พสิษฐ์ โวศรี 6434453423

## 1. Gamma Correction

```python
from PIL import Image
from PIL.Image import Image as IMG

def enhancePowerLaw(image:IMG, c, gamma):
    image = image.convert("L") # convert to grayscale
    max_level = 255 # max gray scale value of this image (255 for 8 bit)
    max_transformed_value = pow(max_level, gamma)
    for x in range(image.height): # height
        for y in range(image.width): # width
            r = image.getpixel((y,x))
            s = c*pow(r, gamma)
            s = s/max_transformed_value * max_level # convert value back to 0-255 range
            image.putpixel((y,x), round(s))
    image.save(f"ImageProcessing/assign2/q1_output.jpg")

img = Image.open("ImageProcessing/assign2/assignment2_image1.jpg")
enhancePowerLaw(img, 1, 0.5)
```

## 2. Global Histogram Equalization

```python
from PIL import Image
from PIL.Image import Image as IMG

def global_histogram_equalize(image:IMG):
    Hk = [0 for _ in range(256)]
    image = image.convert("L") # convert to grayscale
    # compute Hk
    for x in range(image.height):
        for y in range(image.width):
            r = image.getpixel((y,x))
            Hk[r] += 1
    # compute Pk from Hk
    pixel_cnt = image.height * image.width
    Pk = [h/pixel_cnt for h in Hk]
    Sk = [0 for _ in range(256)]
    # compute Sk
    Sk[0] = Pk[0]
    for i in range(1, 256): # exclude i = 0, no need to compute
        Sk[i] = Sk[i-1] + Pk[i]
    # convert Sk to 256 gray level
    Sk = [round(s*255) for s in Sk]
    # apply to image
    for x in range(image.height):
        for y in range(image.width):
            r = image.getpixel((y,x))
            image.putpixel((y,x), Sk[r])
    image.save(f"ImageProcessing/assign2/q2_output.jpg")


img = Image.open("ImageProcessing/assign2/assignment2_image1.jpg")
global_histogram_equalize(img)
```

## 3.Local Histogram Equalization

```
from PIL import Image
from PIL.Image import Image as IMG
import math

def local_histogram_equalize(image:IMG, neighbor_width, neighbor_height, constants:tuple):
    image = image.convert("L") # convert to grayscale
    new_image = Image.new("L", (image.width, image.height))
    # prevent image coordinate out of bound
    clamp = lambda n, upper_bound: max(min(upper_bound, n), 0)
    MAX_X = image.height - 1
    MAX_Y = image.width - 1

    # global statistic
    Hk = [0 for _ in range(256)]
    for x in range(image.height):
        for y in range(image.width):
            r = image.getpixel((y,x))
            Hk[r] += 1
    Pk = [h/(image.height*image.width) for h in Hk]
    global_mean = sum([k*Pk[k] for k in range(256)])
    global_sd = math.sqrt(sum([math.pow((k-global_mean), 2) * Pk[k] for k in range(256)]))

    def histogram_equalize(coordinate:tuple, neighbor_width, neighbor_height, constants:tuple, global_mean,
global_sd):
        x, y = coordinate
        k0, k1, k2 = constants
        x_upper = clamp(round(x + (neighbor_height-1)/2), MAX_X)
        x_lower = clamp(round(x - (neighbor_height-1)/2), MAX_X)
        y_upper = clamp(round(y + (neighbor_width-1)/2), MAX_Y)
        y_lower = clamp(round(y - (neighbor_width-1)/2), MAX_Y)
        # compute Hk
        Hk = [0 for _ in range(256)]
        pixel_cnt = 0
        for i in range(x_lower, x_upper+1):
            for j in range(y_lower, y_upper+1):
                r = image.getpixel((j,i))
                Hk[r] += 1
                pixel_cnt += 1
    # compute Pk from Hk
    Pk = [h/pixel_cnt for h in Hk]

    # local statistic
    local_mean = sum([k*Pk[k] for k in range(256)])
    local_variance = math.sqrt(sum([math.pow((k-local_mean), 2) * Pk[k] for k in range(256)]))
    # Conditions
    old_value = image.getpixel((y, x))
    if (local_mean < k0*global_mean) and (k1*global_sd <= local_variance) and
       (local_variance <= k2*global_sd):
        # compute Sk
        Sk = [0 for _ in range(256)]
        Sk[0] = Pk[0]
        for i in range(1, old_value + 1): # exclude i = 0, no need to compute
            Sk[i] = Sk[i-1] + Pk[i]
```

