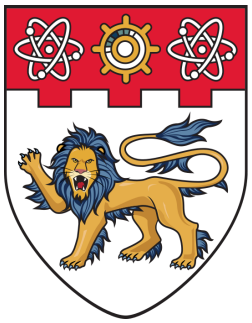


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

College of Computing and Data Science

Master of Science in Artificial Intelligence

AI6121 Computer Vision

Assignment 1 Report

Tan Jia Wei (Matric No.: G2403382B)

Tiffany Tan Ge Ru (Matric No.: G2404908D)

Fung Kwok Pong (Matric No.: G2405595A)

Zhou Chunxi (Matric No.: G2405212J)

Jin Zitong (Matric No.: G2403513F)

This document consists of **23** pages, including the cover page.

Contents

1 The Basic Histogram Equalization Algorithm

In this chapter, we will briefly introduce the basic Histogram Equalization (HE) algorithm, explain its fundamental principles and how it functions as a technique for enhancing image contrast. We will also explore its applications in grayscale and color image processing, highlighting its importance in improving visual clarity in images with poor contrast.

1.1 Introduction to Histogram Equalization

Images with poor contrast, often caused by insufficient or excessive lighting, can be made more visible by adjusting the brightness of darker areas and reducing the intensity of brighter spots. One of the most common methods for enhancing contrast is Histogram Equalization (HE), which redistributes the pixel intensity values more evenly across the entire intensity range. This process creates a flatter image histogram, leading to a more balanced and even spread of intensity distribution.

This report will explore the implementation of the HE algorithm, highlight its advantages and disadvantages, and examine some variants of the techniques that, in certain circumstances, outperform the original approach.

1.2 Setup

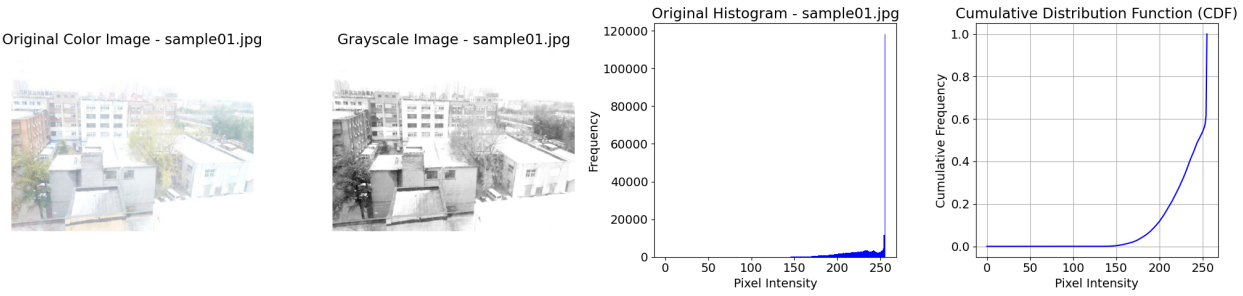
We will use the Jupyter Notebook environment and the Python programming language to implement the HE algorithm. The libraries used are as follows:

1. cv2 (OpenCV library): Handles image processing tasks, such as reading, writing and processing images
2. NumPy: Performs numerical computations, including calculating Cumulative Distribution Function (CDF)
3. Matplotlib: Visualizes images and histograms
4. os: Interacts with the operating system, primarily for file management tasks
5. math: Provides access to various mathematical functions for additional operations

1.3 Original Dataset

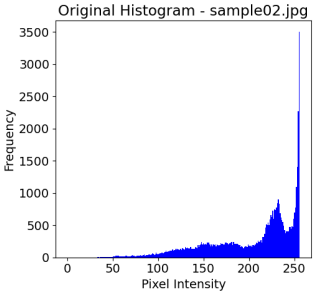
The original dataset consists of 8 sample images in color, labeled `sample01` through `sample08`, in Joint Photographic Experts Group (JPG/JPEG) format.

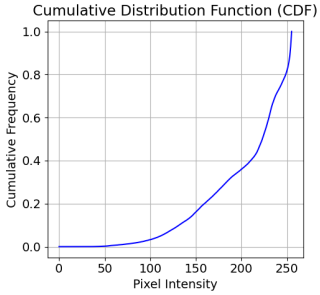
Figure 1 shows the original images and their grayscale counterparts, along with the corresponding histograms and CDFs.



