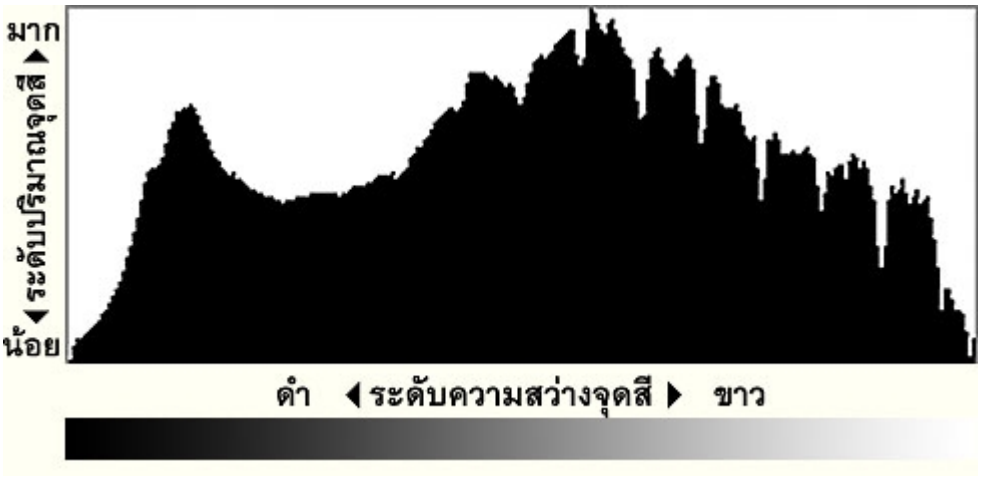
ปฏิบัติการ

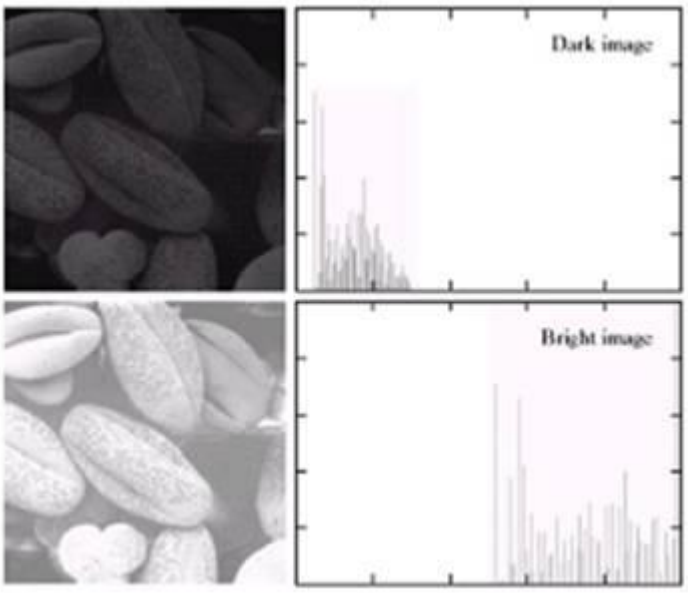
Digital Image Processing and Computer Vision Lab 2 – Histogram