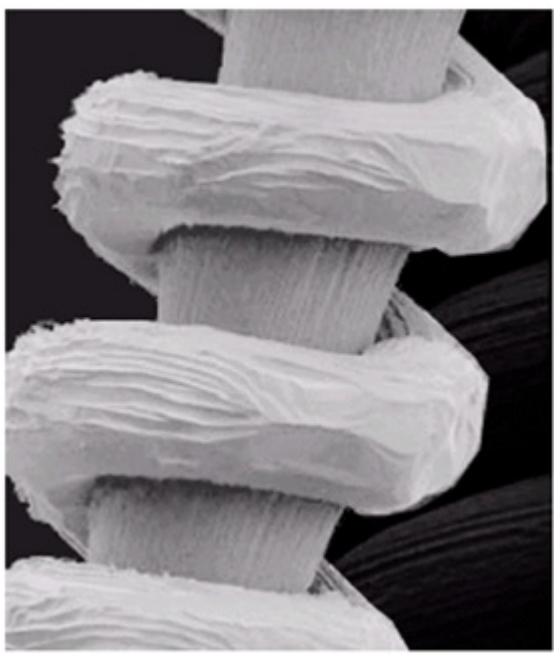
```
            # convert new_value to 256 gray level
            new_value = round(Sk[old_value] * 255)
            # apply to image
            new_image.putpixel((y,x), new_value)
        else:
            new_image.putpixel((y,x), old_value)

    # driver
    for x in range(image.height):
        for y in range(image.width):
            histogram_equalize((x, y), neighbor_width, neighbor_height, constants, global_mean,
                               global_sd)

    new_image.save(f"ImageProcessing/assign2/q3_output_{neighbor_width}x{neighbor_height}.jpg")
# Main
img = Image.open("ImageProcessing/assign2/assignment2_image1.jpg")
local_histogram_equalize(img, 11, 11, (0.4, 0.03, 0.2))
```

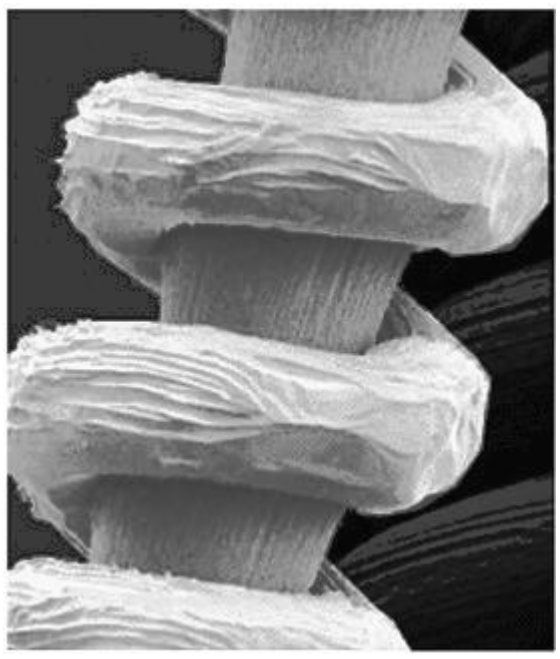
## Output

**Original image**



**gamma correction**


**global histogram equalization**



**Local histogram equalization**

**Neighborhood 3x3**

**k0=0.4,k1=0.02,k2=0.3**

**Neighborhood 7x7**

**k0=0.4,k1=0.03,k2=0.2**

**Neighborhood 11x11**

**k0=0.4,k1=0.03,k2=0.2**

**Best Method to Use?**

**Ans** วิธีที่ทำให้ส่วนมืดด้านขวาเห็นรายละเอียดได้ชัดขึ้นมากที่สุดน่าจะเป็นวิธี Gamma Correction ข้อเสียคือส่วนอื่นของภาพจะถูกปรับไปด้วย วิธีที่ดีรองลงมาคือ Global Histogram Equalization ที่ทำให้ทั้งภาพมีมิติมากขึ้น แต่ด้านมืดทางด้านขวาเห็นรายละเอียดไม่ดีเท่าวิธีที่กล่าวไปข้างต้น ส่วนวิธี Local Histogram Equalization ดึงส่วนมืดออกมาเด่นมาก แต่ด้วยความที่ neighborhood เล็กทำให้ภาพแตก และหลีกเลี่ยงการปรับขอบของ object ไม่ได้ ดังนั้นจะเห็นขอบขาวที่ไม่ต้องการรอบ object