhancing edge sharpness in each area of the image [2]
- The patch size can be adjusted to fit images with different scales

Disadvantages:

- AHE tends to over-amplify noise in relatively uniform areas of the images [2]
- The algorithm has high time overhead, especially when the patch size becomes larger

To overcome the problem that adaptive histograms may enhance noise, a number of AHE variants [6] have also been proposed. In our experiment, we're going to talk about Contrast Limited Adaptive Histogram Equalization (CLAHE).

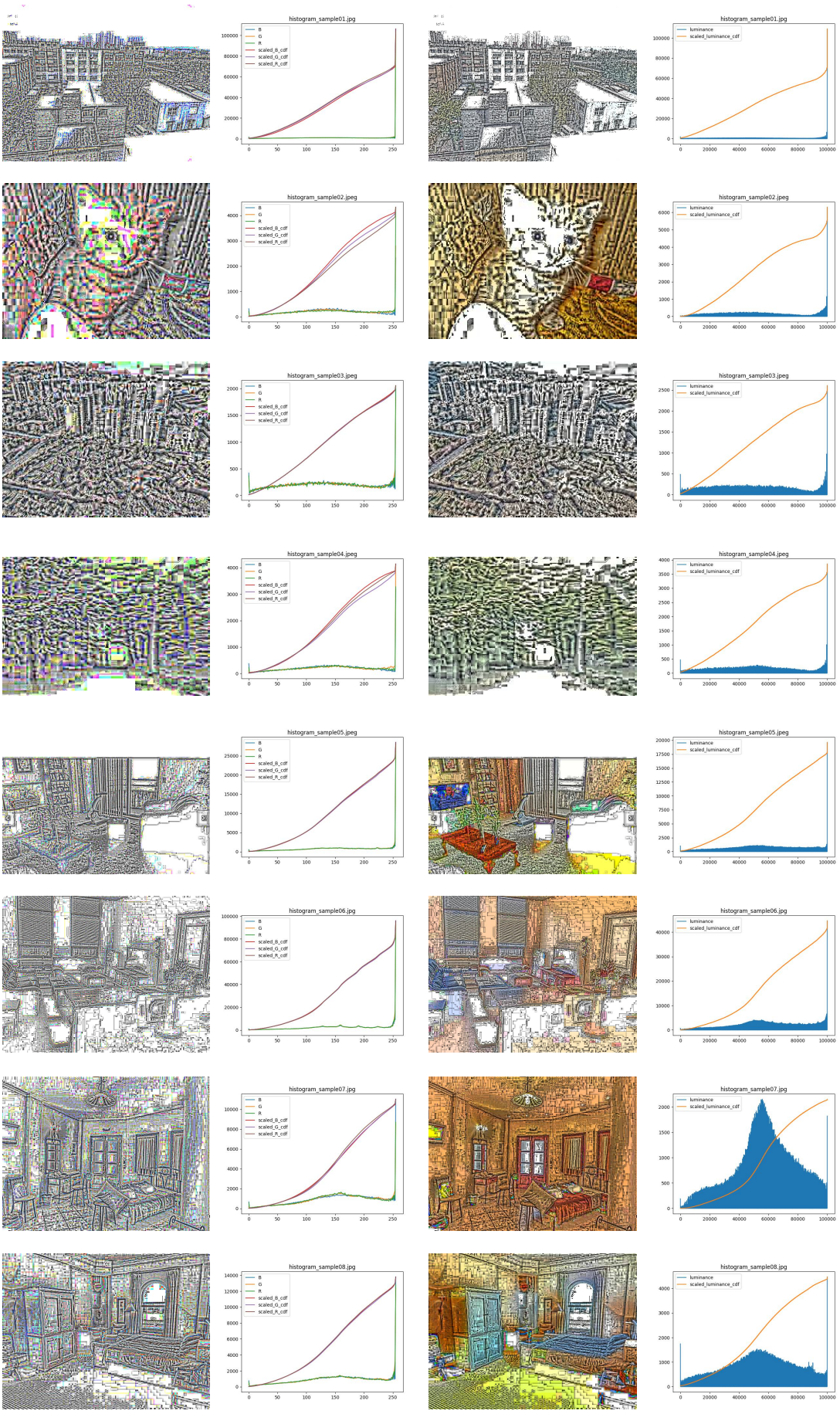


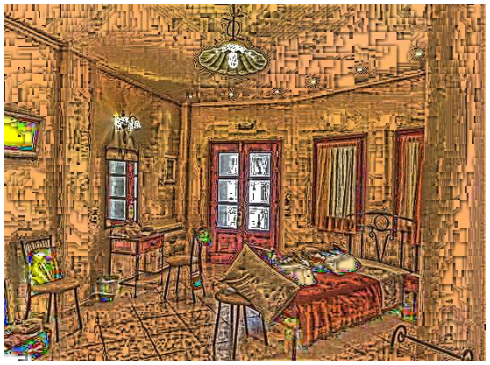
Figure 2.2: performance of AHE on RGB (left column) and HSV (right column)

(a) image01 after AHE with patch size (8*8)

(b) image01 after AHE with patch size (16*16)

(c) image01 after AHE with patch size (32*32)

(d) image01 after AHE with patch size (64*64)



(e) image07 after AHE with patch size (8*8)

(f) image07 after AHE with patch size (16*16)

