Code Section

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
def adjust_gamma(image, gamma=0.5):
    table = np.array([((i / 255.0) ** gamma) * 255 for i in np.arange(0,
256)]).astype("uint8")
    return cv2.LUT(image, table)

def global_histogram_equalization(image):
    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # 1. Compute the histogram
    hist, _ = np.histogram(gray.flatten(), 256, [0,256])
    # 2. Compute the cumulative distribution function (CDF)
    cdf = hist.cumsum()
    # Normalize the CDF to fit in the 0-255 range (8-bit image)
    cdf_normalized = ((cdf - cdf.min()) * 255) / (cdf.max() - cdf.min())
    cdf_normalized = cdf_normalized.astype(np.uint8)
    # 3. Use the CDF to equalize the image
    # The power of this operation is that it performs the entire
transformation
    # for all pixels in the image simultaneously
    equalized_img = cdf_normalized[gray]
    return equalized_img

def local_histogram_enhancement(image, ksize, E=4, k1=0.4, k2=0.02,
k3=0.4):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Global mean and standard deviation
    M_G = np.mean(gray)
    D_G = np.std(gray)
    # Create a padded version of the original for border processing
    padded = np.pad(gray, ksize//2, mode='reflect')
    enhanced_img = np.zeros_like(gray, dtype=np.uint8)
    for i in range(gray.shape[0]):
        for j in range(gray.shape[1]):
            # Extracting local neighborhood
            local_region = padded[i:i+ksize, j:j+ksize]
            # Local mean and standard deviation
            m_S = np.mean(local_region)
            s_S = np.std(local_region)
            # Applying the given conditions
            if m_S <= k1 * M_G and k2 * D_G <= s_S <= k3 * D_G:
                enhanced_img[i, j] = E * gray[i, j]
            else:
                enhanced_img[i, j] = gray[i, j]
    return enhanced_img
```

```
image = cv2.imread("./assignment2_image1.jpg", cv2.IMREAD_COLOR)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert to RGB for
matplotlib
# 1. Gamma Correction
gamma_corrected = adjust_gamma(image, 0.5)
# 2. Global Histogram Equalization
global_hist_eq = global_histogram_equalization(image)
# 3. Local Histogram Equalization
local_hist_eq_3x3 = local_histogram_enhancement(image, 3)
local_hist_eq_7x7 = local_histogram_enhancement(image, 7)
local_hist_eq_11x11 = local_histogram_enhancement(image, 11)

# Displaying the enhanced images using matplotlib
plt.figure(figsize=(15,10))

plt.subplot(2, 3, 1)
plt.imshow(image_rgb)
plt.title('Original Image')

plt.subplot(2, 3, 2)
plt.imshow(gamma_corrected, cmap='gray')
plt.title('Gamma Corrected')

plt.subplot(2, 3, 3)
plt.imshow(global_hist_eq, cmap='gray')
plt.title('Global Histogram Equalization')

plt.subplot(2, 3, 4)
plt.imshow(local_hist_eq_3x3, cmap='gray')
plt.title('Local Histogram Equalization 3x3')

plt.subplot(2, 3, 5)
plt.imshow(local_hist_eq_7x7, cmap='gray')
plt.title('Local Histogram Equalization 7x7')

plt.subplot(2, 3, 6)
plt.imshow(local_hist_eq_11x11, cmap='gray')
plt.title('Local Histogram Equalization 11x11')

