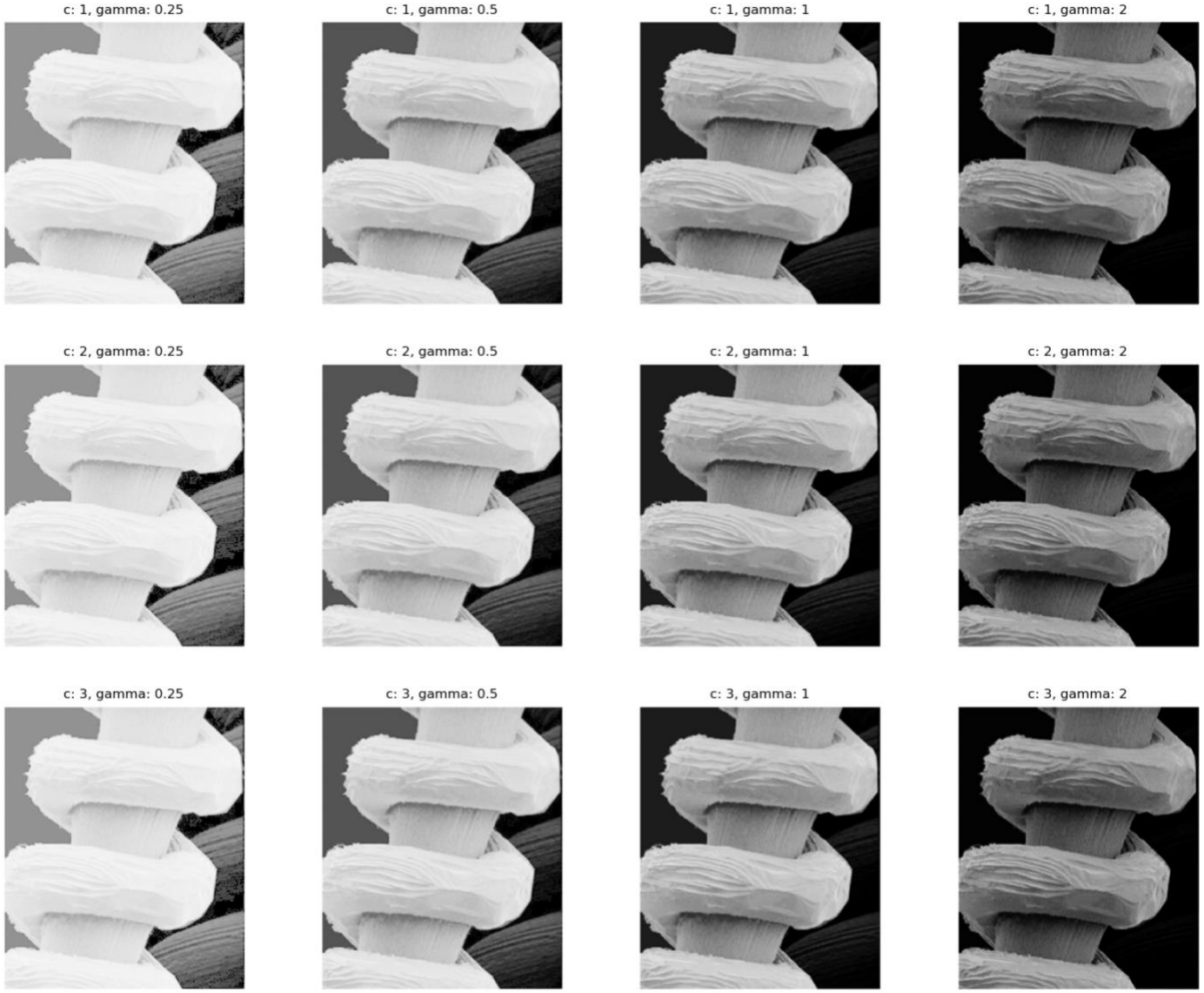
Image Enhancement

Source code Available on GitHub :
https://github.com/ratchanonp/imageprocessing/blob/main/Assignment_2/assignment_2.ipynb

1. **Gamma Correction**
   Source Code:

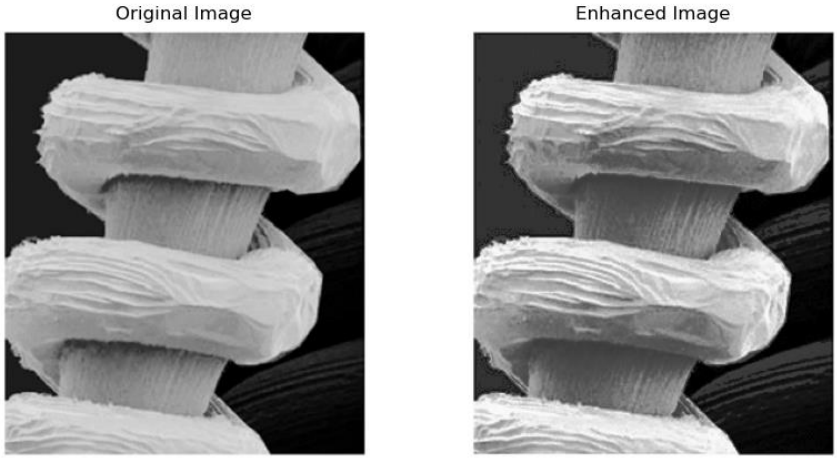
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def gamma_correction(img, c, gamma):
    img = img.astype(np.float32)
    img = c * img ** gamma

    # Normalize the image
    img = img / np.max(img) * 255

    return img.astype(np.uint8)

# %% [markdown]
# # Image Enhancements with Gamma correction
#

# %%
img_path = "./assignment2_image1.jpg"
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

c_vals = [1, 2, 3]
gamma_vals = [0.25, 0.5, 1, 2]

row, col = len(c_vals), len(gamma_vals)
plt.figure(figsize=(20, 16))


# Apply gamma correction
for c in range(len(c_vals)):
    for gamma in range(len(gamma_vals)):
        c_val, gamma_val = c_vals[c], gamma_vals[gamma]
        img_corrected = gamma_correction(img, c_val, gamma_val)

        plt.subplot(row, col, c * col + gamma + 1)
        plt.imshow(img_corrected, cmap="gray")
        plt.axis("off")
        plt.title("c: {}, gamma: {}".format(c_val, gamma_val))

plt.savefig("./assignment2_image1_gamma_correction.png")
plt.show()
```

ผลลัพธ์ที่ได้


c: 1, gamma: 0.25
c: 1, gamma: 0.5
c: 1, gamma: 1
c: 1, gamma: 2
c: 2, gamma: 0.25
c: 2, gamma: 0.5
c: 2, gamma: 1
c: 2, gamma: 2
c: 3, gamma: 0.25
c: 3, gamma: 0.5
c: 3, gamma: 1
c: 3, gamma: 2

2. Global Histogram Equalization
Source Code:

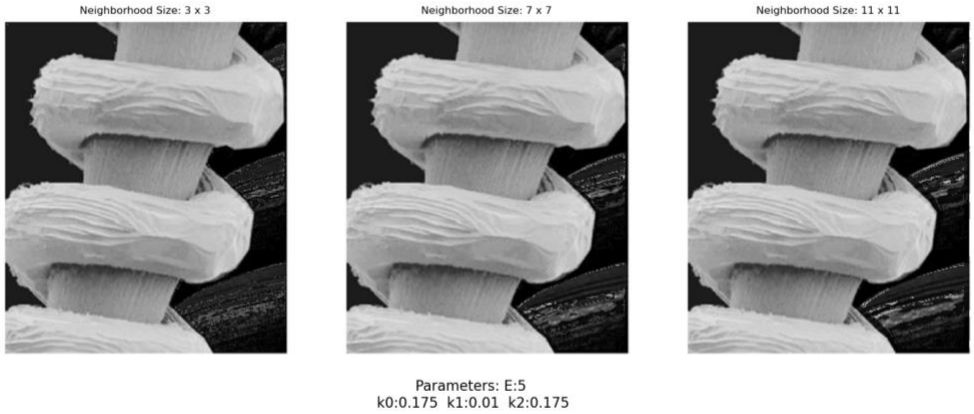
```
import cv2
import matplotlib.pyplot as plt

def globalHistogramEqualiztion(img):