(g) image07 after AHE with patch size (32*32)

(h) image07 after AHE with patch size (64*64)

Figure 2.3: performance of AHE on HSV with different patch sizes

Chapter 3

Contrast Limited Adaptive Histogram Equalization

3.1 Introduction to CLAHE

Considering that the performance of AHE will be affected by the noise problems, which means that sometimes a high peak will be discovered in the histogram, Contrast Limited Adaptive Histogram Equalization [2] (CLAHE) can reduce the noise problem by setting the cliplimit allowing only a maximum number of pixels in each of bins associated with local histograms [7].

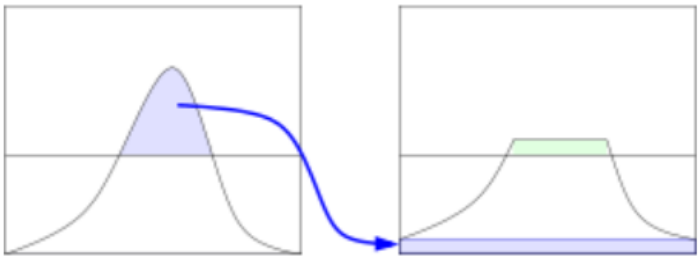


Figure 3.1: Redistribution with cliplimit [2]

The pseudo code of CLAHE is as follows:

Algorithm 3 Contrast Limited Adaptive Histogram Equalization

Input: The subfigure of AHE, $ClipLimit$, the size of subfigure $h * w$, HistSize;
Output: The subfigure after contrast limit;