plt.tight_layout()
plt.show()
```

คำอธิบาย code อย่างย่อ

1. ฟังก์ชั่น adjust_gamma

   ปรับค่าแกมมาของภาพที่ป้อนเพื่อเพิ่มความสว่างหรือความมืด โดยถ้า gamma น้อยภาพจะสว่างมากขึ้น หลักการทำงานคือ คำนวณตามpower-law transformations โดยมีการทำ normalized ตามปกติ แล้วทำการสร้างตารางการค้นหา (Look Up table) เพื่อที่จะได้หยิบค่ามาใช้ได้เลย ไม่ต้องคำนวณตัวที่ซ้ำกันตลอด

2. ฟังก์ชั่น global_histogram_equalization

   ทำการใช้ HE เพิ่มcontrastของภาพส่งผลให้ภาพมีระดับสีขาวและดำที่มองเห็นได้ชัดเจน หลักการของโค้ดนี้ คือ คำนวณฮิสโตแกรมของภาพระดับสีเทา สร้างการแจกแจงสะสมแบบมาตรฐาน (CDF) โดยยึดตาม histogram และใช้ CDF ที่ทำไว้จับคู่กับตำแหน่งpixelนั้นๆ โดยมีการทำ normalized ตามปกติ

3. ฟังก์ชั่น local_histogram_enhancement

   จะปรับปรุงภาพในlocalตามเงื่อนไขที่ระบุ โดยในที่นี้ (ทำตาม lecture) ก็คือจะพยายามที่จะไม่เปลี่ยนแปลงบริเวณที่สว่างอยู่แล้วแต่จะไปทำให้บริเวณที่มืดทำให้ภาพที่ความสว่างให้ชัดเจนมากขึ้น ตามฟังก์ชั่น

$$g(x,y) = \begin{cases} E \times f(x,y) & \text{if } m_S \le k_1 M_G \text{ and } k_2 D_G \le s_S \le k_3 D_G \\ f(x,y) & \text{otherwise} \end{cases}$$

Where:

E, k_1, k_2, k_3 are specified constants

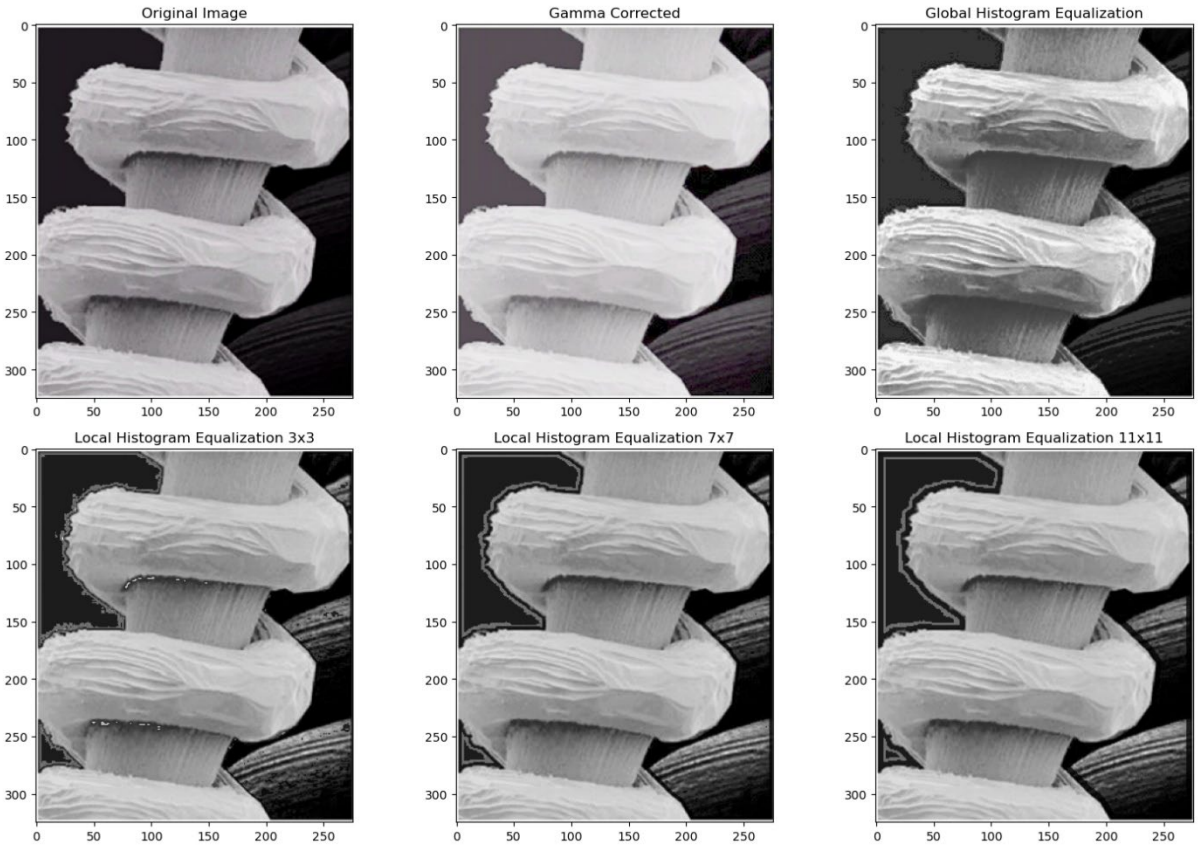
M_G is the global mean of the image

D_G is the global standard deviation

m_S is the local mean

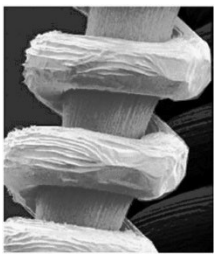