Original Color Image - sample02.jpg

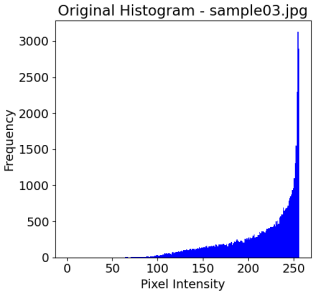
Grayscale Image - sample02.jpg

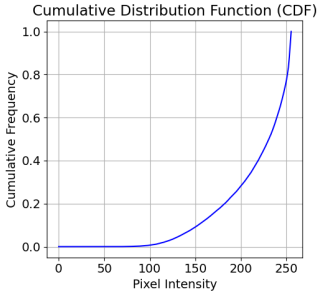




Original Color Image - sample03.jpg

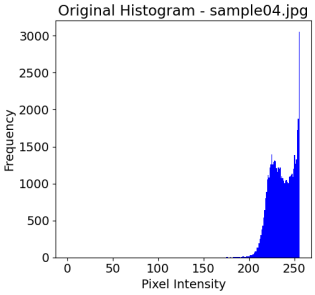
Grayscale Image - sample03.jpg

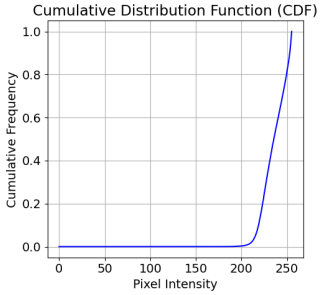




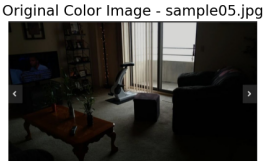
Original Color Image - sample04.jpg

Grayscale Image - sample04.jpg

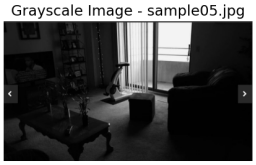


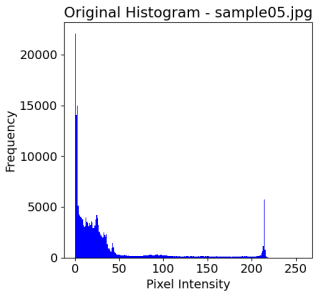


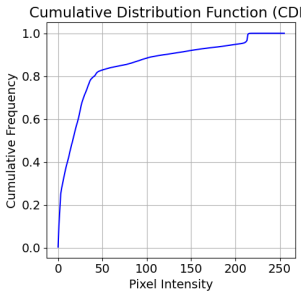
Original Color Image - sample05.jpg



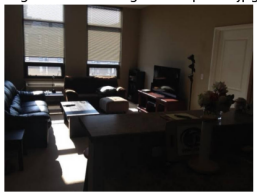
Grayscale Image - sample05.jpg



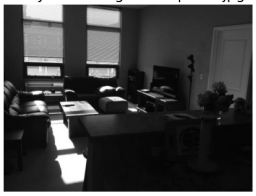


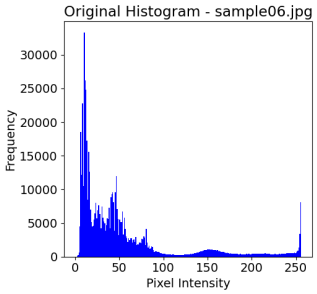


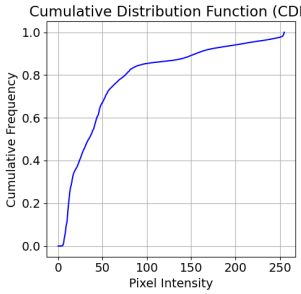
Original Color Image - sample06.jpg



Grayscale Image - sample06.jpg







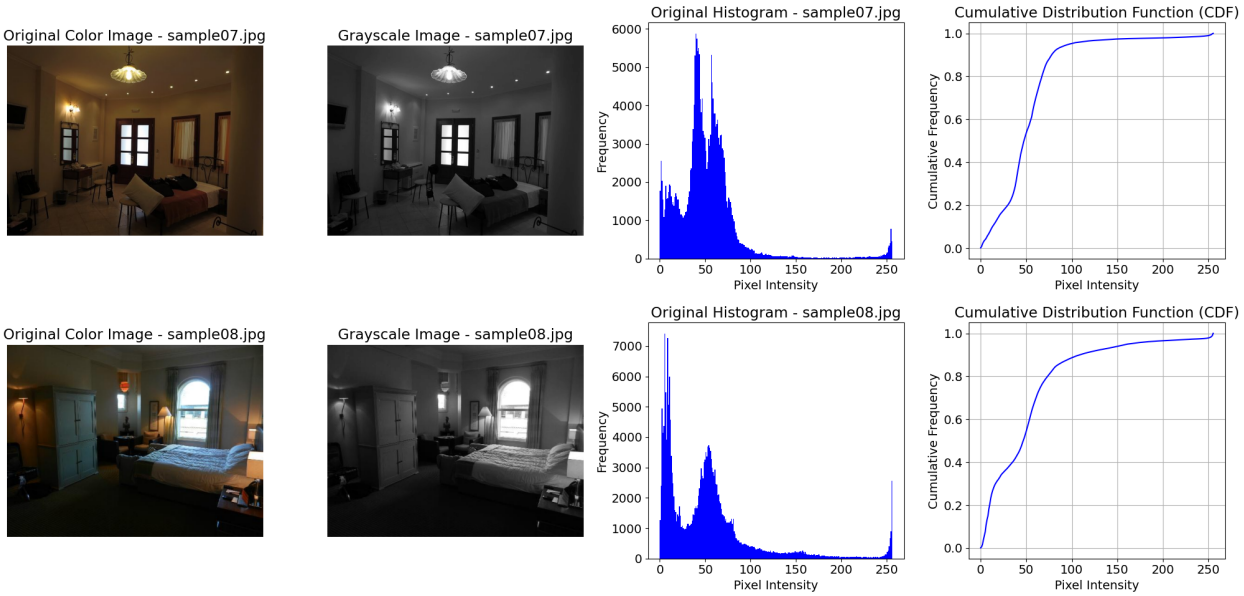


Figure 1: Original images and their grayscale counterparts, along with the corresponding histograms and CDFs, before applying the HE algorithm

1.4 Histogram Equalization Algorithm in Grayscale

We first apply the basic HE algorithm to the grayscale versions of the sample images. Each pixel intensity is represented by a value between 0 and 255, where 0 corresponds to absolute black and 255 to absolute white. This representation is based on the fact that the most common image bit depth is 8-bit, which provides 256 possible intensity levels. All the sample images in this report are stored with an 8-bit depth.

In our source code, we have defined globally that `UINT8_MAX = 255`, which represents the maximum value for grayscale intensity.

1.4.1 Explanation of Code

The following is an excerpt from our implementation of the grayscale HE algorithm.

```python
def histogram_equalization(image):
    # Step 1: Calculate the histogram
    hist, bins = np.histogram(image.flatten(), UINT8_MAX + 1, (0.0, UINT8_MAX + 1.0))

    # Step 2: Calculate the Cumulative Distribution Function (CDF)
    cdf = hist.cumsum()

    # Step 3: Normalize the CDF to calculate Sk values
    cdf_normalized = (cdf - cdf.min()) * UINT8_MAX / (cdf.max() - cdf.min())
    cdf_normalized = cdf_normalized.astype(np.uint8) # Convert to uint8

    # Step 4: Use the CDF to map the original intensities (at CDF index) to equalized
    ↪  intensities (CDF value)
    equalized_image = cdf_normalized[image]
```

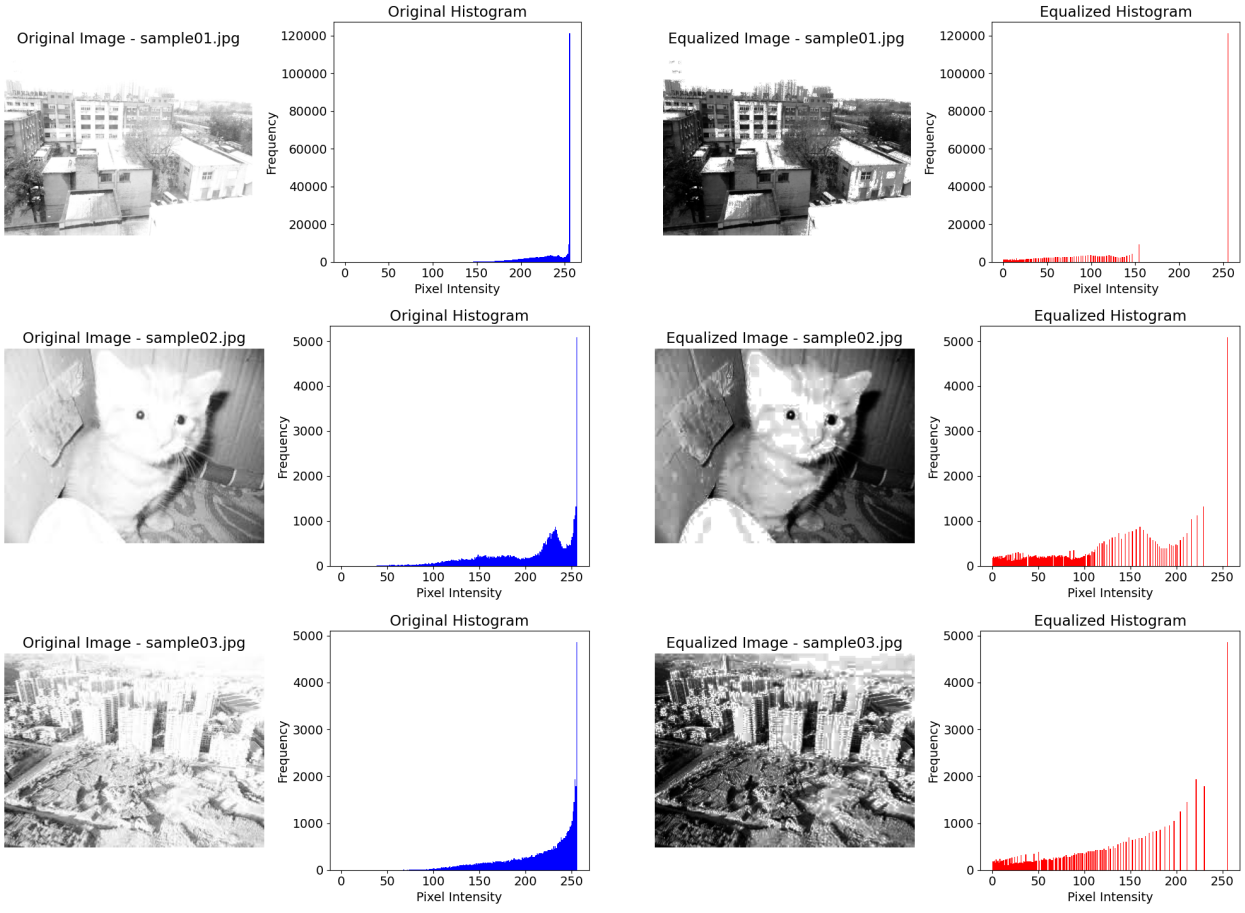
```
14
15      return equalized_image
```

First, we use NumPy to create a histogram of the image, which shows the frequency and occurrence of each intensity value (from 0 to 255). Based on this histogram, a CDF is computed as an intermediate step.