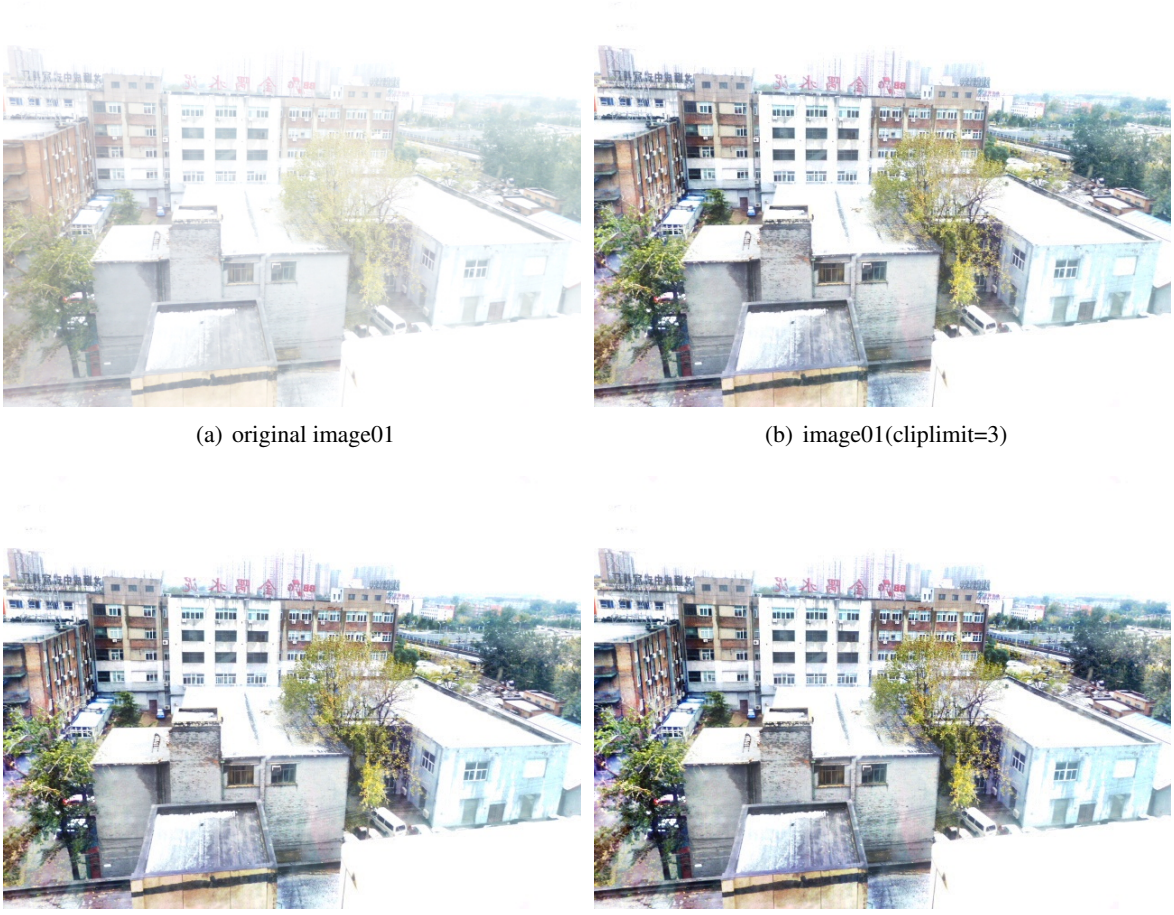
```
Calculating the histgram of the original subfigure S;
clipped ← 0
for i in HistSize do
    if S_i > ClipLimit then
        clipped ← clipped + S_i − ClipLimit
        S_i ← ClipLimit
    end if
end for
Distributing the clipped pixels evenly on the histgram
Use the redistributed histgram as the new histgram for HE
return The subfigure after contrast limit;
```

We add a cliplimit section on the AHE algorithm of chapter2, redistributing the pixels evenly in each of bins of the histogram. The cliplimit should be large enough to ensure that there will be spaces for

distributed pixels.

3.2 Analysis of the Performance

The performance of CLAHE is determined by the parameter cliplimit. We set the value of cliplimit 3,4,5 as follows:



(a) original image01 (b) image01(cliplimit=3)

