Homework 2

Histogram Equalization

1. Screenshots

（本次作業均無使用 equalizeHist()等等 API）

Original img

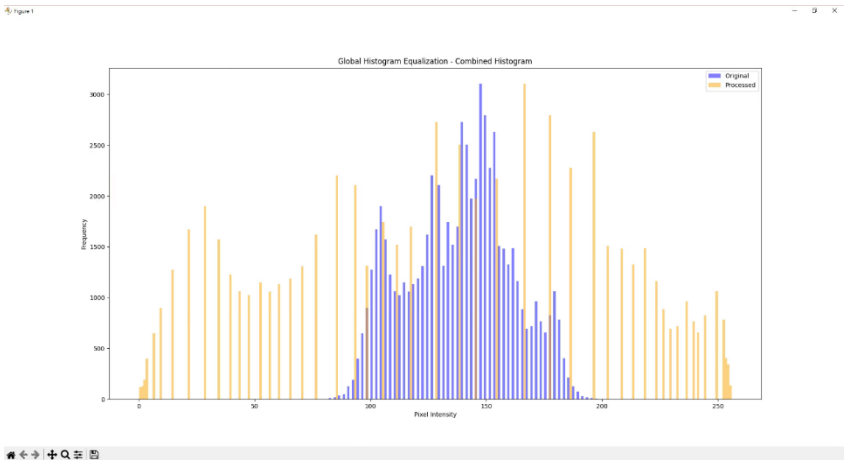
Global HE

HSPNR =28.424

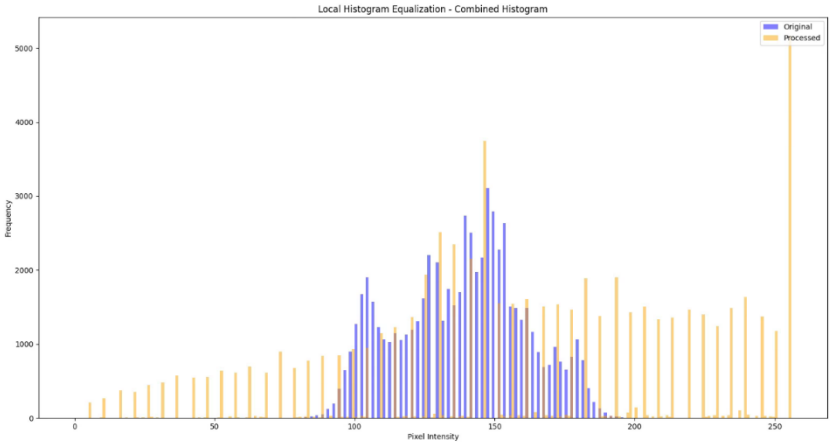
Local HE (Window size: 7x7)

HSPNR =27.979

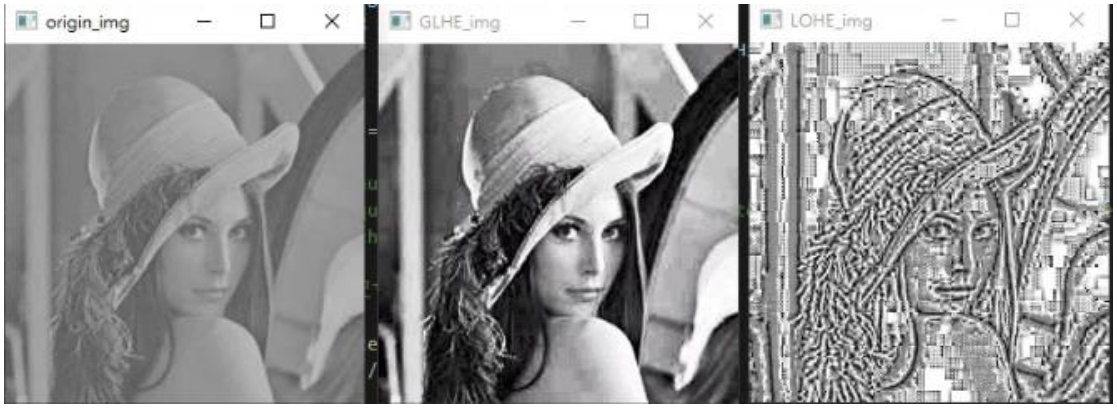
Global HE 直方圖



Local HE(Window size: 7x7) 直方圖



整體:

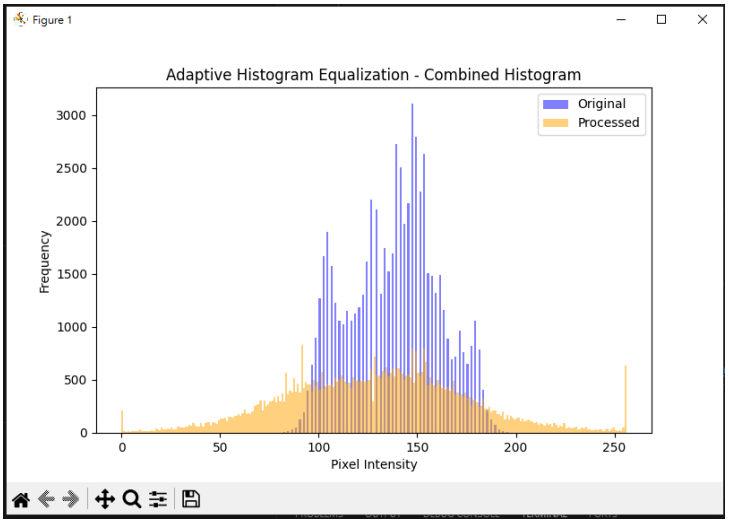


Bonus: 用動差進行Histogram Equalization(mean and variance)