The CDF is then normalized to calculate the S_k values, which are the new pixel intensities corresponding to the original ones. As a result, the original pixel values are mapped to new values S_k based on the normalized CDF, ensuring that pixel intensities are spread across the full range (0 to 255). This process enhances the contrast of the image.

1.4.2 Output of Grayscale HE

Figure 2 shows the grayscale images and their equalized counterparts, along with the corresponding histograms.



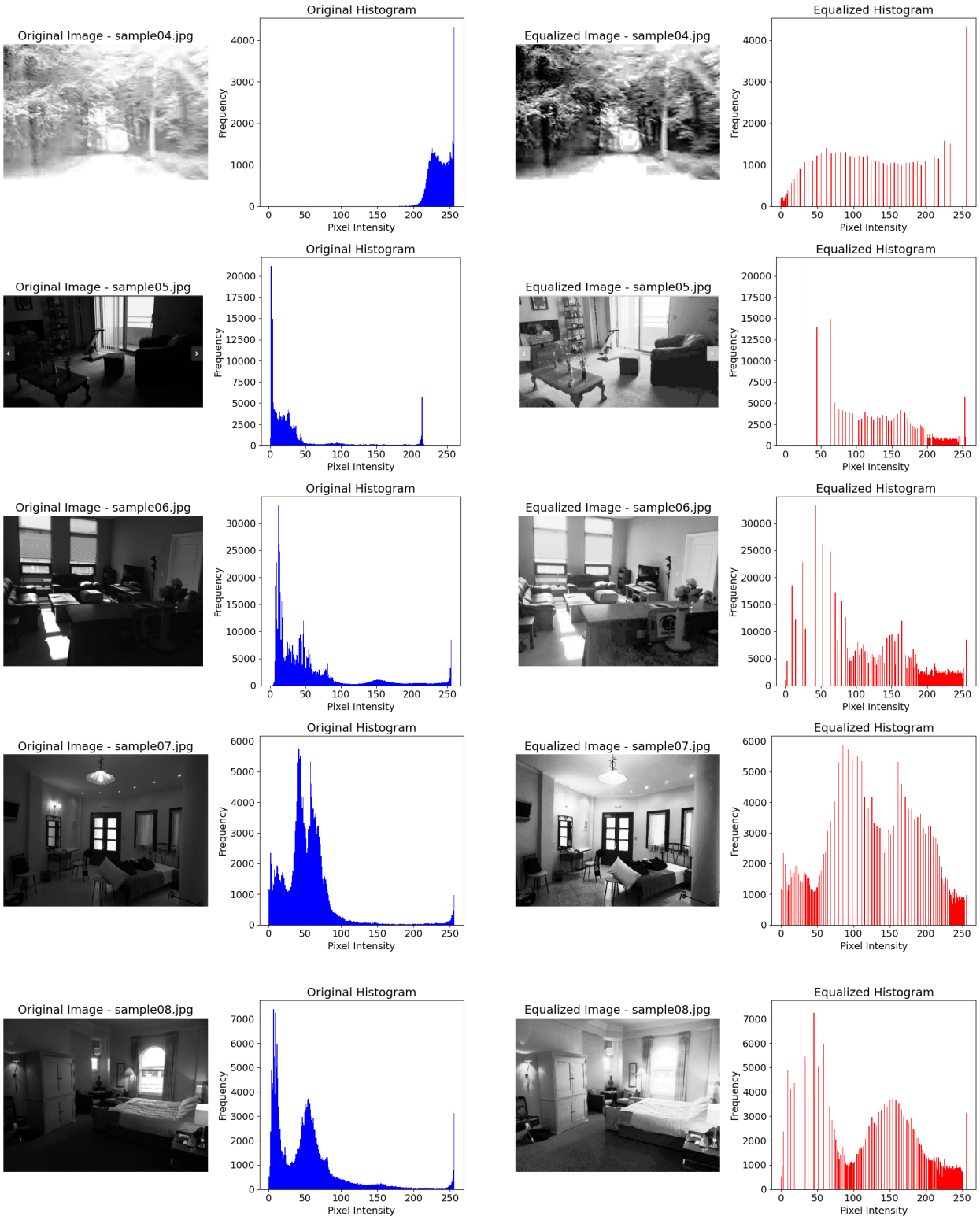


Figure 2: Grayscale images and their equalized counterparts, along with the corresponding histograms, after applying the grayscale HE algorithm

1.4.3 Evaluation

When comparing the grayscale and equalized images, the latter generally show improved overall contrast, as the pixel intensities are distributed more evenly across the image. However, it also suffers from a loss of detail in certain regions where the localized pixel densities are concentrated near the extremes of the intensity spectrum. The issue arises from applying HE uniformly across the entire image.

To mitigate this, we implemented Adaptive Histogram Equalization (AHE), Contrast Limited Adaptive Histogram Equalization (CLAHE) (Zuiderveld, 1994), and Bright-Wise Histogram Equalization (BWHE), which are discussed in Chapter 3 of this report.

Typically, HE is applied to grayscale images, such as those from Magnetic Resonance Imaging (MRI), X-ray and ultrasound, where the source contains no color. However, when the source contains color, directly applying HE can be undesirable, as it may result in information loss. In such cases, it is more effective to apply HE using a color format, which we have implemented in the following section.

1.5 Histogram Equalization Algorithm in RGB

Since the images are colored, we can apply the HE algorithm to the RGB color space. This involves applying the HE algorithm individually to each of the three channels—Red, Green and Blue—and then combining the mapped channels to form the final output image.

1.5.1 Explanation of Code

The following is an excerpt from our implementation of the RGB HE algorithm.

```
def histogram_equalization(img_in):
    # Segregate color streams
    b, g, r = cv2.split(img_in)

    # Calculate histograms
    h_b, _ = np.histogram(b.flatten(), UINT8_MAX + 1, (0.0, UINT8_MAX + 1.0))
    h_g, _ = np.histogram(g.flatten(), UINT8_MAX + 1, (0.0, UINT8_MAX + 1.0))
    h_r, _ = np.histogram(r.flatten(), UINT8_MAX + 1, (0.0, UINT8_MAX + 1.0))

    # Calculate CDF
    cdf_b = np.cumsum(h_b)
    cdf_g = np.cumsum(h_g)
    cdf_r = np.cumsum(h_r)

    # Mask all pixels with value = 0 and replace with mean of the pixel values
    cdf_m_b = np.ma.masked_equal(cdf_b, 0)
    cdf_m_b = (cdf_m_b - cdf_m_b.min()) * UINT8_MAX / (cdf_m_b.max() - cdf_m_b.min())
    cdf_final_b = np.ma.filled(cdf_m_b, 0).astype(np.uint8)

    cdf_m_g = np.ma.masked_equal(cdf_g, 0)
    cdf_m_g = (cdf_m_g - cdf_m_g.min()) * UINT8_MAX / (cdf_m_g.max() - cdf_m_g.min())
    cdf_final_g = np.ma.filled(cdf_m_g, 0).astype(np.uint8)

    cdf_m_r = np.ma.masked_equal(cdf_r, 0)
    cdf_m_r = (cdf_m_r - cdf_m_r.min()) * UINT8_MAX / (cdf_m_r.max() - cdf_m_r.min())
```