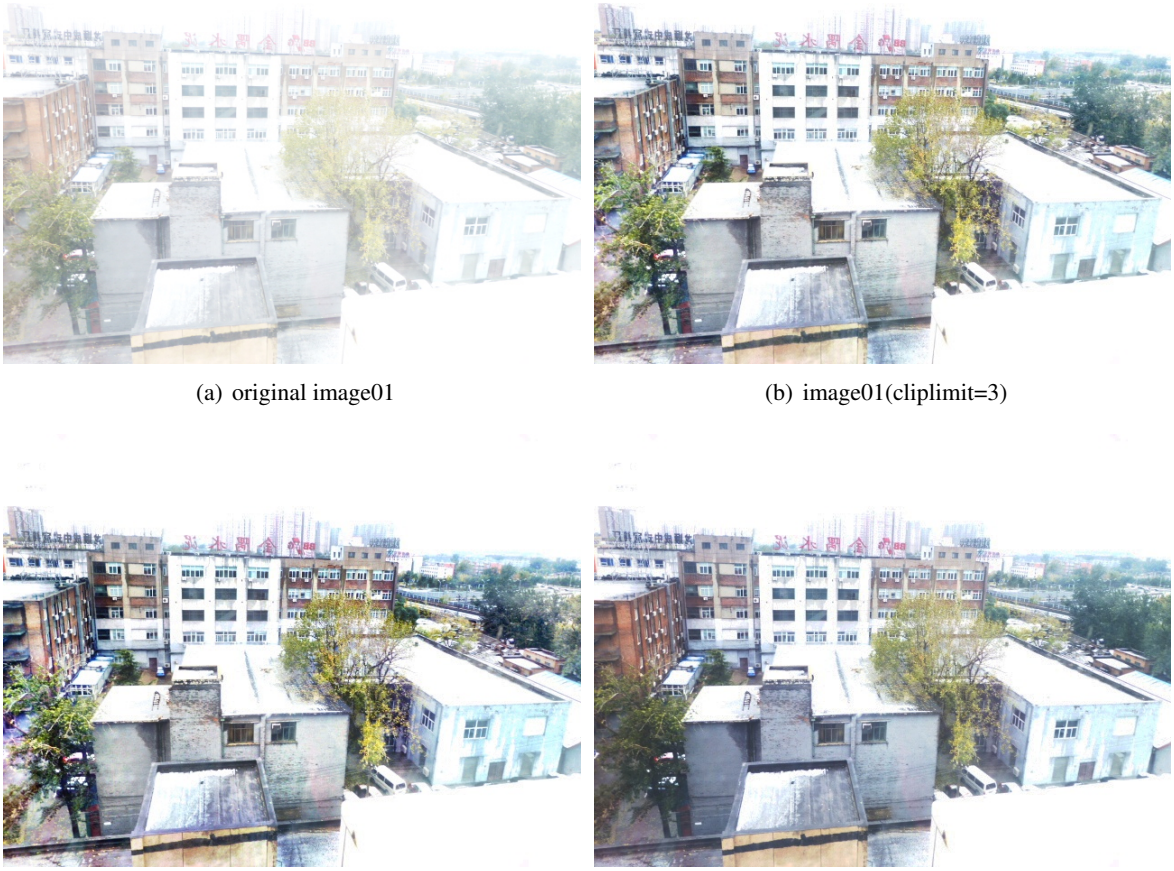
(c) image01(cliplimit=4) (d) image01(cliplimit=5)

Figure 3.2: performance of CLAHE on RGB

The contrast ratio increases compared with the original image, and less noise points appear due to the cliplimit. However, we could still find overexposure in some local areas even in cliplimit=3 image. Similar situations also appear in other 7 images, because of same problems among these images: too dark or bright not in local areas, but of the whole image.

For these images, CLAHE is not a good decision due to the AHE section. AHE will perform well when the contrast ratio in some local areas needs to be raised, but in the overall situation, it could bring side effects. AHE stretches the pixels of histgram in each subfigure, but in these samples, we need to consider the global contrast ratio. So we try to drop the AHE section, and only use the cliplimit to reduce the noise points of the images(CLHE).

In a word, when we are setting the cliplimit of CLAHE, there is a trade-off between local over-exposure and global performance. The contrast ratio in the bottom leftland side of sample01 suffers when raising the global contrast ratio, but with CLHE, we can achieve the balance easily with global considerations.



(a) original image01 (b) image01(cliplimit=3)

(c) image01(cliplimit=5) (d) image01 after CLHE

Figure 3.3: comparison between CLAHE and CLHE on RGB

Besides, when we decrease the cliplimit when doing both CHLE and CLAHE on RGB channel, which means more pixels in peaks are redistributed, the color of the picture is also distorted due to incongruous RGB three-color ratio (as show in figure 3.4). Therefore, we also do the experiments of CLHE and CLAHE on the HSV color space.