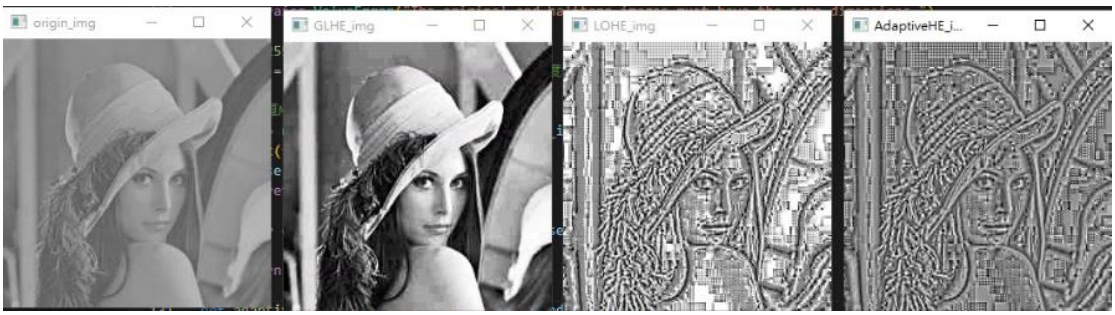
Window size 7*7

HSPNR = 28.131

Window size: 7x7 直方圖



整體:



2. Explain

MSE、HSPNR 計算結果:(window size 7*7)

```
MSE = 92.73744201660156
The HPSNR value of Global_Histogram_img is: 28.424123206901818
MSE = 102.73258972167969
The HPSNR value of Local_Histogram_img is: 27.97959197129442
MSE = 99.21910095214844
The HPSNR value of Adaptive_Histogram_img is: 28.13072145767384
```

方法解釋:

(1) Global Histogram Equalization:

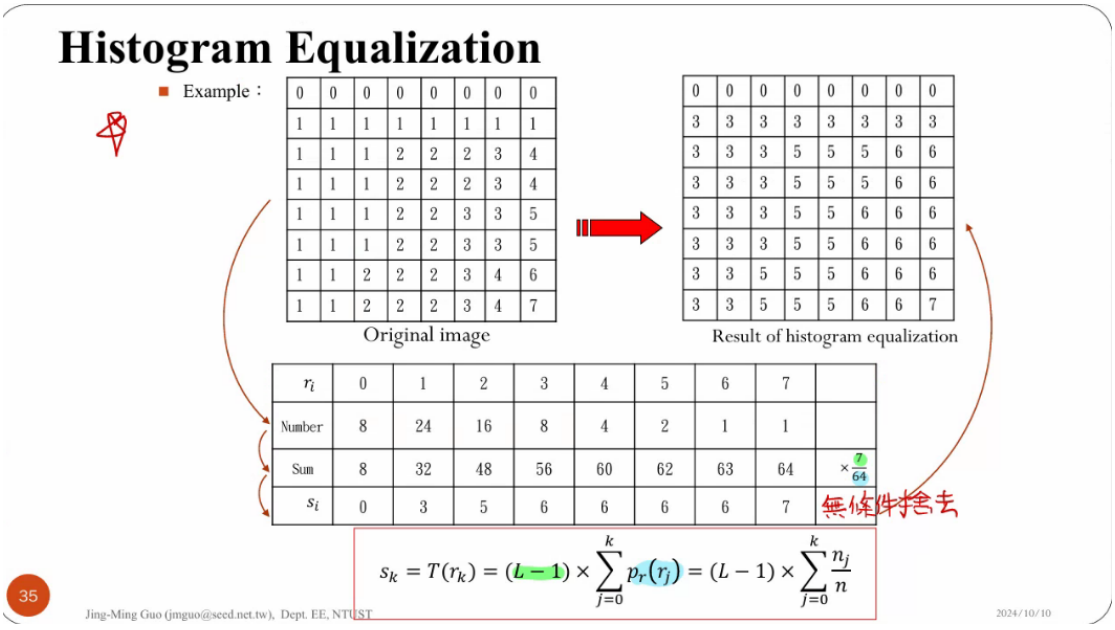
步驟:
Step1:計算直方圖 → 獲取每個灰階值的像素數量。
Step2:計算累積分佈函數（CDF）→ 將灰階值正規化到新的分佈區間。
Step3:建立灰階映射表 → 將原影像的灰階值映射到新的分佈。
Step4:應用映射表替換像素值 → 生成對比度增強的影像。