1. Histogram

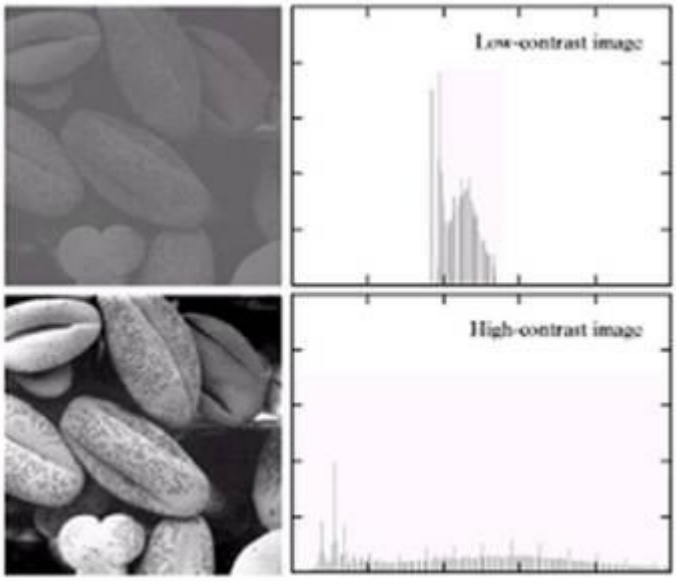


เป็นกราฟที่แสดงจำนวน pixels ในแต่ละความสว่างต่าง ๆหรือข้อมูลค่าสี R,G,B ของรูปภาพ digital ในภาพ gray scale ในแกนนอนจะแสดงความสว่างดังกล่าว ซึ่งมีความสว่างตั้งแต่ 0-255 (แบ่งเป็น 256 ระดับความแตกต่างสี) โดยทางด้านซ้ายของกราฟจะมีค่าความสว่างน้อย ภาพจะมีสีเข้มเข้าใกล้สีดำ ส่วนทางด้านขวามือจะมีความสว่างสูง ภาพจะสว่างเข้าใกล้สีขาว ส่วนบริเวณตรงกลางกราฟแสดงส่วนน้ำหนักสีกลาง ส่วนในแนวแกนตั้งจะแสดงจำนวน pixels ในแต่ละความสว่างซึ่งในแกนตั้งนี้ ไม่มีขอบเขตจำกัด ถ้าหากภาพมีความมืดมาก กราฟจะไปกองรวมกันทางด้านซ้ายมือ โดยที่ไม่มีขอบเขตจำกัด

คุณสมบัติของ Histogram



หากการกระจายส่วนใหญ่อยู่ทางด้านซ้ายของกราฟ แสดงว่าภาพนั้นมีความสว่างของภาพน้อย ในทางกลับกัน หากการกระจายส่วนใหญ่อยู่ทางด้านขวาของกราฟ แสดงว่าภาพนั้นมีความสว่างของภาพมาก



หากการกระจายของกราฟเป็นกลุ่มแคบ ๆ แสดงว่าภาพนั้นเป็นภาพที่ Low-contrast และหากการกระจายของกราฟมีการกระจายอย่างสม่ำเสมอทั่วทั้งกราฟ แสดงว่าภาพนั้นเป็นภาพที่ High-contrast

ประโยชน์ของ Histogram

1. ภาพที่มองจากจอภาพอาจมีการคลาดเคลื่อนได้ แต่ข้อมูลจาก histogram จะบอกความสว่างความเข้มของรูปภาพได้อย่างแท้จริง

2. ข้อมูลนี้จะช่วยให้เราเลือกโหมดในการถ่ายภาพได้ดียิ่งขึ้น โดยช่วยในการเลือกการชดเชยแสงของภาพเมื่อต้องถ่ายภาพในที่ที่มีความสว่างของภาพสูงหรือต่ำมากได้

3. สามารถนำข้อมูลมาใช้ประกอบในการประมวลผลและปรับแต่งภาพได้

```
procedure FillHistogram()

for each b ∈ H
    b ← b × α

for each x ∈ I
    if x > T
        for each b ∈ h(x)
            b ← b + (1 − α)
```

Idea for histogram computation algorithm.

```
Function CreateHistogram(image):
    If image is colored (RGB):
        Separate the image into three channels: Red, Green, and Blue
        Initialize three arrays to store pixel counts for each channel:
            histogramRed[256] = {0}
            histogramGreen[256] = {0}
            histogramBlue[256] = {0}

        For each pixel in the image:
            Extract Red, Green, Blue values from the pixel
            Increment histogramRed[Red value] by 1
            Increment histogramGreen[Green value] by 1
            Increment histogramBlue[Blue value] by 1

        Return histogramRed, histogramGreen, histogramBlue

    Else if image is grayscale:
        Initialize an array to store pixel counts:
            histogramGray[256] = {0}

        For each pixel in the image:
            Extract the pixel intensity value
            Increment histogramGray[intensity value] by 1

        Return histogramGray

    Else:
        Print "Unsupported image format"
        Return None

End Function
```

Pseudocode for histogram computation algorithm.