    # Calculate the histogram
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])

    # Calculate the cumulative sum
    cdf = hist.cumsum()

    # Normalize the cdf
    cdf_normalized = cdf * hist.max() / cdf.max()

    # Calculate the new pixel values
    img_new = cdf_normalized[img]

    return img_new

enchanced_img = globalHistogramEqualiztion(img)

plt.figure(figsize=(10, 5))

# Plot the image and subplot histogram
plt.subplot(1, 2, 1)
plt.imshow(img, cmap="gray")
plt.axis("off")
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(enchanced_img, cmap="gray")
plt.axis("off")
plt.title("Enhanced Image")

plt.savefig("./assignment2_image1_global_histogram_equalization.png")
plt.show()
```



3. Local Histogram Equalization
Source Code:

```
def local_enhancement(img, E,  k0, k1, k2, neighborhood_size):

    # Calculate the mean intensity and standard deviation of the entire image.
    global_mean_intensity, global_standard_deviation = cv2.meanStdDev(img)

    # Calculate the local mean intensity and local standard deviation.
    local_mean_intensity = cv2.blur(img, (neighborhood_size, neighborhood_size))
    local_mean_squared_intensity = cv2.blur(np.square(img), (neighborhood_size, neighborhood_size))
    local_standard_deviation = np.sqrt(local_mean_squared_intensity - np.square(local_mean_intensity))

    # Select the candidate pixels.
    candidate_pixels = []
    for x in range(img.shape[0]):
        for y in range(img.shape[1]):

            mean_condition = local_mean_intensity[x, y] <= k0 * global_mean_intensity
            std_condition = k1 * global_standard_deviation <= local_standard_deviation[x, y] <= k2 *
global_standard_deviation

            if mean_condition and std_condition:
                candidate_pixels.append((x, y))

    # Enhance the candidate pixels.
    enhanced_img = img.copy()
    for x, y in candidate_pixels:
        # Enhance the candidate pixel.
        enhanced_img[x, y] = min(max(img[x, y] * E, 0), 255)

    return enhanced_img

neighbourhood_sizes = [3, 7, 11]

row, col = 1, len(neighbourhood_sizes)

plt.figure(figsize=(20, 16))

E = 6
k0 = 0.125
k1 = 0.01
k2 = 0.125

for i in range(len(neighbourhood_sizes)):
    neighbourhood_size = neighbourhood_sizes[i]
    enhanced_img = local_enhancement(
        img,
        E=E,
        k0=k0,
        k1=k1, k2=k2,
        neighborhood_size=neighbourhood_size
    )

    plt.subplot(row, col, i + 1)
    plt.imshow(enhanced_img, cmap="gray")
    plt.axis("off")
    plt.title(f"Neighborhood Size: {neighbourhood_size} x {neighbourhood_size}")
```

```
# Add text at bottom
plt.figtext(0.5, 0.275, f"Parameters: E:{E}\n k0:{k0}  k1:{k1}  k2:{k2}", ha="center", fontsize=16)
plt.savefig("./assignment2_image1_local_enhancement.png")
plt.show()
```



ค่า k0 k1 k2 ที่ใช้คือ

K0 = 0.175 K1 = 0.01 และ k2 = 0.175

ที่เลือกใช้ค่าดังกล่าว เนื่องจากต้องการให้ Candidate pixel ที่จะทำการ enhance นั้นเป็น pixel บริเวณสีดำ

Method ที่ควรใช้คือ

Local Histogram Equalization เนื่องจากภาพที่เราต้องการ Enhancement นั้นส่วนที่ต้องการ Enhancement เป็นเฉพาะส่วนที่มืดบริเวณเล็กๆ และไม่ต้องการที่จะสูญเสีย Contrast บริเวณสีขาวที่ชัดอยู่แล้วไป โดยเราสามารถกำหนดค่า Parameter เพื่อให้ Algorithm นั้นไป Enhancement เฉพาะส่วนที่เราต้องการได้ และยังรักษา Contrast ของภาพโดยรวมเอาไว้