Code:

```python
def calculate_histogram(img):
    # 建立大小為256的直方圖陣列
    histogram = np.zeros(256, dtype = int)

    height, width = img.shape

    for i in range(height):
        for j in range(width):
            pixel_value = img[i, j]
            histogram[pixel_value] +=1

    return histogram
```

```python
def Global_HE(img):

    # Step 1: 計算像素強度的出現次數（直方圖）
    histogram = calculate_histogram(img)

    # Step 2: 正規化直方圖，使其總和為影像中像素數量
    height, width = img.shape
    num_pixels = height * width
    cdf = np.zeros(256, dtype = float) #累積分部函數(CDF)
    cdf[0] = histogram[0] / num_pixels

    #計算累積分部函數(CDF)
    for i in range(1, 256):
        cdf[i] = cdf[i-1] + histogram[i] / num_pixels

    #Step 3: 建立對應的像素值轉換表
    equalized_lut = np.round(cdf * 255).astype(np.uint8) #將CDF映射到[0, 255]

    #Step 4: 根據轉換表對原始影像進行像素強度調整
    equalized_img = np.zeros_like(img, dtype = np.uint8)
    for i in range(height):
        for j in range(width):
            equalized_img[i, j] = equalized_lut[img[i, j]]

    plot_combined_histogram(img, equalized_img, "Global Histogram Equalization")
    Global_Histogram_img_HPSNR = HPSNR(img, equalized_img)
    print(f"The HPSNR value of Global_Histogram_img is: {Global_Histogram_img_HPSNR}")
    return equalized_img
```

講義說明:



r_i	0	1	2	3	4	5	6	7	
Number	8	24	16	8	4	2	1	1	
Sum	8	32	48	56	60	62	63	64	$\times \frac{7}{64}$
s_i	0	3	5	6	6	6	6	7	

(2) Local Histogram Equalization:

步驟:

Step1: 劃分影像為局部區域

Step2: 計算局部直方圖(需考慮邊界)

Step3: 計算累積分佈函數 (CDF) → 將灰階值正規化到新的分佈區間。

Step4: 建立灰階映射表 → 將原影像的灰階值映射到新的分佈。

Step5: 應用映射表替換像素值 → 生成對比度增強的影像。

Code:

```
def Local_HE(img, window_size = 3):

    # Step 1: 初始化輸出影像與尺寸資訊
    height, width = img.shape
    locals_equlized_img = np.zeros_like(img, dtype = np.uint8)
    half_window = window_size // 2

    # Step 2: 對每個像素進行局部直方圖等化
    for i in range(height):
        for j in range(width):
            #定義局部範圍(考慮邊界)
            row_min = max(0 , i - half_window)
            row_max = min(height , i + half_window + 1) #要 +1 後面切片範圍才會是[i - 1, i + 1] , j同理
            col_min = max(0 , j - half_window)
            col_max = min(width, j + half_window + 1)

            #提取局部區域
            local_region = img[row_min:row_max, col_min:col_max]

            #計算局部區域的直方圖
            local_hist = calculate_histogram(local_region)

            #計算局部區域的 CDF
            num_pixels = local_region.size #.size回傳pixels數
            local_cdf = np.zeros(256 , dtype = float)
            local_cdf[0] = local_hist[0] / num_pixels
            for k in range(1, 256):
                local_cdf[k] = min(1, local_cdf[k-1] + local_hist[k] / num_pixels)

            #使用局部 CDF 建立轉換表
            local_lut = np.round(local_cdf * 255).astype(np.uint8)

            # 根據轉換表更新像素值
            locals_equlized_img[i, j] = local_lut[img[i, j]]

    plot_combined_histogram(img, locals_equlized_img, "Local Histogram Equalization")
    Local_Histogram_img_HPSNR = HPSNR(img, locals_equlized_img)
    print(f"The HPSNR value of Local_Histogram_img is: {Local_Histogram_img_HPSNR}")
    return locals_equlized_img
```

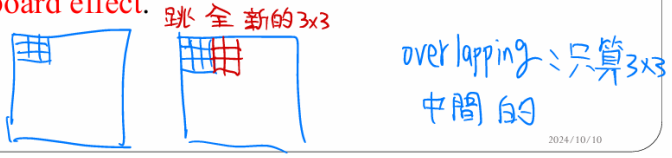
講義說明:

Local Enhancement 時複較高

- The histogram processing methods discussed above are global, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image.
- However, there are cases in which it is necessary to enhance details over small areas in an image.
- The procedure is to define a square or rectangular neighborhood and move the center of this area from pixel to pixel.

Local Enhancement

- At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained.
- This function is finally used to map the gray level of the pixel centered in the neighborhood, the center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated.
- Another approach used some methods to reduce computation is to utilize nonoverlapping regions, but this method usually produces an undesirable checkerboard effect.



(3) Use moments do Histogram Equalization:

步驟:
Step1: 劃分影像為局部區域
Step2: 計算局部影像的動差
Step3: 對影像進行標準化
Step4: 將標準化後的灰階值映射回 [0,255] 的灰階範圍。
Step5: 使用重新分佈的灰階值替代原始影像的像素值