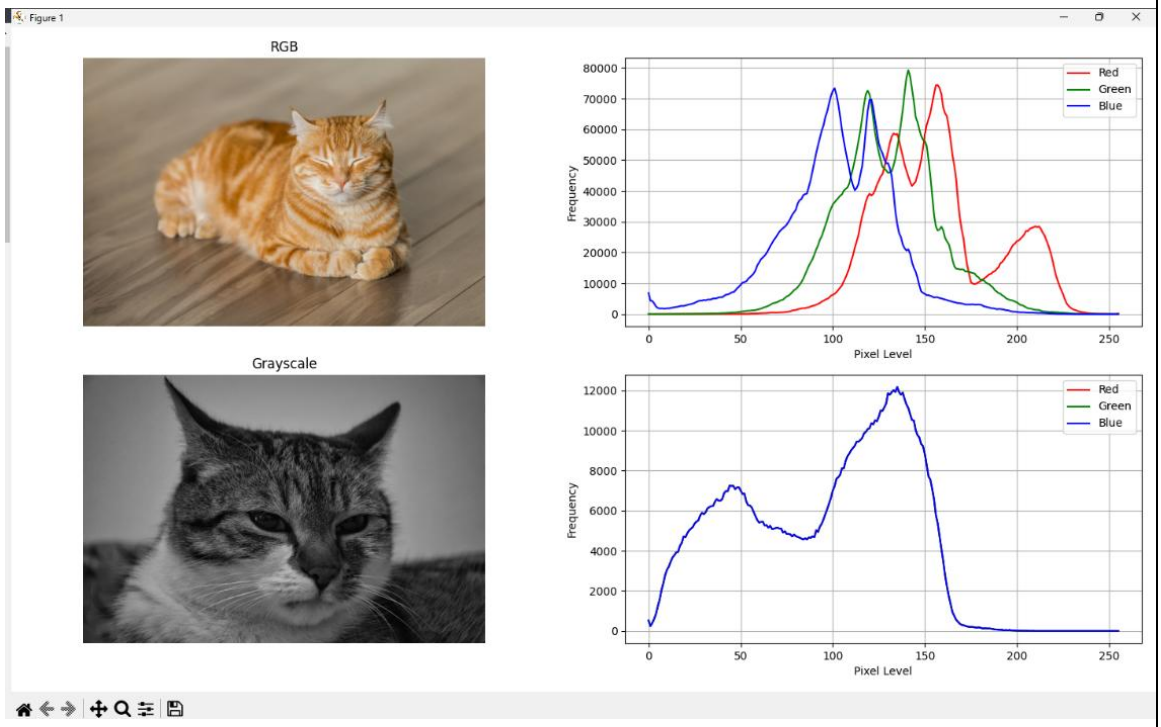
```
from PIL import Image
```

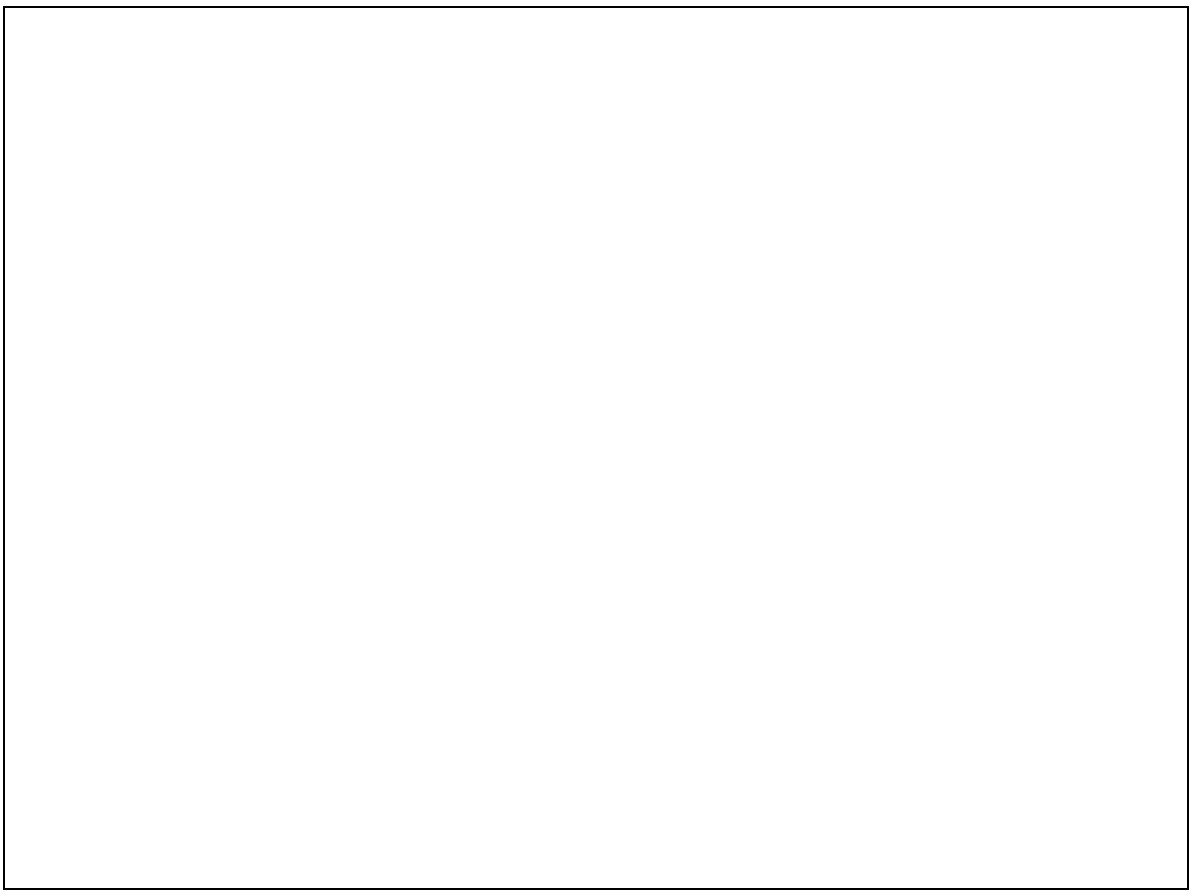
```
import numpy as np

def create_histogram(image_path):
    # Load the image
    image = Image.open(image_path)
    data = np.array(image)

    if len(data.shape) == 3:
        # Initialize histograms
        histogramRed = np.zeros(256)
        histogramGreen = np.zeros(256)
        histogramBlue = np.zeros(256)

        # Update histograms
        for row in data:
            for pixel in row:
                r, g, b = pixel[:3]
                histogramRed[r] += 1
                histogramGreen[g] += 1
                histogramBlue[b] += 1

        return histogramRed, histogramGreen, histogramBlue

    elif len(data.shape) == 2:
        # Initialize histogram
        histogramGray = np.zeros(256)

        # Update histogram
        for row in data:
            for pixel in row:
                histogramGray[pixel] += 1

        return histogramGray

    else:
        print("Unsupported image format")
```

```
        return None


# Example usage:
histRed, histGreen, histBlue = create_histogram('path_to_color_image.jpg')
histGray = create_histogram('path_to_grayscale_image.jpg')
```