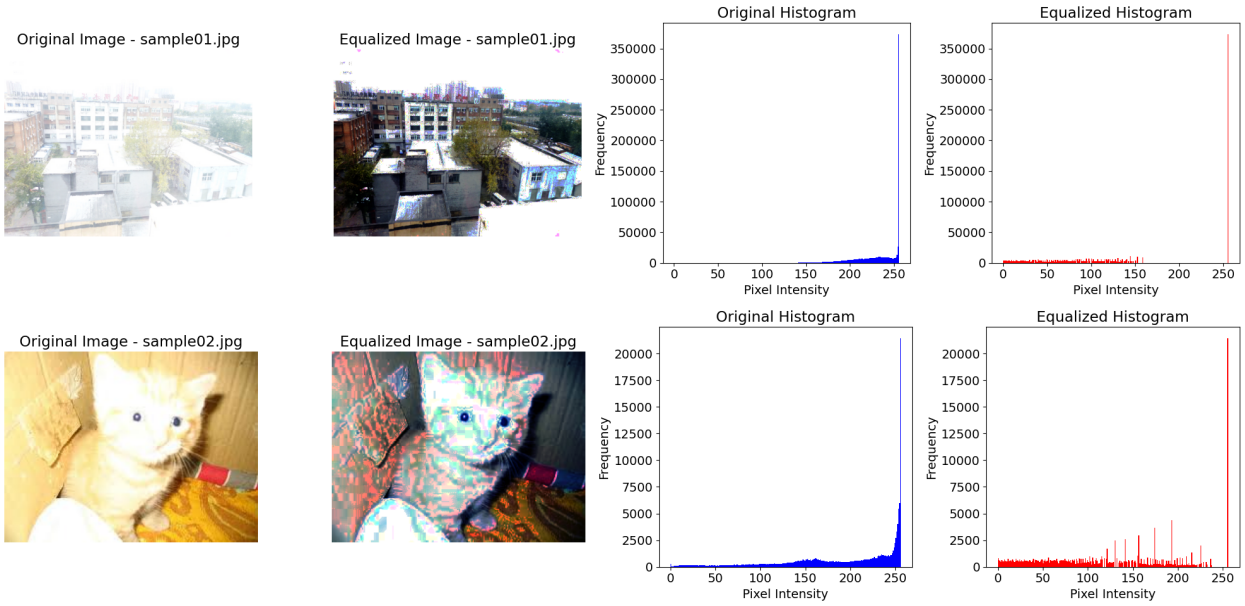
```
    cdf_final_r = np.ma.filled(cdf_m_r, 0).astype(np.uint8)

    # Apply the CDF to the original image channels
    img_b = cdf_final_b[b]
    img_g = cdf_final_g[g]
    img_r = cdf_final_r[r]

    # Merge the channels back into a color image
    img_out = cv2.merge((img_b, img_g, img_r))

    return img_out
```

First, we split the image into its three color channels: Blue, Green, and Red. In an RGB image, each pixel contains three values corresponding to the intensity of these three colors. As with the basic HE for grayscale images, we plot histograms to calculate the frequency of each intensity value (from 0 to 255) for each channel. Next, we compute the corresponding CDF, normalize it, and map the intensity values to a range of 0 to 255.

The original pixel intensities are then replaced with the new equalized values for each of the R, G, and B channels. Finally, we use the `cv2.merge()` function to combine the three channels back into a color image with enhanced contrast.

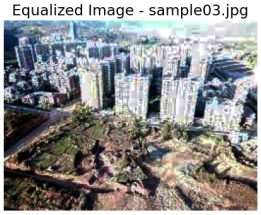
1.5.2 Output of RGB HE

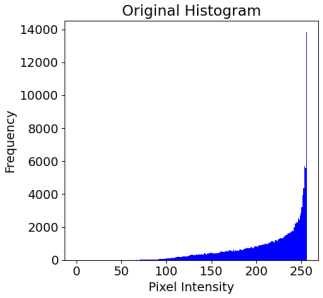
Figure 3 shows the original images and their equalized counterparts, along with the corresponding histograms.

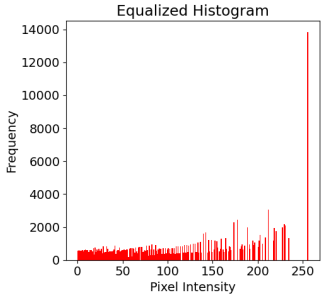


Original Image - sample03.jpg

Equalized Image - sample03.jpg

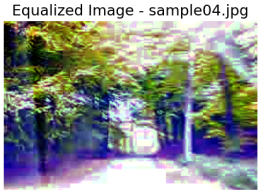


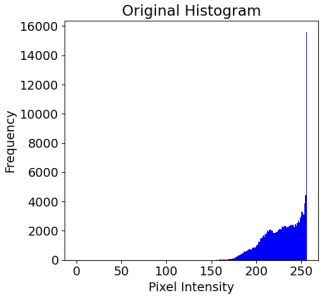


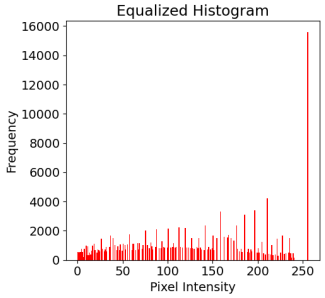


Original Image - sample04.jpg

Equalized Image - sample04.jpg

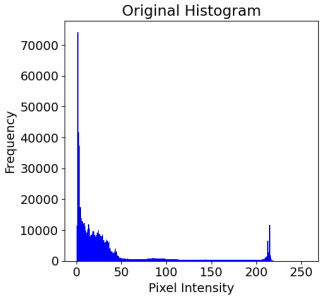


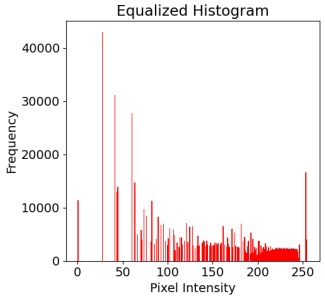




Original Image - sample05.jpg

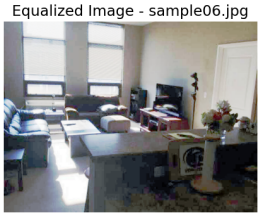
Equalized Image - sample05.jpg

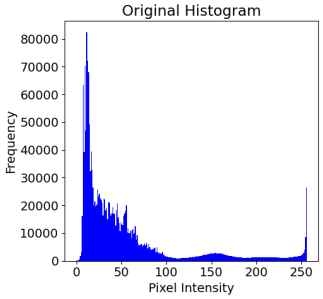


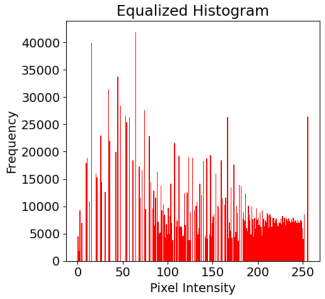


Original Image - sample06.jpg

Equalized Image - sample06.jpg

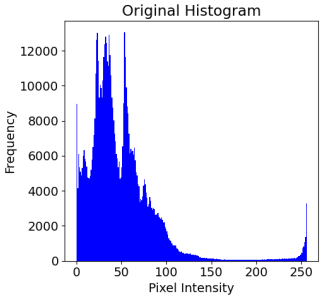


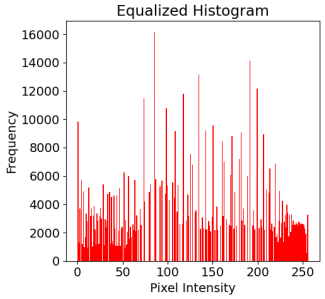




Original Image - sample07.jpg

Equalized Image - sample07.jpg





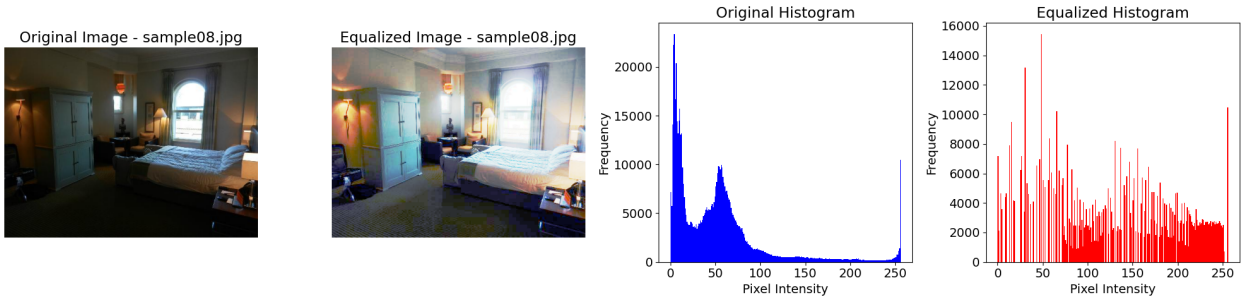


Figure 3: Original images and their equalized counterparts, along with the corresponding histograms, after applying the RGB HE algorithm

1.5.3 Evaluation

We observe that some of the enhanced images exhibit accidental colors. For example, red stripes appear in `sample02`, and a stronger blue tint is visible in `sample04`. This occurs because these colors were not prominent in the original images and, in some cases, represent redundant noise within the image.