s_S is the local standard deviation.

ในกรณีของlocal HE จะกำหนดให้ใช้ E=4.0, k1=0.4, k2=0.02และ k3=0.4 ซึ่งจะได้ผลลัพธ์ดังนี้



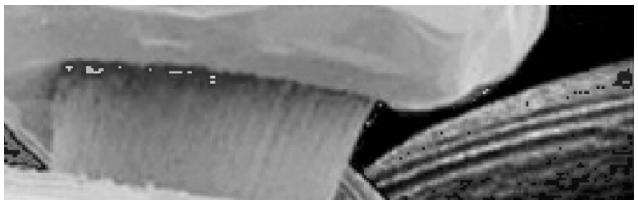
1.) ทำ Gamma Correction วิธีการนี้จะปรับระดับความสว่างของภาพทั้งหมด ซึ่งจะสังเกตเห็นการเปลี่ยนแปลงในด้านขวาซึ่งภาพดังเดิมมืดแต่ตอนนี้เห็นรายละเอียดมากขึ้น แต่อย่างไรก็ตาม สิ่งสำคัญคือ การปรับปรุงนี้ไม่ได้จำกัดอยู่เพียงบริเวณที่มืดเท่านั้น ภาพทั้งหมด รวมถึงบริเวณที่สว่างอยู่แล้ว ก็จะสว่างขึ้นเช่นกัน ซึ่งทำให้บางกรณี เช่นหากเราต้องการคงความสว่างไว้ในบริเวณที่มีแสงสว่างเพียงพออยู่แล้ว ก็จะทำไม่ได้

2.) ทำ global histogram equalization การใช้วิธีนี้จะทำการกระจายค่าความเข้มให้ขยายออกไป พุดง่ายๆก็คือ ทำให้contrastสูงมากขึ้นตามที่อธิบายไปแล้ว เมื่อเราใช้วิธีการนี้ เราก็จะเห็นถึงการกระจายความสว่างที่สมดุลมากขึ้น ทำให้บริเวณที่มืดและสว่างสามารถแยกแยะได้ดีขึ้น แต่วิธีการนี้ได้ผลดีกับภาพ ที่มีทั้งบริเวณที่มืดและสว่างเกินไป ซึ่งจากรูปเส้นใยเหล็กของเราภาพเส้นใยตรงกลางภาพ แสดงภาพได้ดีแล้ว พอเราทำวิธีนี้กลับทำให้ภาพเส้นใยดูไม่เหมือนจริง



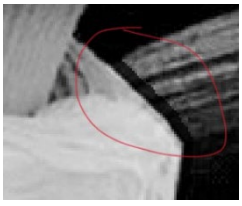
3.) Local histogram equalization ซึ่งเราแบ่งเป็น3แบบคือใช้ 3 neighborhood sizes ได้แก่ 3x3, 7x7, และ 11x11

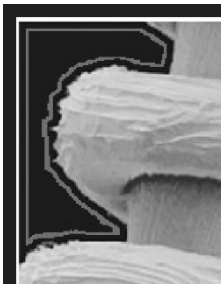