Coding for histogram computation algorithm.

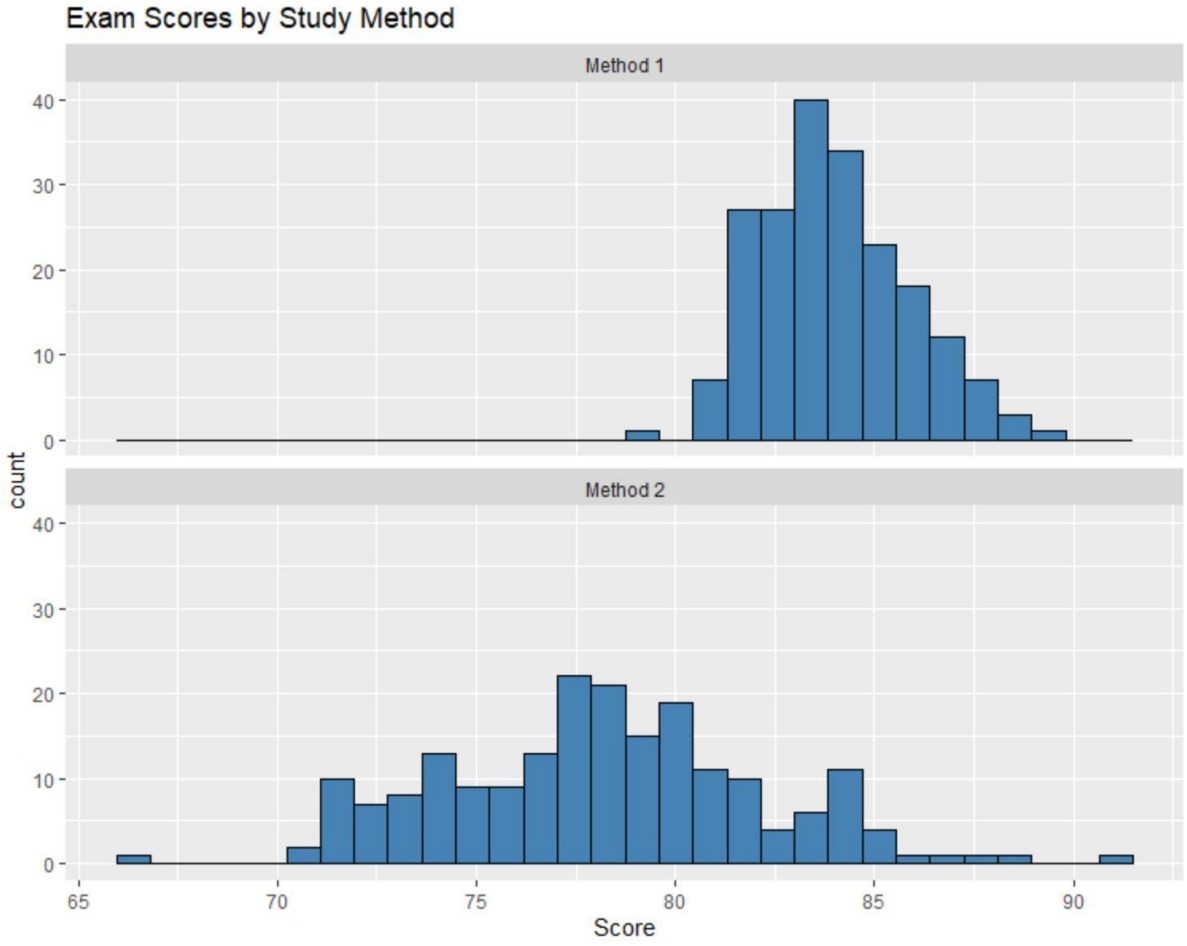
#1 [แก้ไขโค้ดให้แสดงภาพจากการทำ Histogram ทั้งภาพสีและภาพระดับเทา จากนั้นแคปภาพผลลัพธ์แปะในกรอบด้านล่าง **เลือกใช้รูปจากอินเตอร์เน็ต]





