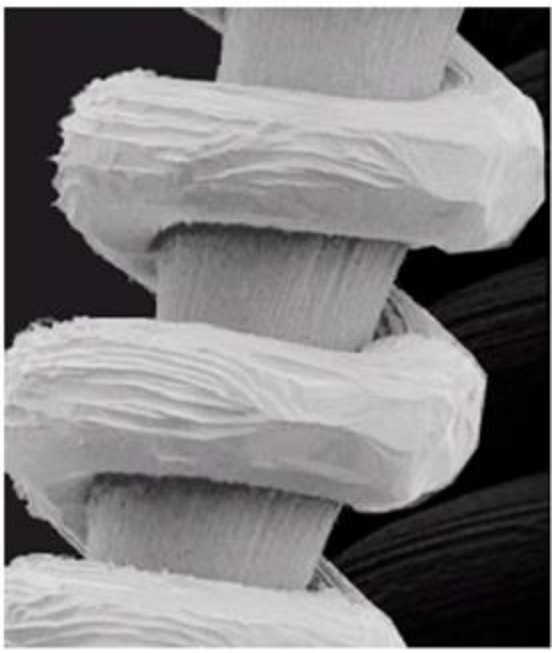
ข้อ 1

Gamma correction

```python
import os
import cv2
import numpy as np

def gamma_correction(image, gamma):
    enhanced_image = np.power(image / 255.0, gamma)
    enhanced_image = np.uint8(255 * enhanced_image)
    return enhanced_image

img = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
output_folder = "./gammaCorrection"

if not os.path.exists(output_folder):
    os.makedirs(output_folder)

gammaList = [x/10 for x in range(1, 10)] + [x for x in range(1, 11)]

for i in gammaList:
    tempImg = gamma_correction(img, i)
    output_path = os.path.join(output_folder, f"gamma={i}.jpg")
    cv2.imwrite(output_path, tempImg)
```