a. 3x3 – จะทำให้สำหรับแต่ละpixelในรูปภาพ พื้นที่เล็กๆ รอบๆ จะได้รับการพิจารณาสำหรับการปรับปรุง ทำให้สำหรับขนาดเล็กที่สุดนี้นี้จะดึงรายละเอียดปลีกย่อยออกมา อย่างไรก็ตาม จากการสังเกต ผมพบว่าภาพที่ได้จะมี noise ที่มาก อาจจะเนื่องจากพื้นที่ขนาดเล็กอาจได้รับอิทธิพลจากสีที่อยู่รอบๆข้างน้อยทำให้ตอนปรับภาพจึงมีสีอื่นเข้ามาแทรก(แสดงภาพทางซ้าย) แต่โดยรวมแล้วภาพที่ได้ก็เห็นรายละเอียดที่มองไม่ค่อยชัดในตอนแรกได้ดีขึ้น(แสดงภาพทางขวา)



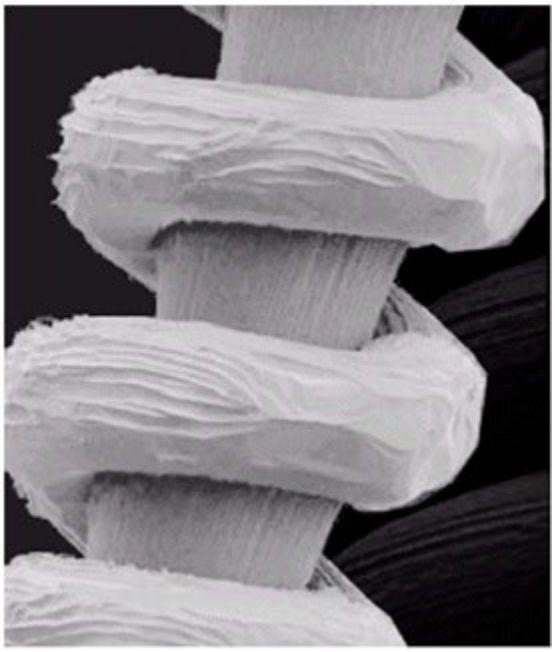
b. 7x7 - การเพิ่ม neighborhood size ให้เป็น 7x7 จะสามารถกำจัด noise ที่พูดถึงใน 3x3 ได้ เพราะว่าเมื่อเทียบกับหน้าต่าง 3x3 จุดขาวหรือจุดดำที่มีอยู่ได้หายไปแล้ว ซึ่งก็อาจจะเป็นเพราะเมื่อเราเพิ่มขนาดให้ใหญ่ขึ้น ภาพก็จะยังคงรักษาโครงสร้างบางส่วนให้เหมือนกับบริเวณรอบข้างทำให้เรา noise ลดลงไป

c. 11x11 – สำหรับขนาด neighborhood ที่ใหญ่ที่สุด คล้ายกับการใช้ neighborhood ขนาด 7x7 เพียงแต่ว่ามีจุดที่สังเกตได้คือ บริเวณขอบของภาพจะเห็นว่ามีเส้นขอบที่เกิดขึ้นจากการคำนวณ histogram ซึ่งจาก 7x7 ขนาดของเส้นขอบยังไม่ใหญ่มาก แต่พอมา11x11แล้วจะเห็นเส้นขอบที่หนาจนเห็นได้ชัด ซึ่งก็จะทำให้รายละเอียดของภาพบางส่วนก็จะหายไปด้วยเช่นกัน





ดังนั้น จึงคิดว่าภาพที่ดีที่สุดมาจากการใช้วิธี local histogram equalization ขนาด 7x7 neighborhood size เพราะได้ภาพที่อยู่ด้านหลังที่ชัดเจน เส้นขอบไม่หนาจนเกินไป (ทำให้ในอนาคตเราอาจหาวิธีในการลบเส้นขอบนี้ออกไปได้ง่าย) เปรียบเทียบอีกครั้งกับภาพต้นฉบับด้านล่าง



original image

local histogram equalization with 7x7 neighborhood size