When equalization is applied to each channel individually, all color channels are treated equally, which can significantly amplify previously inconspicuous color areas. As a result, this method has a drawback: it does not consider the relationship between the colors, leading to unintended amplification of certain colors.

1.6 Histogram Equalization Algorithm in YUV

To address the issue of accidental color amplification, we need an approach that enhances contrast based on lightness while preserving the original color relationships. A well-known formula (**bt2011studio**) is commonly used to convert color to grayscale by efficiently combining the color channels and extracting the luminance from each pixel:

$$Y = 0.299R + 0.587G + 0.114B$$

where R, G, and B represent the original red, green, and blue channel values of each pixel, and Y is the calculated luminance or grayscale value.

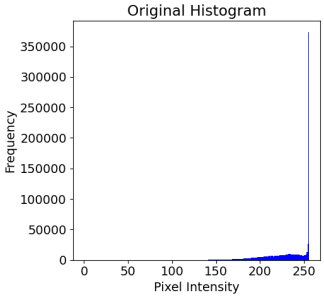
Inspired by this conversion process, we can extend grayscale processing algorithms to the color space by converting from RGB to YUV. We then apply the HE algorithm to the Y channel (luminance) individually. Since the color information is encoded in the U and V channels, the relationships between colors remain intact, ensuring that the color of the equalized image is not drastically altered.

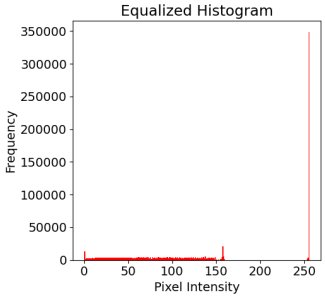
1.6.1 Output of YUV HE

Figure 4 shows the original images and their equalized counterparts, along with the corresponding histograms.