Code:

```python
def adaptive_histogram_equalization(img, window_size=7): #Bonus 用動差進行 Histogram Equalization
    # Step 1: 初始化輸出影像
    height, width = img.shape
    enhanced_img = np.zeros_like(img, dtype=np.uint8)
    half_window = window_size // 2

    # Step 2: 對每個像素位置進行處理
    for i in range(height):
        for j in range(width):
            # 定義局部區域的範圍，考慮邊界
            row_min = max(0, i - half_window)
            row_max = min(height, i + half_window + 1)
            col_min = max(0, j - half_window)
            col_max = min(width, j + half_window + 1)

            # 提取局部區域
            local_region = img[row_min:row_max, col_min:col_max]

            # 計算局部平均值和變異數
            local_mean = np.mean(local_region)
            local_std = np.sqrt(np.var(local_region))  # 標準差

            # Step 3: 根據局部統計量調整像素值(做標準化)
            pixel_value = img[i, j]
            if local_std > 0:  # 避免除以 0
                normalized_value = (pixel_value - local_mean) / local_std
                enhanced_value = 128 + 64 * normalized_value  # 將值映射到 [0, 255] 範圍內
            else:
                enhanced_value = local_mean

            # 將結果裁切到 [0, 255] 範圍
            enhanced_img[i, j] = np.clip(enhanced_value, 0, 255)

    plot_combined_histogram(img, enhanced_img, "Adaptive Histogram Equalization")
    Adaptive_Histogram_img_HPSNR = HPSNR(img, enhanced_img)
    print(f"The HPSNR value of Adaptive_Histogram_img is: {Adaptive_Histogram_img_HPSNR}")

    return enhanced_img
```

講義說明:

Use of Histogram Statistics for Image Enhancement

2階动差 Variance

- Moments (動差) can be determined directly from a histogram much faster than they can from the pixels directly.
- Let r denote a discrete random variable representing discrete gray-levels in the range [0, L-1], and let $p(r_i)$ as an estimate of the probability of occurrence of gray level r_i. The nth moment of r about its mean is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$$

i=0~L-1 r_i

where m is the mean value of r (its average gray level) r_i = 像素值

$$m = \sum_{i=0}^{L-1} r_i p(r_i)$$

Use of Histogram Statistics for Image hancement

- It follows from $\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$ and $m = \sum_{i=0}^{L-1} r_i p(r_i)$ that $\mu_0 = 1$ and $\mu_1 = 0$.
- The second moment is given by

2階动差

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) = \sigma^2$$

- We recognize this expression as the variance of r, which is denoted conventionally by $\sigma^2(r)$. The standard deviation is defined simply as the square root of the variance.
- In terms of enhancement, we are interested primarily in the mean, which is a measure of average gray level in an image, and the variance, which is a measure of average contrast.

Use of Histogram Statistics for Image Enhancement

- We consider the use of the mean and variance (standard deviation) for enhancement purposes.
- The global mean and variance are measured over an entire image and are useful primarily for gross adjustments of overall intensity and contrast.
- The mean and standard deviation for a local region are useful for correcting for large-scale changes in intensity and contrast.

3. Discussion

(1) Global HE 跟 Local HE 優缺點比較:

Global HE:

優點：

- 計算速度較快，容易實作。
- 適合對於整張影像對比度提升的需求。

缺點：

- 忽略了區域細節：如果影像不同區域有不同的亮度，可能會導致某些區域效果不佳。例如在亮部和暗部都有豐富細節的影像，某些部分可能會被過度強化或壓縮。

Local HE:

優點：

- 更適合具有亮度或對比度不均的影像，可以提升局部細節。
- 適合影像中有陰影、光暈或亮暗分佈不均的情況。

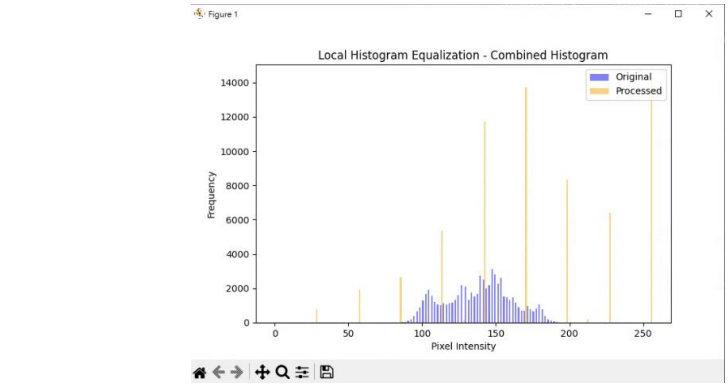
缺點：

- 計算量較大，處理時間較長。
- 可能會引入區塊效應（如果沒有平滑過渡），導致某些區塊之間出現不自然的邊界。

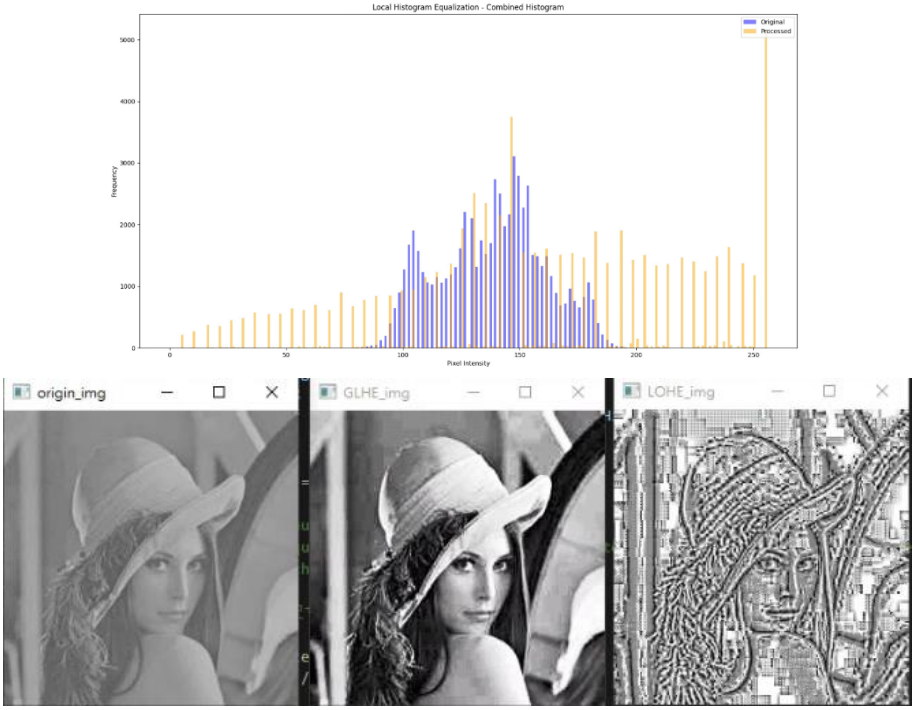
特性	Global Histogram	Local Histogram
範圍	全影像	區域影像（小區塊）
應用情境	全域對比增強，如亮度或對比調整	局部細節提升，如 CLAHE
處理效果	適合全影像亮度均一的場景	適合亮度或對比不均勻的影像
計算複雜度	較低	較高
潛在問題	可能無法處理亮度不均的影像	可能產生區塊效應