แก้ไขโค้ดสำหรับภาพแบบ Grayscale และแสดงผลลัพธ์

2. Histogram Comparison



- To compare two histograms (H_1 and H_2), first we have to choose a *metric* ($d(H_1, H_2)$) to express how well both histograms match.
- OpenCV implements the function **cv::compareHist** to perform a comparison. It also offers 4 different metrics to compute the matching:

1. **Correlation (CV_COMP_CORREL)**

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}}$$

where

$$\bar{H}_k = \frac{1}{N} \sum_J H_k(J)$$

and N is the total number of histogram bins.

2. **Chi-Square (CV_COMP_CHISQR)**

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

3. **Intersection (method=CV_COMP_INTERSECT)**

$$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I))$$

4. **Bhattacharyya distance (CV_COMP_BHATTACHARYYA)**

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}$$

1. Start

2. Load Image1 and Image2
   - Ensure both images are in the same color space (e.g., RGB, grayscale)

3. Calculate Histogram for Image1
   - For each pixel in Image1:
     - Increment the histogram bin corresponding to the pixel's value
   - Normalize the histogram (so that the sum of all bins equals 1)

4. Calculate Histogram for Image2
   - Repeat steps similar to step 3 for Image2

5. Compare Histograms
   - Initialize similarity_measure to 0
   - For each bin in the histograms:
     - Calculate the difference between corresponding bins of Image1 and Image2
     - Optionally, use a similarity metric such as:
       - Chi-Squared Test
       - Intersection
       - Bhattacharyya Distance
       - Correlation
     - Update the similarity_measure based on the metric

6. Output the similarity_measure
   - A higher value typically indicates more similarity (depends on the metric used)

7. End

Pseudocode for histogram comparison algorithm.