ภาพต้นฉบับ



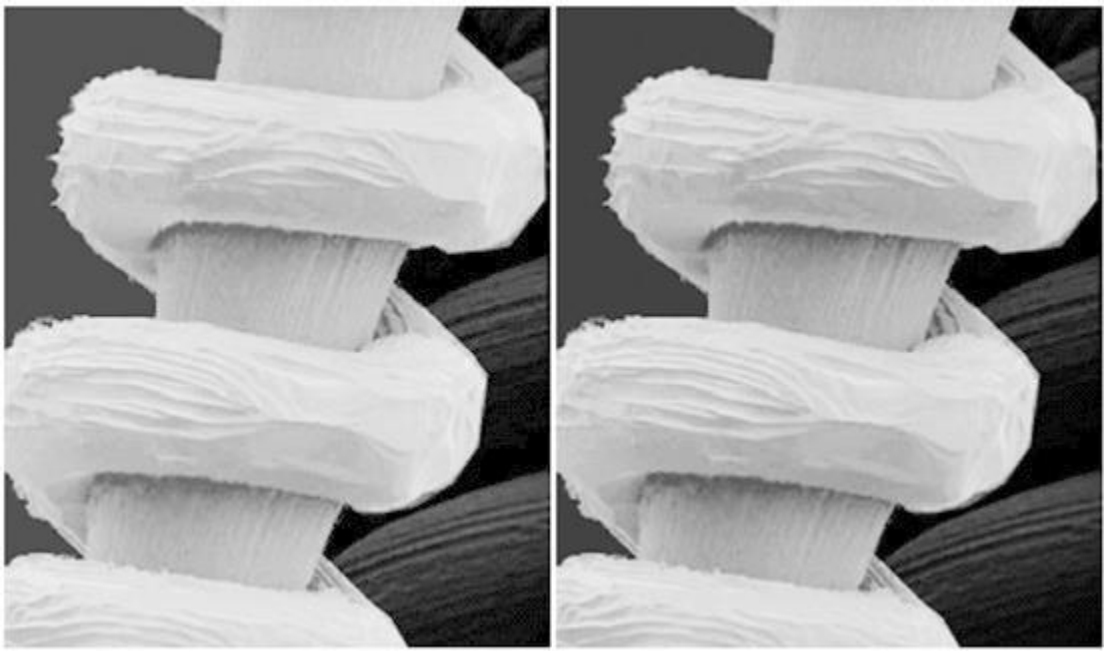
γ = 0.1

γ = 0.2

γ = 0.3

γ = 0.4

γ = 0.5 γ = 0.6



อธิบายเพิ่มเติม : ภาพที่ได้จากการทำ gamma correction จะได้ภาพที่ขาวหรือดำขึ้นกว่าภาพต้นฉบับ ซึ่งจะขึ้นอยู่กับค่า γ โดย ถ้าค่า γ อยู่ระหว่าง 0 ถึง 1 ภาพจะสว่างขึ้นแต่ถ้า γ มากกว่า 1 ภาพจะมืดลง เมื่อนำมาปรับใช้กับภาพนี้แล้วค่า γ ที่ใช้ก็ควรจะอยู่ในช่วง 0 ถึง 1 เพราะเราต้องการพิจารณาส่วนด้านขวาของภาพที่มีลักษณะมืดจึงควรใช้ gamma correction ในการปรับภาพให้สว่างขึ้น

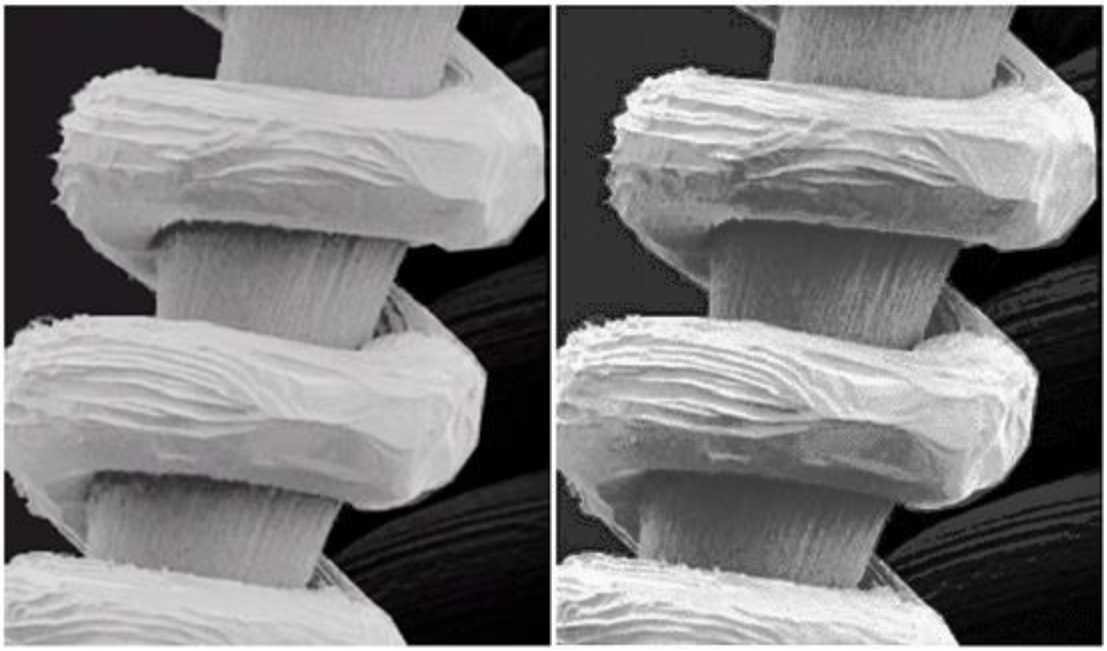
ข้อ 2

Global histogram equalization

```
import os
import cv2
import numpy as np

def global_histogram_equalization(img):
    hist, _ = np.histogram(img.flatten(), 256, [0, 256])
    cdf = hist.cumsum()
    cdf_normalized = ((cdf - cdf.min())* 255) / (cdf.max() - cdf.min())
    img_equalized = cdf_normalized[img]
    return img_equalized.astype(np.uint8)

img = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
imgGlobalEqualized = global_histogram_equalization(img)

output_folder = "./globalEqualization"
if not os.path.exists(output_folder):
```

```
    os.makedirs(output_folder)

cv2.imwrite(os.path.join(output_folder,f"global equalization.jpg"),
imgGlobalEqualized)
```

ภาพต้นฉบับ ภาพ global histogram equalization



อธิบายเพิ่มเติม : การทำ global histogram equalization เป็นการใช้เทคนิค cumulative distribution function ในการรวมความน่าจะเป็นในการเจอ pixel ตั้งแต่ตัวก่อนหน้าจนถึงตัวที่ต้องการ ซึ่งเทคนิคนี้จะทำให้ส่วนของภาพที่มองเห็นได้ยากมองเห็นไได้ชัดเจนมากขึ้น