(2) Local HE 不同的 window size 效果:

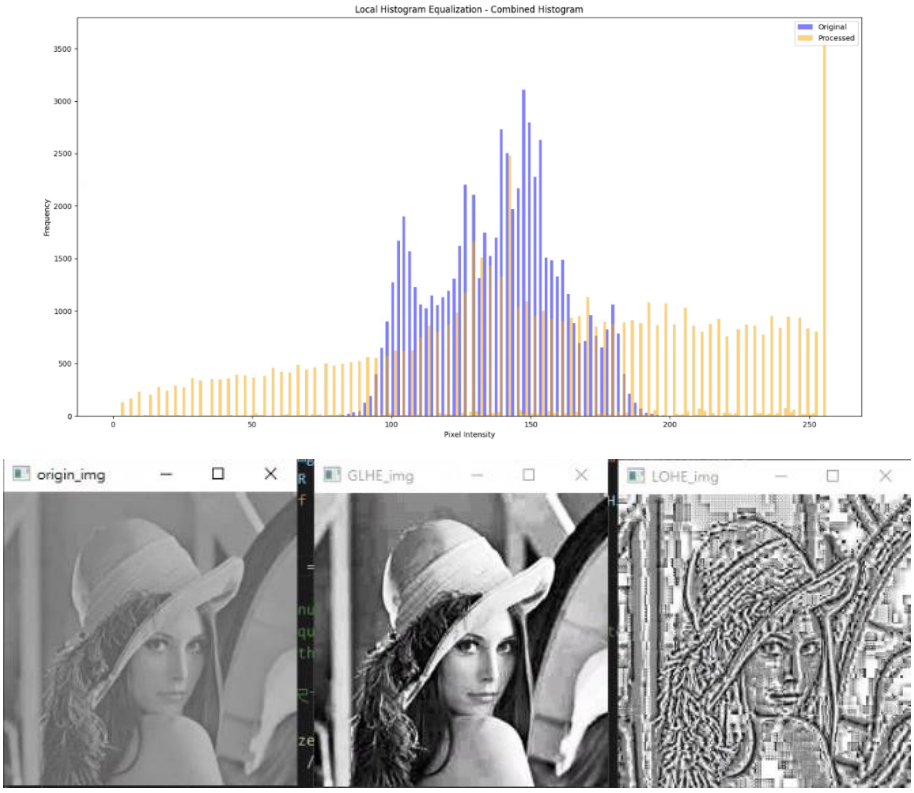
3x3



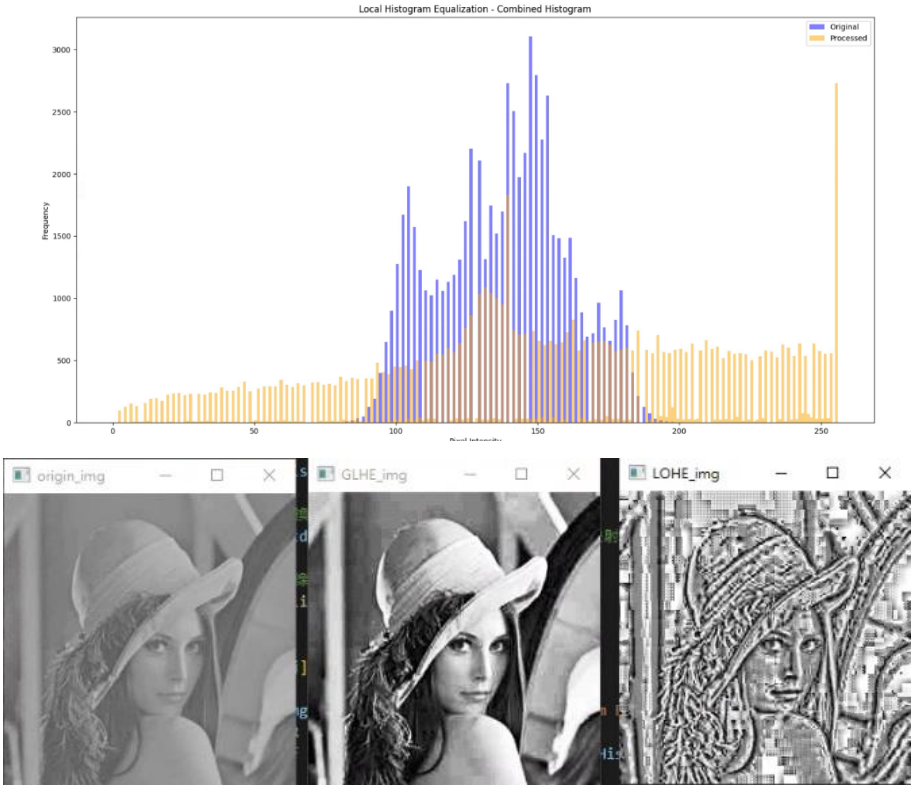
7x7



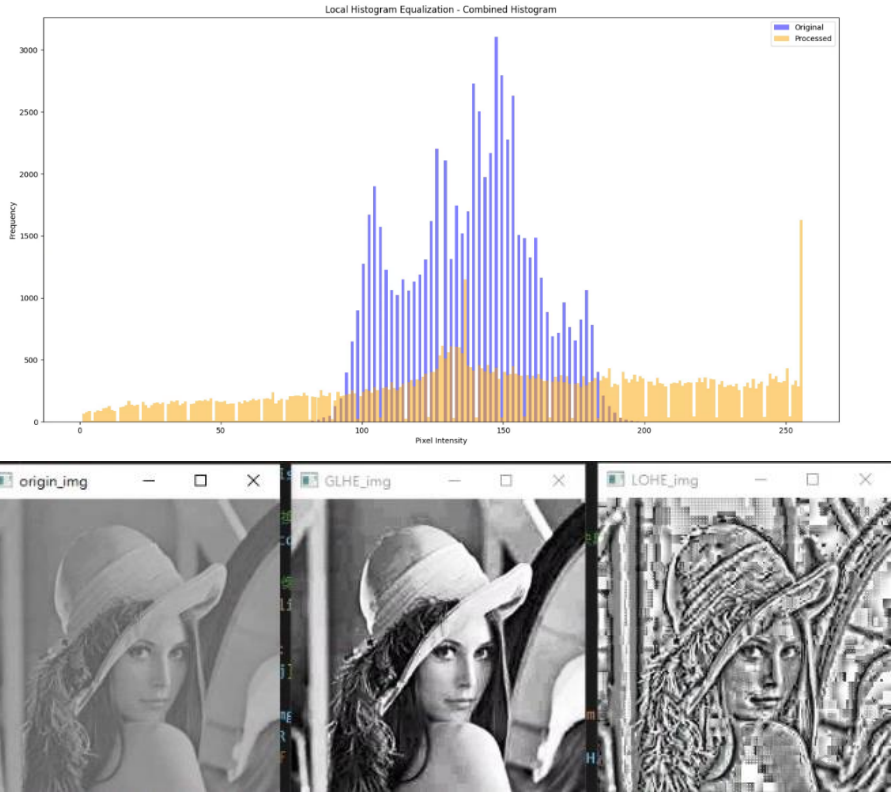
8*8


Local Histogram Equalization - Combined Histogram
