ข้อ 1

Gamma correction

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)]

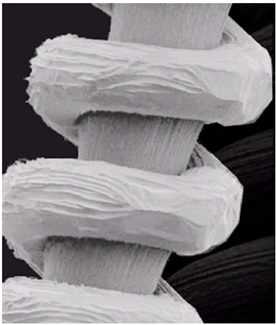
for i in gammaList:

    tempImg = gamma\_correction(img, i)

    output\_path = os.path.join(output\_folder, f"gamma={i}.jpg")

    cv2.imwrite(output\_path, tempImg)

ภาพต้นฉบับ



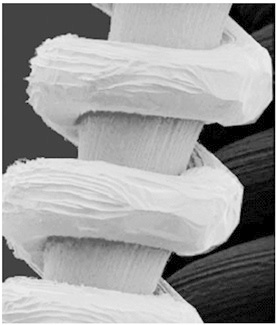
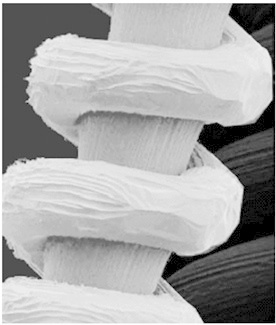
ɣ = 0.1 ɣ = 0.2



ɣ = 0.3 ɣ = 0.4



ɣ = 0.5 ɣ = 0.6



อธิบายเพิ่มเติม : ภาพที่ได้จากการทำ gamma correction จะได้ภาพที่ขาวหรือดำขึ้นกว่าภาพต้นฉบับ ซึ่งจะขึ้นอยู่กับค่า ɣ โดย ถ้าค่า ɣ อยู่ระหว่าง 0 ถึง 1 ภาพจะสว่างขึ้นแต่ถ้า ɣ มากกว่า 1 ภาพจะมืดลง เมื่อนำมาปรับใช้กับภาพนี้แล้วค่า ɣ 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)

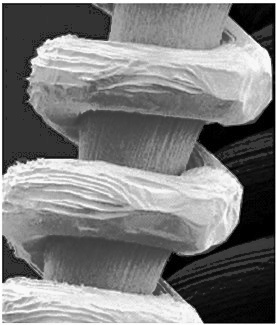
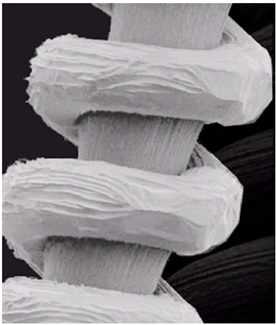
output\_folder = "./globalEqualization"

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:

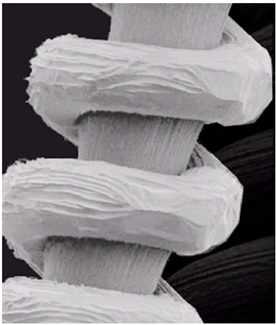
    for k1 in setk1:

        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 (k0=0.3 k1=0.02 k2=0.3)



7x7 (k0=0.3 k1=0.01 k2=0.3)



11x11 (k0=0.3 k1=0.01 k2=0.3)



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