Original Image - sample01.jpg

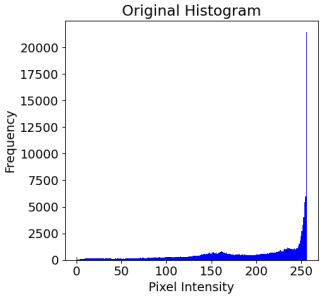
Equalized Image - sample01.jpg

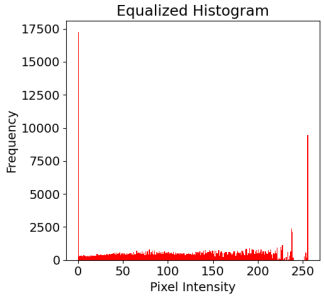




Original Image - sample02.jpg

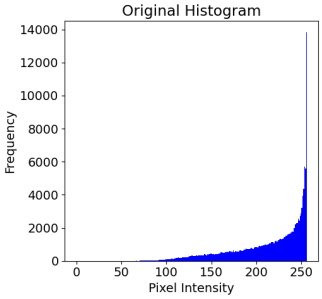
Equalized Image - sample02.jpg

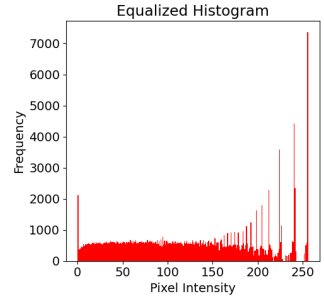




Original Image - sample03.jpg

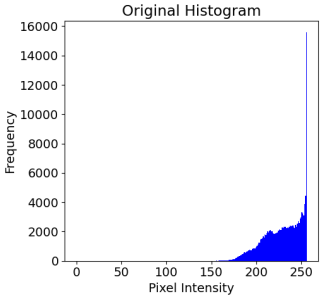
Equalized Image - sample03.jpg

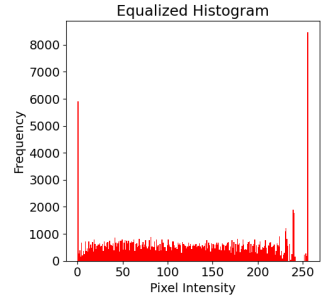




Original Image - sample04.jpg

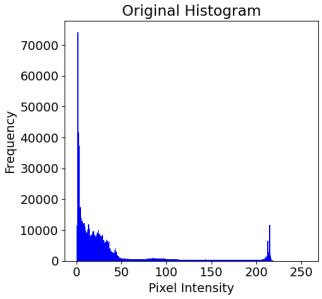
Equalized Image - sample04.jpg

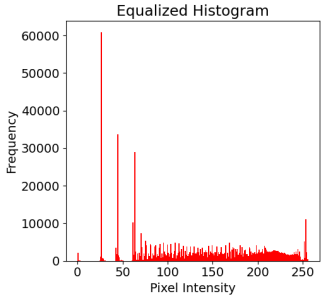




Original Image - sample05.jpg

Equalized Image - sample05.jpg





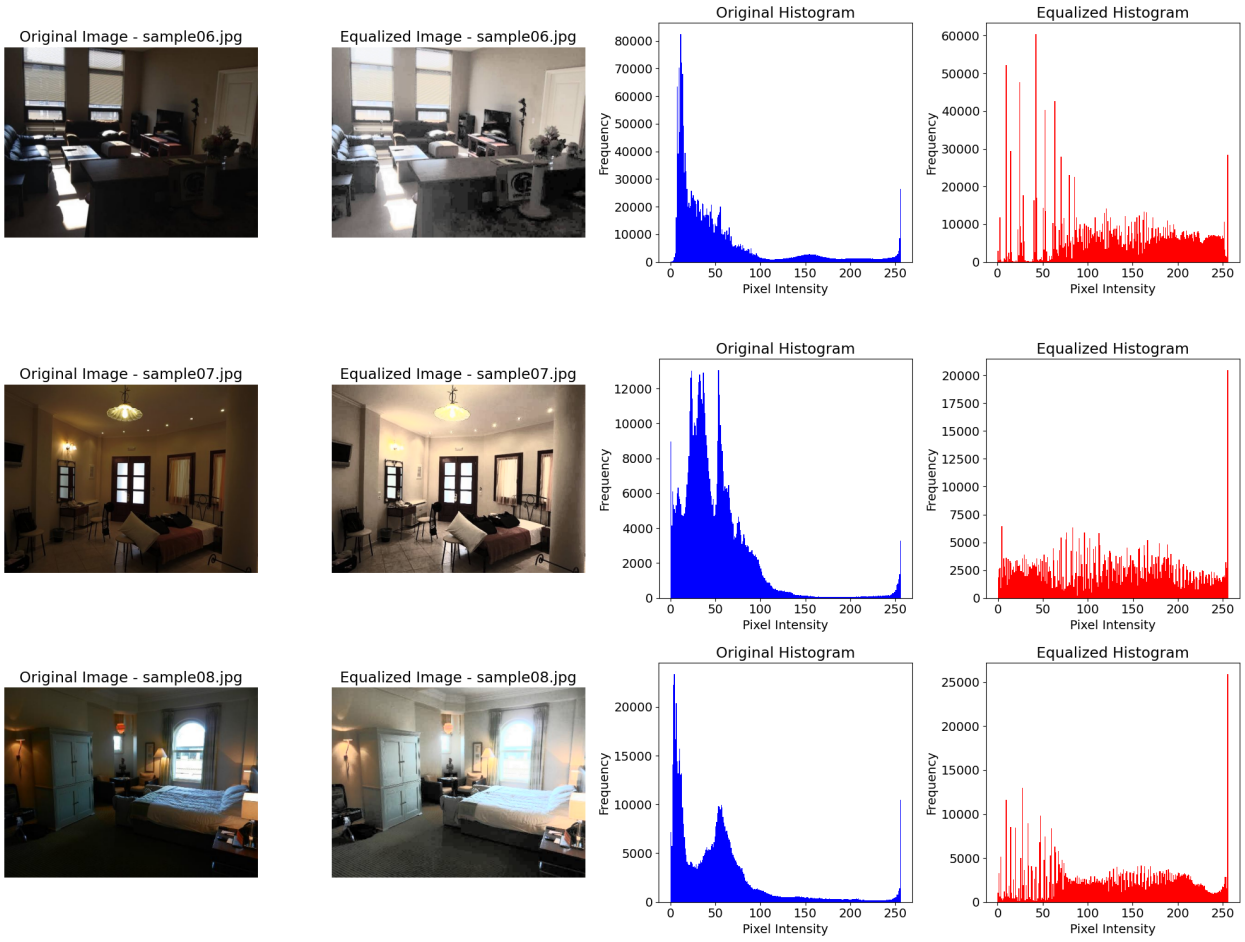


Figure 4: Original images and their equalized counterparts, along with the corresponding histograms, after applying the YUV HE algorithm

2 Discussion of the Histogram Equalization Algorithm

In this chapter, we will discuss the mathematical concepts behind HE, examine the advantages and disadvantages of applying the algorithm to the sample images, and analyze the potential causes of unsatisfactory results.

2.1 Mathematics Behind Histogram Equalization Algorithm

This section provides a brief overview of the mathematical principles behind the HE algorithm. When an image histogram is uniformly distributed, the entropy is maximized, which leads to increased contrast of the image. Thus, the initial idea of HE is to transform the original grayscale distribution of pixel intensities to a uniform distribution.

But how do we determine the necessary transformation? To answer this, we will first assume that the histogram is a continuous function, as it provides a straightforward and intuitive perspective.

In Continuous Case. In this case, we approximate the histogram as a continuous function. To proceed, let's first introduce two fundamental lemmas, which play important roles in the forthcoming mathematical derivations.

Lemma 1 (Transformation of Random Variables (Wood, 2015)). *Let X and Y be continuous random variables, with probability density functions f_x and f_y (f_y is unknown) respectively. Given $Y = g(X)$, where g is an invertible function, then we have*

$$f_y(y) = f_x(g^{-1}(y)) \left| \frac{dx}{dy} \right|.$$

Proof. Here is a simplified proof. Let the cumulative distribution function of X and Y be $F_x(x)$ and $F_y(y)$ respectively. Because g is invertible, by definition, we have,

$$\begin{aligned}
F_y(y) &= Pr[Y \leq y] \\
&= Pr[g^{-1}(Y) \leq g^{-1}(y)] \qquad \textbf{(if } g \textbf{ is increasing)} \\
&= Pr[X \leq g^{-1}(y)] \\
&= F_x(g^{-1}(y)) \\
\textbf{or} \qquad &= Pr[g^{-1}(Y) > g^{-1}(y)] \qquad \textbf{(if } g \textbf{ is decreasing)} \\
&= Pr[X > g^{-1}(y)] \\
&= 1 - F_x(g^{-1}(y)).
\end{aligned}$$

Lastly, after differentiating both sides and taking the absolute value, we obtain

$$f_y(y) = F_y^{'}(y) = F_x^{'}(g^{-1}(y)) \left| \frac{dx}{dy} \right| = f_x(g^{-1}(y)) \left| \frac{dx}{dy} \right|.$$

□

Lemma 2 (The Derivative Rule of Inverse Function). *Suppose the function $y = f(x)$ is continuous and strictly monotonic in an interval $(a, b) \in \mathbb{R}$. Let $\alpha = \min[f(a^+), f(b^-)]$ and $\beta = \max[f(a^+), f(b^-)]$. The inverse function $x = f^{-1}(y)$ is differentiable in the interval $(f(\alpha), f(\beta))$, and we have*

$$[f^{-1}(y)]^{'} = \frac{1}{f^{'}(x)}.$$

Now we can begin discussing the HE algorithm. Based on the idea introduced at the beginning of this section, to improve the contrast of an image, we need to map its original histogram to a uniformly distributed one. To achieve this, we must determine the appropriate transformation required for this mapping, starting with the continuous assumption (though it is not true actually).

Suppose X and Y are random variables, with $P_x(x)$ and $P_y(y)$ as their probability density functions respectively. We require Y to be uniformly distributed. Let $y = T(x)$ be the unknown transformation function, which is continuous and monotonically increasing. We now aim to determine $y = T(x)$. Let L be the maximum grayscale value of the image (for example, $L = 256$). Thus, we have

$$x = T^{-1}(y), \qquad 0 \leq y \leq L - 1.$$

By Lemma 1,

$$P_y(y) = P_x(x) \left| \frac{dx}{dy} \right| = P_x(T^{-1}(y)) \left| [T^{-1}(y)]^{'} \right|.$$

Because $P_y(y)$ is uniformly distributed, we have

$$P_y(y) = \frac{1}{L-1}, \quad \text{for } 0 \leq y \leq L-1.$$

Using Lemma 2, combining with the above equations,

$$P_y(y) = P_x(x)\left|\frac{1}{T'(x)}\right| = \frac{1}{L-1}.$$

Hence,

$$T^{'}(x) = (L-1)P_x(x).$$

Integrating both sides of the equation from 0 to x, we obtain

$$T(x) = (L-1)\int_0^x P_x(z)dz + C.$$

We could determine that $C = 0$, since $T(0) = 0$. Therefore, we have successfully derived the transformation function,

$$y = T(x) = (L-1)\int_0^x P_x(z)dz.$$

In Discrete Case. The discrete case is more complicated than the continuous case, as there are many different situations to consider. However, the underlying idea remains the same as in continuous case. Instead of working with the integration of the probability density function, we deal with the sum of probabilities.

Let x and y, for $0 \leq x, y \leq L$ and $x, y \in \mathbb{Z}$, represent the grayscale values, and $p(x)$ represent the number of pixels at grayscale value x in the histogram. Additionally, let w and h denote the width and height of the image respectively. Then the transformation function $y = T(x)$ is given by

$$T(x) = (L-1)\sum_{z=0}^{x}\frac{p(z)}{w \times h}.$$

In this case, using this transformation function, we cannot actually guarantee that the result will be a uniform distribution. However, it will still become closer to a uniform distribution, which is meaningful.

Further Discussion. From the mathematics derivation, we can observe that the HE algorithm has some drawbacks. For instance, when the transformation function is in continuous form, the mapping is invertible. However, the mapping becomes non-invertible once it is changed to the discrete form. This may lead to a loss of information.

Specifically, the HE algorithm merges grayscale values that appear infrequently, resulting in a loss of some fine details. Conversely, for grayscale values that occur more frequently, the transformation stretches them, causing them to occupy more grayscale values while compressing others. For example, HE will often map an image with dominant dark areas to a washed-out image. This happens because a large number of pixels' grayscale values concentrated in a small region of the grayscale are mapped to the upper end of the grayscale range. We can observe this phenomenon in the HE equalized images `sample01`, `sample05`, `sample06`, and `sample08`, as shown in Figure 4.

Several methods can mitigate the negative effects caused by this drawback. For example, we can take the square root of the histogram to reduce the influence of frequently occurring grayscale values. Additionally, the histogram can be truncated by directly removing portions that exceed a certain threshold, thereby reducing the impact of frequently occurring grayscale levels. Furthermore, based on this idea, the excess portion can be evenly distributed across each bin of the histogram. In fact, this method is used in Contrast Limited Adaptive Histogram Equalization (CLAHE) (Zuiderveld, 1994), which we will introduce later as an improvement to the basic HE algorithm.

In the next section, we will discuss the pros and cons of the HE algorithm in more detail.