3.3 Overall Performance

(a) original image02

(b) image02 after clhe when cliplimit = imgsize*0.002

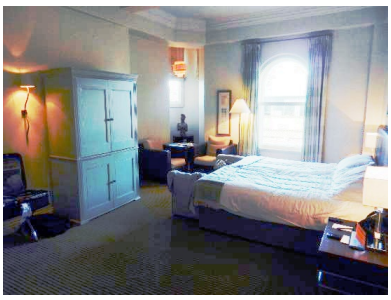
(c) image02 after clahe when cliplimit = 10 in opencv

(d) original image04

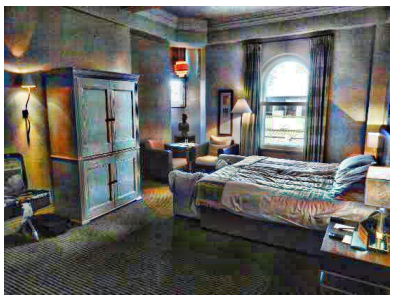
(e) image04 after clhe when cliplimit = imgsize*0.002

(f) image04 after clahe when cliplimit = 10 in opencv

(g) original image08



(h) image08 after clhe when cliplimit = imgsize*0.002



(i) image04 after clahe when cliplimit = 10 in opencv

Figure 3.4: Color disortion

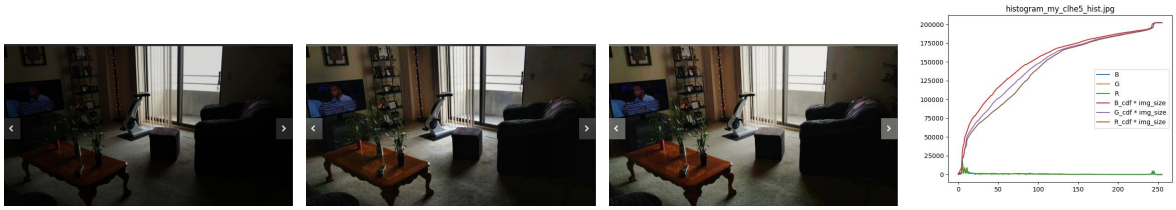
(a) original image01 (b) image01 after CLAHE (c) image01 after CLHE (d) image01 CLHE histgram

(e) original image02 (f) image02 after CLAHE (g) image02 after CLHE (h) image02 CLHE histgram

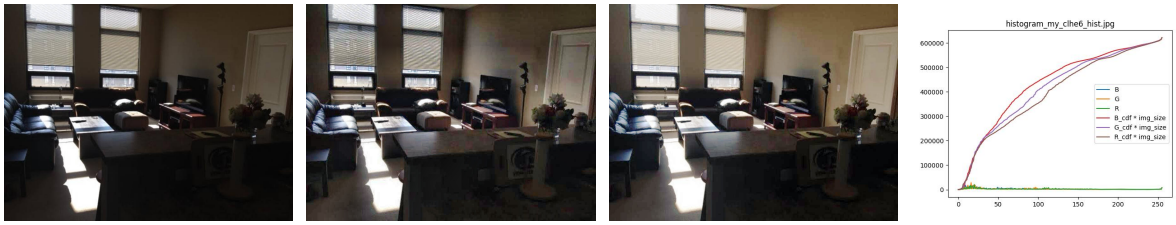
(i) original image03 (j) image03 after CLAHE (k) image03 after CLHE (l) image03 CLHE histgram

(m) original image04 (n) image04 after CLAHE (o) image04 after CLHE (p) image04 CLHE histgram

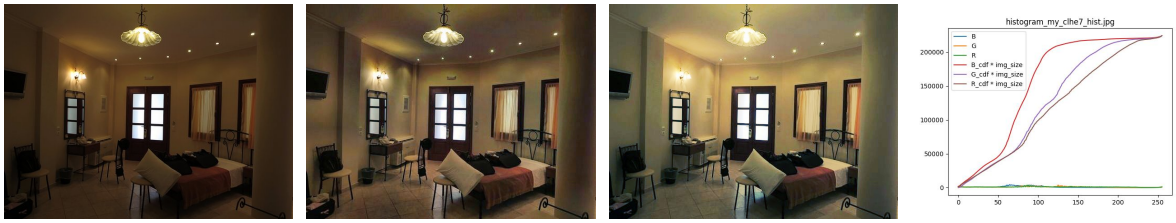
Figure 3.5: performance of CLAHE and CLHE on RGB in image01 to 04



(a) original image05 (b) image05 after CLAHE (c) image05 after CLHE (d) image05 CLHE histgram



(e) original image06 (f) image06 after CLAHE (g) image06 after CLHE (h) image06 CLHE histgram