Original
Processed
Frequency
Pixel Intensity
origin_img
GLHE_img
LOHE_img

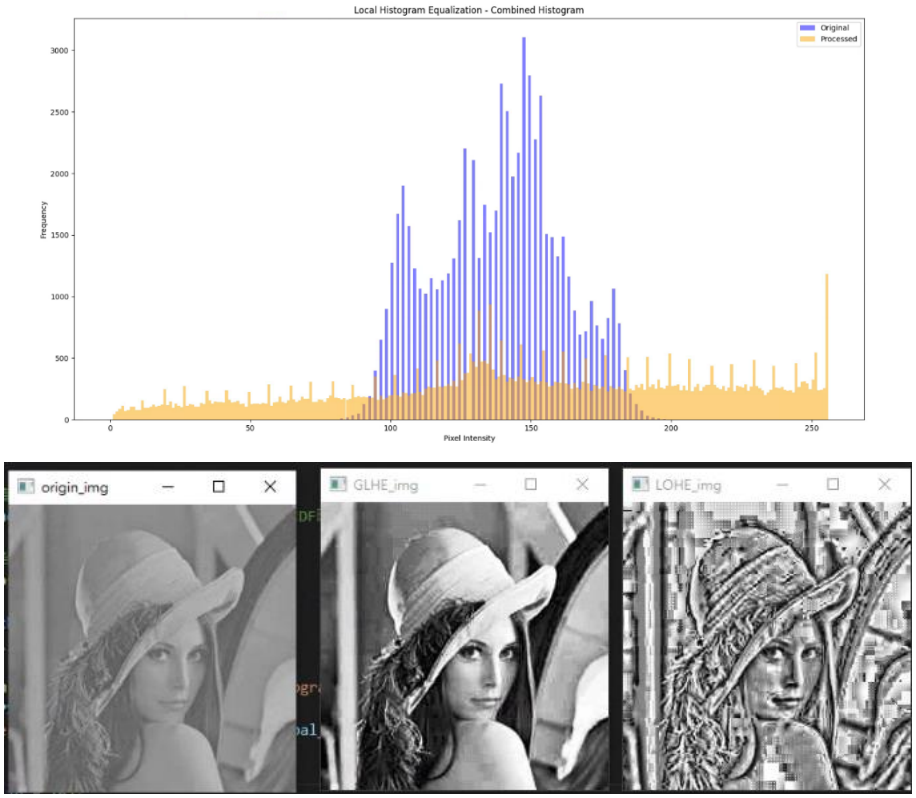
10x10


Local Histogram Equalization - Combined Histogram
Original
Processed
Frequency
origin_img
GLHE_img
LOHE_img