2.2 Pros and Cons

In this section, we will examine the advantages and disadvantages of using the HE algorithm on the sample images and analyze the possible causes of unsatisfactory results.

2.2.1 Grayscale HE

The following are the pros of using the grayscale HE algorithm:

- Contrast Improvement: The HE algorithm is highly effective in improving the contrast of grayscale images, especially when the image has poor or uneven lighting, as it redistributes the intensity values evenly across the available range (0 to 255). This is demonstrated by the increased visibility of the equalized images of `sample04` through `sample08`.
- Computationally Inexpensive: The HE algorithm is computationally inexpensive to apply to grayscale images as they contain only one channel. Additionally, it does not require any complex parameter tuning.
- No Color Distortion: There is no need to worry about color shifting or distortion as grayscale images do not contain any color information.

The following are the cons of using the grayscale HE algorithm:

- Inappropriate for Dominant Dark Areas: In images with extensive dark regions, HE may map a large number of pixel intensities to the upper end of the grayscale, resulting in a light and washed-out appearance, as seen in the equalized image of `sample01`.
- Loss of Detail: In regions where pixel intensities are clustered closely, the HE algorithm can merge these values, resulting in a loss of fine details. This effect is visible in the equalized images of `sample01` through `sample04`.
- Noise Over-Amplification: Areas with low contrast or subtle gradients may have noise amplified, making the image appear grainy or harsh. This is evident in the equalized images of `sample01` through `sample04`.
- Global Adjustment: HE applies a uniform adjustment to the entire image, which can lead to unnatural brightness changes in certain regions, as observed from the equalized images of `sample01` through `sample04`.

2.2.2 RGB HE

The following are the pros of using the RGB HE algorithm:

- Contrast Improvement: When applied individually to the R, G and B channels, HE can improve the overall contrast of the image, making details in darker and lighter regions more visible.
- Channel-Wise Control: HE can enhance different aspects of the image, such as improving contrast in specific color regions (e.g., boosting reds or blues).

The following are the cons of using the RGB HE algorithm:

- Color Distortion: Applying HE separately to the R, G, and B channels can lead to unwanted color shifts or artifacts. Colors that were previously balanced might become exaggerated, leading to unrealistic or "accidental" colors. This effect is noticeable in many of the equalized images.
- Loss of Color Relationship: HE treats each color channel independently, so the relationship between the colors (hue, saturation) is lost, often leading to unnatural color combinations or washed-out colors, as seen in several of the equalized images.
- Inconsistent Results: HE is not optimal for images with subtle variations in color, as the algorithm may exaggerate certain colors, making some areas look unnatural. This issue is also apparent in many of the equalized images.

2.2.3 YUV HE

The following are the pros of using the YUV HE algorithm:

- Luminance-Based Adjustment: By applying HE to the Y channel (luminance), HE improves contrast without affecting the chrominance (color information). This avoids the color distortion issues that occur with RGB images, preserving the natural color relationships, as evidenced by the equalized images of `sample05` through `sample08`.
- Better Contrast Control: The YUV color space separates luminance (brightness) from chrominance (color), allowing the HE algorithm to work on the brightness component only, leading to more controlled and visually pleasing results, as portrayed by the equalized images of `sample05` through `sample08`.

The following are the cons of using the YUV HE algorithm:

- Computational Complexity: Applying HE to the YUV color space requires converting the image from RGB to YUV, applying HE to the Y channel, and converting it back to RGB, adding computational complexity.
- Loss of Subtle Gradients: Similar to grayscale images, subtle gradients can be lost, and noise may be amplified in certain regions of the image. This occurs in the equalized images of `sample01` through `sample04`.

2.3 Possible Causes of Unsatisfactory Outputs

The following are some of the factors that can cause HE to produce unsatisfactory outputs:

- Uneven Distribution of Pixel Intensities: When the original image has pixel intensities concentrated within a narrow range, HE may struggle to redistribute them evenly, leading to suboptimal contrast enhancement.
- Inappropriate for Dominant Dark Areas: Images with dominant dark areas can cause HE to map a large number of pixels' grayscale values concentrated in some small regions of the grayscale to the upper end of the grayscale range, which leads to washed-out areas. We have discussed this phenomenon from a mathematical perspective in Section 2.1.
- Presence of Noise: HE may exacerbate noise in images by enhancing the contrast of every pixel, including noise, which can further degrade image quality.
- Loss of Local Contrast: HE's global adjustment may not preserve local contrast variations in areas with different brightness levels, leading to images that might appear unnatural or excessively harsh in certain regions.

In summary, while HE is a valuable tool for enhancing contrast in grayscale images, its application can present challenges, particularly in images with noise, dominant dark areas, varying local brightness, or in color images, where applying HE separately to each color channel can result in color distortion and unnatural hues. Alternative methods, such as Contrast Limited Adaptive Histogram Equalization (CLAHE) or color space conversion techniques, may offer improved results by addressing some of the limitations associated with standard HE in both grayscale and color images.

3 Improvements in the Basic Histogram Equalization Algorithm

In this chapter, we will explore potential improvements to the basic HE algorithm, focusing on Adaptive Histogram Equalization (AHE), Contrast Limited Adaptive Histogram Equalization (CLAHE), and Bright-Wise Histogram Equalization (BWHE). These methods address the limitations of basic HE, such as over-amplification of noise, loss of local details, and uneven brightness distribution.

3.1 Adaptive Histogram Equalization (AHE)

We implemented AHE to increase contrast while preserving original details. In AHE, each pixel is modified based on its neighboring pixels by applying HE locally. Although the resulting images are not entirely optimal, the method is somewhat effective, with certain details being successfully preserved. Figure 5 shows the preserved contrast in `sample07` using AHE.

Figure 5: Contrast preservation in `sample07` using AHE

Pure AHE suffers from three main drawbacks. The first is the over-amplification of regions with similar pixel intensities. As seen in the image below, the background of the original image has almost no contrast, and after the AHE process, it results in a noticeable black patch. Figure 6 shows the pitch-dark patches in the background of `sample01`.

Figure 6: Regional over-amplification in `sample01`

The second drawback is that border regions do not have enough neighboring pixels to accurately compute their intensities. A common corrective technique is to mirror the existing neighboring pixels across the border to fill in the missing ones before applying the HE algorithm. Figure 7 shows the unprocessed borders in `sample04`.

Figure 7: Unprocessed borders in `sample05`

The third drawback is that AHE is computationally expensive. The time complexity of performing AHE is $O(mn)$, where n is the number of pixels in the image, and m is the number of neighboring pixels considered for each pixel. For high-resolution images, n can grow significantly, making the algorithm computationally infeasible for practical use in such cases, especially when processing a large number of images.

3.2 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Contrast Limited Adaptive Histogram Equalization (CLAHE) addresses the three issues mentioned above through Histogram Clipping, Tiling and Interpolation (Pizer et al., 1987).

To prevent regional over-amplification, CLAHE uses Histogram Clipping to redistribute excess pixels from high bin counts (those that exceed a predefined threshold β) evenly across the various histogram bins. This process is iterative as the redistribution is repeated if the resulting intensity still exceeds β. Consequently, the resulting image intensity is generally darker compared to one processed by AHE. Figure 8 shows an illustration of Histogram Clipping.

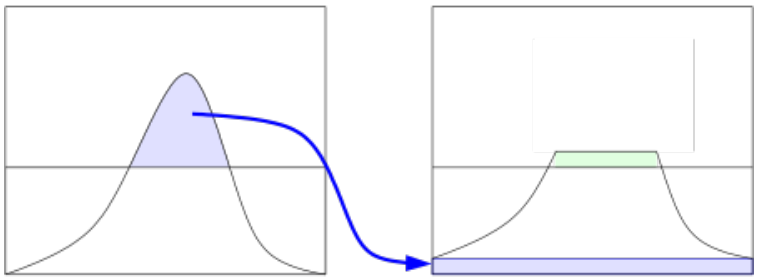


Figure 8: Histogram Clipping (Vswitchs, 2011)

Improved computational performance is achieved by splitting the image into tiles to compute local histograms. Linear and bilinear interpolations are then used to compute the border pixels and smooth the transitions between tiles, thereby removing abrupt changes and preserving continuity.