(i) original image07 (j) image07 after CLAHE (k) image07 after CLHE (l) image07 CLHE histgram

(m) original image08 (n) image08 after CLAHE (o) image08 after CLHE (p) image08 CLHE histgram

Figure 3.6: performance of CLAHE and CLHE on RGB in image05 to 08

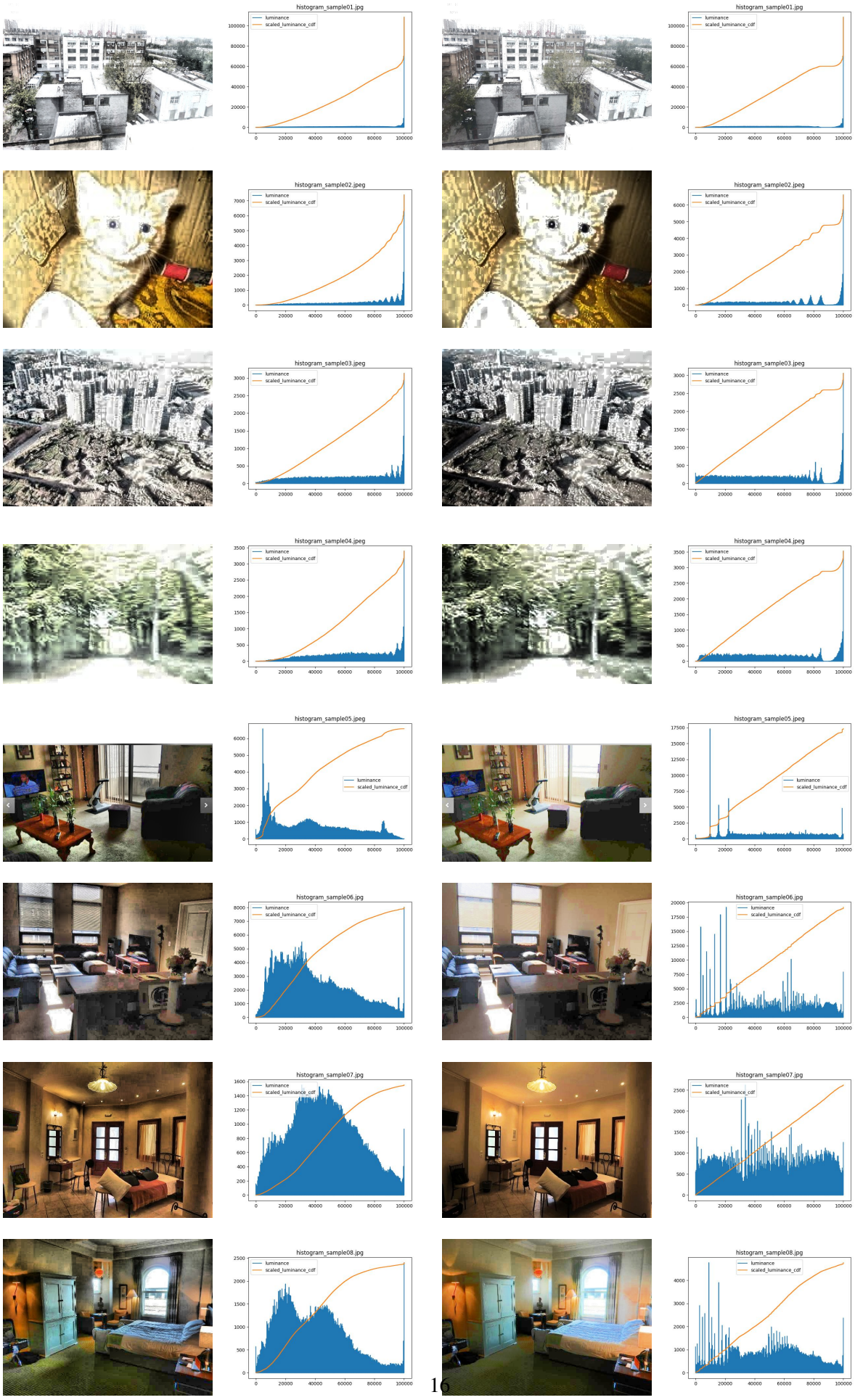


Figure 3.7: Performance of CLAHE (left) and CLHE(right) on HSV

Chapter 4

Conclusion

In our AI6121 assignment 1, we implement basic Histogram Equalization (HE) algorithm, Adaptive Histogram Equalization (AHE) algorithm and Contrast Limited Histogram Equalization (CLHE) by ourselves. We also explore different color spaces and do the experiments for comparing the performance on different color spaces by different algorithm. Besides, We also do the experiments on HE and CLAHE built-in function in python-openCV package, and tried the algoritthms on RGB, HSV, LAB and YUV color spaces. The results are similar to what we implemented. All the programming code and the experiment results can be found on github.

Our conclusion is that HE and AHE algorithm if applied in RGB three channels, will cause a certain color distortion problem, so the solution is to convert RGB images to the color spaces with lightness or luminance channel and then apply algorithms on that channel only. As is discussed, each algorithm has its own advantages and disadvantages, and different images may have different performance corresponding to different parameters.

In summary, CLAHE and CLHE are relatively better than others. However, considering the samples given, AHE section could lead to the side effect, so CLHE performs better by avoiding the trade-off between local overexposure and global performance.

Bibliography

[1] Wikipedia contributors. Histogram equalization — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-September-2022].

[2] Wikipedia contributors. Adaptive histogram equalization — Wikipedia, the free encyclopedia, 2021. [Online; accessed 12-September-2022].

[3] Wikipedia contributors. Hsl and hsv — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-September-2022].

[4] Wikipedia contributors. Cielab color space — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-September-2022].

[5] Wikipedia contributors. Yuv — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-September-2022].

[6] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

[7] Etta D Pisano, Shuquan Zong, Bradley M Hemminger, Marla DeLuca, R Eugene Johnston, Keith Muller, M Patricia Braeuning, and Stephen M Pizer. Contrast limited adaptive histogram equalization image processing to improve the detection of simulated spiculations in dense mammograms. *Journal of Digital imaging*, 11(4):193–200, 1998.