ข้อ 3

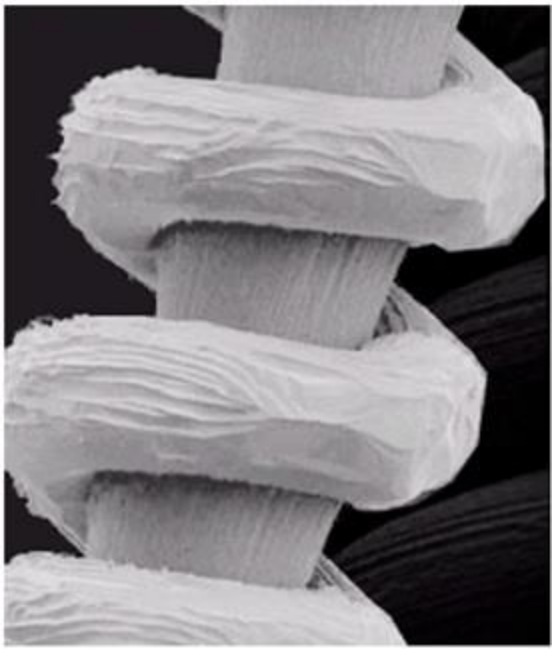
Local histogram equalization

```
import os
import cv2
import numpy as np

def local_histogram_equalization(img, neighborhood_size=3, k0=0.4, k1=0.02, k2=0.4):
    half_size = neighborhood_size// 2
    # For edge of picture
    padded_img = np.pad(img, ((half_size, half_size), (half_size, half_size)),
mode='reflect')
    img_equalized = np.copy(img)
```

```
    global_mean = np.mean(img)
    global_deviation = np.std(img)
    for i in range(half_size, img.shape[0] - half_size):
        for j in range(half_size, img.shape[1] - half_size):
            local_region = padded_img[i-half_size:i+half_size+1, j-
half_size:j+half_size+1]
            local_mean = np.mean(local_region)
            local_deviation = np.std(local_region)

            if local_mean < k0 * global_mean and k1 * global_deviation <=
local_deviation <= k2 * global_deviation:
                hist, _ = np.histogram(local_region.flatten(), 256, [0, 256])
                cdf = hist.cumsum()
                cdf_normalized = (cdf - cdf.min()) * 255 / (cdf.max() -
cdf.min())
                img_equalized[i, j] = cdf_normalized[img[i, j]]
    return img_equalized

image_path = 'image.jpg'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

setk0 = [i/10 for i in range(1,11)]
setk1 = [i/100 for i in range(1,11)]
setk2 = [i/10 for i in range(1,11)]

# # 3x3
for k0 in setk0:
    for k1 in setk1:
        for k2 in setk2:
            tempImg = local_histogram_equalization(image,3,k0,k1,k2)
            cv2.imwrite(os.path.join("localEqualization","3x3", f'3x3
k0={k0},k1={k1},k2={k2}.jpg'), tempImg)

# # 7x7
for k0 in setk0:
    for k1 in setk1:
        for k2 in setk2:
            tempImg = local_histogram_equalization(image,7,k0,k1,k2)
            cv2.imwrite(os.path.join("localEqualization","7x7", f'7x7
k0={k0},k1={k1},k2={k2}.jpg'), tempImg)
# # 11x11
for k0 in setk0:
    for k1 in setk1:
        for k2 in setk2:
```

```
            tempImg = local_histogram_equalization(image,11,k0,k1,k2)
            cv2.imwrite(os.path.join("localEqualization","11x11", f'11x11
k0={k0},k1={k1},k2={k2}.jpg'), tempImg)
```

ภาพต้นฉบับ



ภาพ local histogram equalization ที่คิดว่าดีที่สุดของ

3x3 (k_0=0.3 k_1=0.02 k_2=0.3)

7x7 (k_0=0.3 k_1=0.01 k_2=0.3)

11x11 (k_0=0.3 k_1=0.01 k_2=0.3)

อธิบายเพิ่มเติม : การทำ local histogram equalization จะแตกต่างกับ global ตรงที่จะพิจารณาเพียงแค่บริเวณๆนึงของ pixel นั้นๆ โดยขนาดก็ขึ้นอยู่กับค่าของ neighborhood ที่กำหนดเองได้ แล้วจะนำค่าบริเวณนั้นไปคำนวณหาค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานเพื่อนำไปเปรียบเทียบกับค่าของ global เพื่อดูว่าค่าของ pixel ไหนที่ควรได้รับการ equalized