Figure 9: Side-by-side comparison of `sample07` after applying the various HE algorithms

Table 1 shows the execution times of the various HE algorithms for the 8 sample images using Google Colab.

Algorithm	Execution Time
HE	0.099198 seconds
AHE	883.284085 seconds
CLAHE	0.020542 seconds

Table 1: Execution times of the HE algorithms using Google Colab

3.3 Bright-Wise Histogram Equalization (BWHE)

Another drawback of the basic HE algorithm is that it flattens the histogram across the entire range of possible grayscale values, from absolute black (0) to absolute white (255), regardless of the image content. Figure 10 shows `sample08` after applying the various HE algorithms.

(a) Original (b) HE

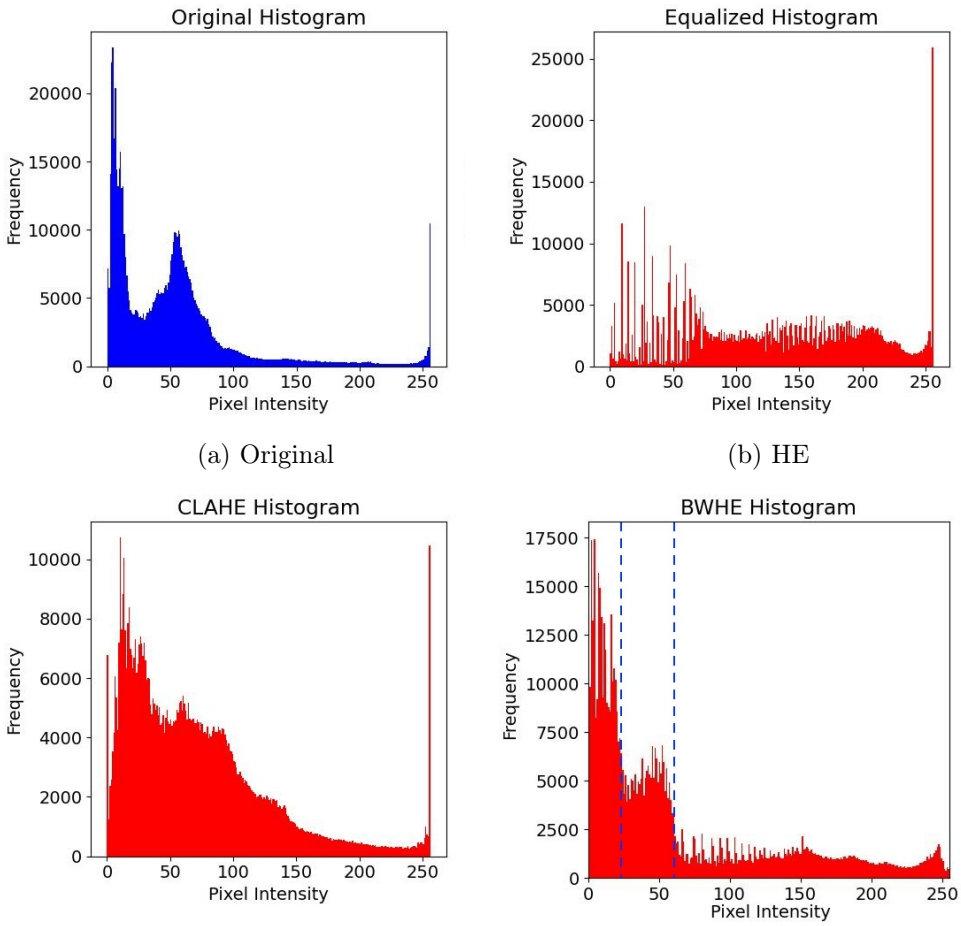
(c) CLAHE (d) BWHE

Figure 10: Original image of `sample08`, along with its equalized counterparts using the HE, CLAHE, and BWHE algorithms

From the original image shown in Figure 10(a), the basic HE algorithm is applied to the Y channel (corresponding to luminance) in the YUV color space, as seen in Figure 10(b). We observe that the portrayal of the content is significantly affected, as the global lightness is amplified, altering the overall atmosphere of the image. Similarly, while the CLAHE algorithm in Figure 10(c) enhances the contrast of the original image, it also enlarges noise in darker regions, such as the shadowed side of the bed, making it more noticeable.

In situations where contrast enhancement is not needed across the entire brightness range, BWHE can equalize the image separately for three levels of brightness: dark, middle and bright pixels (Srinivasan and Balram, 2006). As demonstrated in Figure 10(d), this approach preserves the dynamic range of the image, preventing rough color blocks from becoming more prominent.

The BWHE algorithm is not computationally expensive, as it divides the image into three brightness levels rather than processing the entire range at once, similar to the basic HE algorithm. This results in a time complexity of a $O(n)$, where n is the number of pixels in the image.



(a) Original (b) HE

(c) CLAHE (d) BWHE

Figure 11: Original histogram of `sample08`, along with its corresponding histograms using the HE, CLAHE and BWHE algorithms

In the histograms of the same image after applying different algorithms, as shown in Figure 11, BWHE preserves the brightness of the pixels within their respective regions, with the histograms for the three regions equalized separately. The widths of these regions vary because the number of pixels in each region is not equal. For instance, in `sample08`, most pixels are dark, with more than one-third of the pixels having a brightness below 25 and more than two-thirds below 60. The BWHE algorithm does not force these pixels to become significantly brighter, thereby preserving the overall dark atmosphere of the image—in this case, the ambiance of a dark room illuminated by lamps and sunlight.

However, there is a drawback. If the histogram frequencies are concentrated in a narrow range—meaning that most pixels are either very dark or very bright—and the global brightness needs smoothing, BWHE would perform worse than the basic HE algorithm.

4 Conclusion

In this report, various HE algorithms were implemented to enhance the contrast of 8 sample images. The basic HE algorithm was applied to the grayscale version of the images, as well as to the channels in the RGB and YUV color spaces. The report outlined the mechanisms and the pros and cons of each method were evaluated.

Overall, the basic HE algorithm was found to be effective for enhancing contrast in grayscale images. However, it struggles with color images due to variations in local brightness and the amplification of noise. To address these issues, AHE, CLAHE, BWHE were implemented as improvements.

While AHE helps preserve original details, its primary drawback is its high computational cost, leading to long execution times. CLAHE mitigates regional over-amplification and noise using a clipping mechanism, redistributing pixel intensities within a certain threshold. This method is effective for enhancing local contrast while minimizing noise, making it suitable for applications requiring global contrast improvement without excessive noise amplification. Lastly, BWHE uses a segmented approach to equalize pixels across three brightness levels—dark, middle, and bright—making it useful for images with extreme brightness variations while preserving overall tonal balance.

In conclusion, AHE, CLAHE, and BWHE all outperformed the basic HE algorithm. Among the three, CLAHE and BWHE strike a good balance between execution time and image quality, making them more practical for real-world applications. The choice of algorithm ultimately depends on the specific application and the characteristics of the image being processed.

References

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, *39*(3), 355–368. https://doi.org/10.1016/S0734-189X(87)80186-X

Srinivasan, S., & Balram, N. (2006). Adaptive contrast enhancement using local region stretching. *Proceedings of the 9th Asian Symposium on Information Display*, 152–155.

Vswitchs. (2011). Excess redistribution in contrast-limited adaptive histogram equalization. https://commons.wikimedia.org/wiki/File:Clahe-redist.svg

Wood, S. (2015). Core statistics. Cambridge University Press. https://books.google.com.sg/books?id=LeJwBwAAQBAJ

Zuiderveld, K. (1994). Contrast limited adaptive histogram equalization. In *Graphics gems iv* (pp. 474–485). Academic Press.