14x14


Local Histogram Equalization - Combined Histogram
Original
Processed
Frequency
Pixel Intensity
origin_img
GLHE_img
LOHE_img

16x16

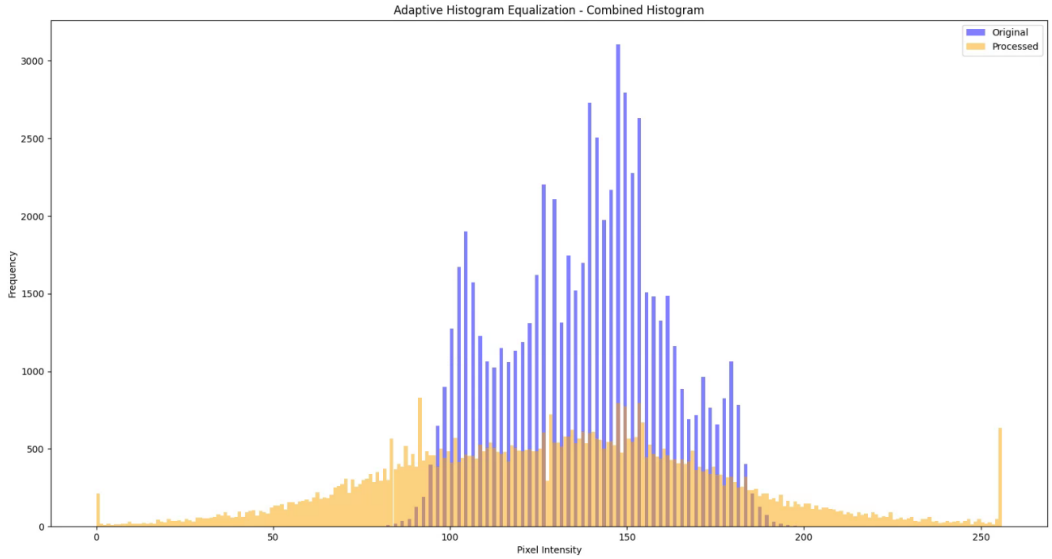

Local Histogram Equalization - Combined Histogram
Original
Processed
Frequency
Pixel Intensity
origin_img
GLHE_img
LOHE_img

可以看到在 window size 比較小的時候局部細節會比較突出，他更能解決全局光照不均的情況，但是缺點是容易產生區塊效應，導致影像在 window 邊界處會出現不連續的視覺效果。另外在光滑區域中（如大面積的背景），可能會放大噪聲，導致影像品質下降。

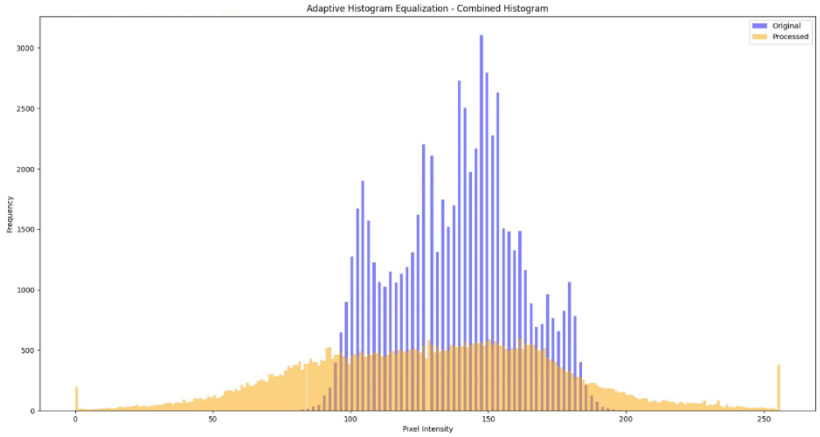
相對地在 window size 比較大的時候，他能夠更平滑地調整較大區域內的像素對比度，減少區塊效應的影響。但缺點是細節增強效果較弱，可能會忽略一些細小的局部特徵。另外當窗口過大時，可能接近 Global HE 的效果，這樣也失去局部增強的意義了。

(3) moments HE 不同的 window size 效果:

7x7

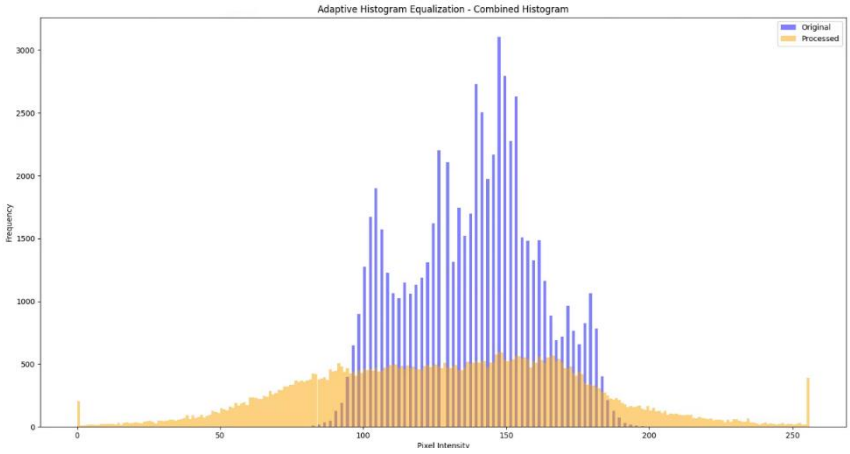


12x12


Adaptive Histogram Equalization - Combined Histogram
Original
Processed
Frequency
Pixel Intensity