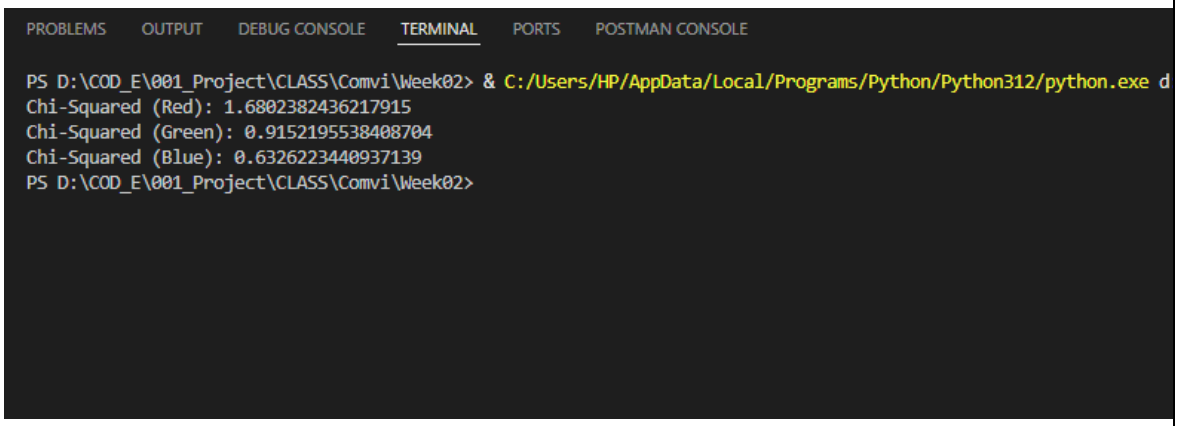
Function CompareHistogramsChiSquared(histogram1, histogram2):

```
Initialize chi_squared_value = 0

For each bin (bin1, bin2) in histogram1, histogram2:
    Calculate expected_value = (bin1 + bin2) / 2

    If expected_value ≠ 0:
        Calculate squared_difference = (bin1 - expected_value)²
        Calculate contribution = squared_difference / expected_value
        Add contribution to chi_squared_value

Return chi_squared_value
```

Pseudocode for Chi-Squared Test algorithm.

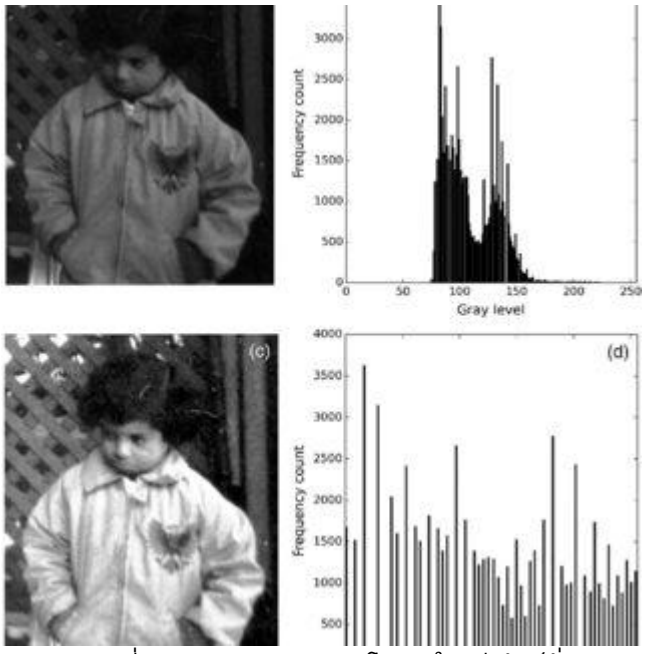
```
def compare_histograms_chi_squared(histogram1, histogram2):
    chi_squared_value = 0.0  # Initialize chi-squared value

    # Iterate through bins in both histograms
    for bin1, bin2 in zip(histogram1, histogram2):
        expected_value = (bin1 + bin2) / 2  # Calculate the expected value

        if expected_value != 0:  # Avoid division by zero
            squared_difference = ....  # Squared difference
            contribution = .....  # Contribution to chi-squared
            chi_squared_value += contribution  # Add contribution to the total

    return chi_squared_value
```

#2 จงเขียนแก้ไขโค้ดเพื่อพิสูจน์อัลกอริทึม Chi-Squared Test พร้อมแคปโค้ดที่แก้ไขพร้อมผลการทดลองแปะในกรอบด้านล่าง




```
for bin1, bin2 in zip(histogram1, histogram2):
        expected_value = (bin1 + bin2) / 2  # Calculate the expected value

        if expected_value > 0:  # Avoid division by zero
            squared_difference = (bin1 - bin2) ** 2  # Squared difference
            contribution = squared_difference / expected_value  # Contribution to chi-squared
            chi_squared_value += contribution  # Add contribution to the total

    return chi_squared_value
```