Appendix

Here are the functions we implemented in our programs:

histogram_equalization.py

```
# histogram\_equalization.py

def rgb_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # output: image after histogram equalization on RGB color space

def rgb2hsv(r, g, b):
    # input: r, g, b value of a pixel
    # output: h, s, v value of the pixel

def hsv2rgb(h, s, v):
    # input: h, s, v value of a pixel
    # output: r, g, b value of the pixel

def hsv_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # output: image after histogram equalization on HSV color space
```

adaptive_histogram_equalization.py

```
# adaptive_histogram_equalization.py

def rgb_AHE(img_name, img_path, tileGridSize = (8, 8)):
    # input: image name, image path and tileGridSize
    # output: image after adaptive histogram equalization on RGB color space

def rgb2hsv_AHE(img_name, img_path, tileGridSize = (8, 8)):
    # input: image name, image path and tileGridSize
    # output: image after adaptive histogram equalization on HSV color space
```

my_clhe.py

```
# my_clhe.py

def my_clhe(img, cliplimit, pixel_range):
    # input: one image channel, cliplimit and pixel_range
    # output: image channel after contrast limited histogram equalization
```

histogram_equalization_opencv.py

```
# histogram_equalization_opencv.py
```

```
def bgr_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after histogram equalization on BGR color space

def bgr2yuv_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after histogram equalization on YUV color space

def bgr2hsv_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after histogram equalization on HSV color space

def bgr2lab_histogram_equalization(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after histogram equalization on LAB color space

def bgr_CLAHE(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after CLAHE on BGR color space

def bgr2yuv_CLAHE(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after CLAHE on YUV color space

def bgr2hsv_CLAHE(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after CLAHE on HSV color space

def bgr2lab_CLAHE(img_name, img_path):
    # input: image name and image path
    # use python-opencv
    # output: image after CLAHE on LAB color space
```

plot.py

```
# plot.py

def plot_rgb_distribution(img_name, img_path, save_path):
    # input: image name, image path and fath for saving result
    # output: RGB distribution of the image

def plot_hsv_distribution(img_name, img_path, save_path):
    # input: image name, image path and fath for saving result
    # output: V channel distribution of the image
```