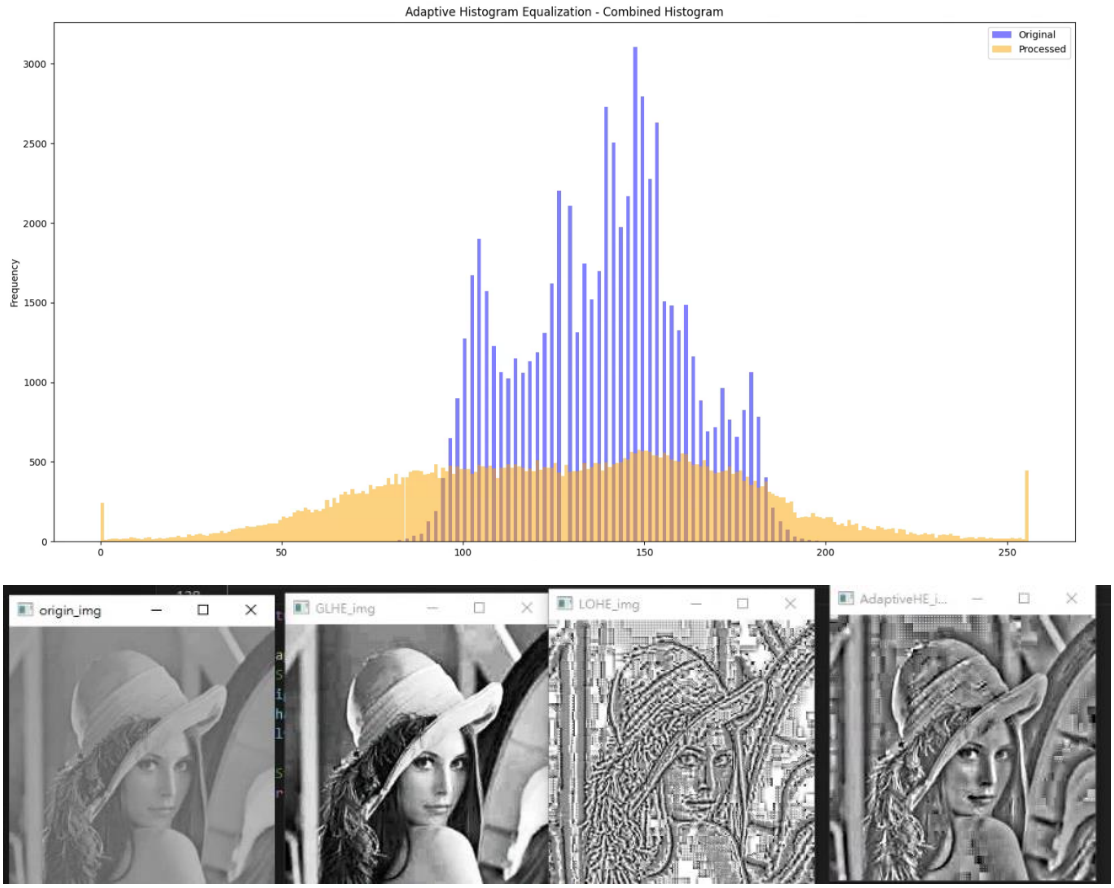

origin_img
GLHE_img
LOHE_img
AdaptiveHE_i...

16x16


Adaptive Histogram Equalization - Combined Histogram
Original
Processed
Frequency
Pixel Intensity


origin_img
GLHE_img
LOHE_img
AdaptiveHE_i...

20x20



Window 大小影響跟剛剛的 Local HE 大同小異。Window size 小的時候每個窗口內的動差計算範圍較小，主要反映局部區域的特徵。對比度會根據小窗口內的灰階分佈進行細緻的增強。一樣可能會導致區塊效應 。Window size 大的時候窗口能捕捉更廣泛的區域統計特性。每個窗口內的動差計算範圍較大，反映的是全局性的特徵。這樣局部對比度的增強會更平滑，減少區塊效應。但由於窗口範圍較大，會淡化局部細節的特徵，導致細節部分的對比度增強效果不明顯。

(4) 減緩區塊效應方法:

在 Local HE 中，為了減緩區塊效應，可以在直方圖等化後對影像進行平滑濾波處理。這樣可以平滑區域之間的邊界，減少不連續的區塊痕跡。以下是幾種常用的濾波方法，適用於減少區塊效應：

1. 高斯模糊 (Gaussian Blur)
2. 中值濾波 (Median Blur)
3. 均值濾波 (Averaging/Box Blur)

4. Github

https://github.com/Labibibidu/DIP/tree/4e9d0534f2fd0c8dc79b911199fcb743d1f704bc/HW2