3. Histogram Equalization



การปรับ contrast ของภาพด้วยการเปลี่ยนแปลงค่าของ pixel โดยจะใช้วิธี ฟังชั่นการกระจายสะสม ในการทำให้ภาพเกิดความสมดุลและ ภาพมี dynamic สำหรับภาพ หลังจากการทำ equalization ซึ่งวิธีการนี้มีทั้งข้อดีและข้อเสีย โดยข้อดีของวิธีการนี้คือ การปรับภาพให้ชัดเจนมากขึ้นและทำให้ภาพมีความสมดุลมีนความลึกของภาพ ส่วนข้อเสียนั้น การปรับภาพวิธีการนี้ จะทำให้ pixel ผิดแปลกไปจากเดิมในบางกรณี

```
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread("data/test_2.jpg")
if img is None:
    print('Could not open or find the image:',)
    exit(0)
## [Resize image]
scale_percent = 100 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
src= cv.resize(img, dim, interpolation = cv.INTER_AREA)

## [Convert to grayscale]
src = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
## [Convert to grayscale]

## [Apply Histogram Equalization]
dst = cv.equalizeHist(src)
## [Apply Histogram Equalization]

## [Display results]
cv.imshow('Source image', src)
cv.imshow('Equalized Image', dst)
## [Display results]

## [Wait until user exits the program]
cv.waitKey()
## [Wait until user exits the program]
```

โค้ดตัวอย่างการทำ histogram equalization

#3จงแคปผลลัพธ์ของการทำงานด้วยรูปภาพของตนเอง

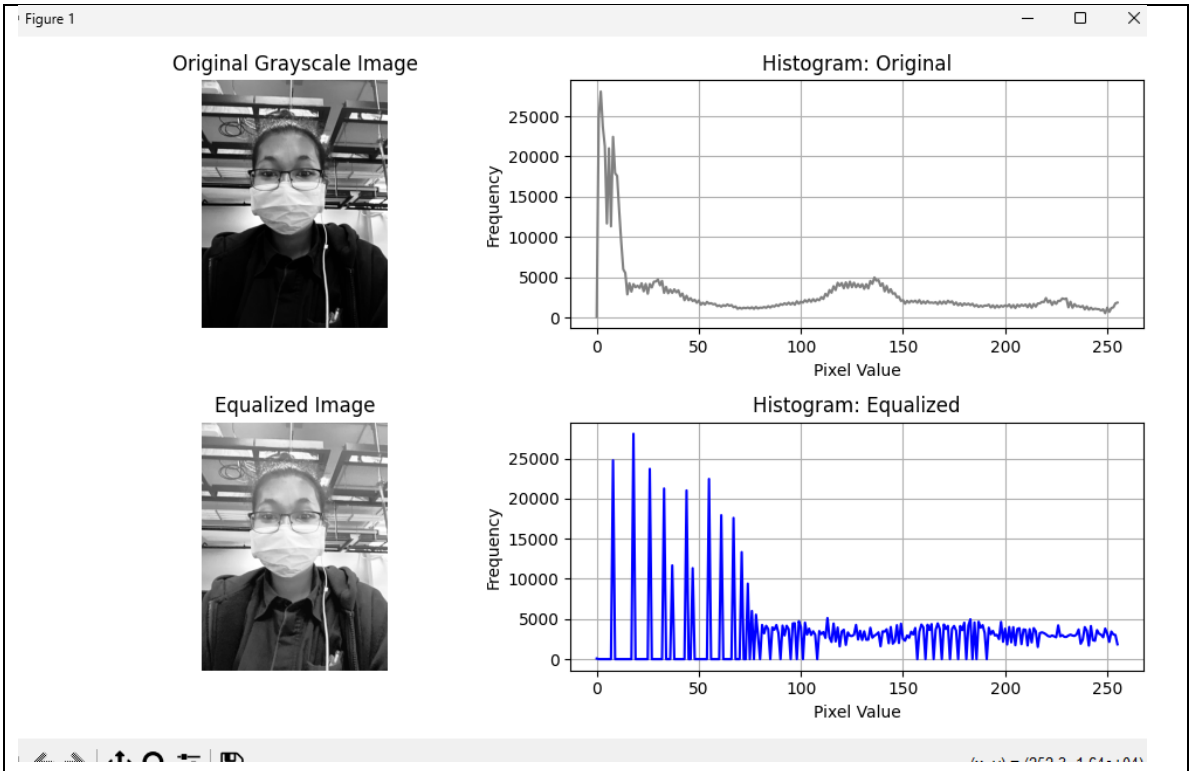


Figure 1
Original Grayscale Image
Histogram: Original
Frequency
Pixel Value
Equalized Image
Histogram: Equalized
Frequency
Pixel Value