DIP homework 3

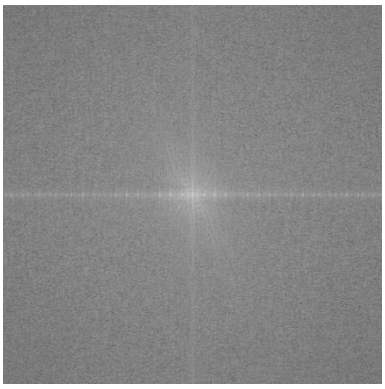
4109061012 陳柏翔

Problem 8：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np

IMAGE_PATH = "homework_3/src/Fig0441(a)(characters_test_pattern).tif"
TARGET_FOLDER = "homework_3/result"

if __name__ == "__main__":
    img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)

    # Problem (a): Compute (centered) Fourier spectrum
    ft_img = np.fft.fft2(img)
    ave = abs(ft_img[0, 0]) / (img.shape[0] * img.shape[1])
    ft_img = np.fft.fftshift(ft_img)

    # Problem (c): Average value
    print(f">> Average value of image = {ave}")
    print(f">> Average value of image in fourier spectrum = {np.average(ft_img)}")

    # Problem (b): Display (a)
    ft_img = np.log(1 + abs(ft_img))
    ft_img = ft_img / ft_img.max() * 255
    cv2.imwrite(f"{TARGET_FOLDER}/P8_centered_FT_image.jpg", ft_img)
```

Terminal outputs:

```
>> Average value of image = 207.31469924621416
>> Average value of image in fourier spectrum = (228.99999999999994+1.441836854825255e-14j)
```

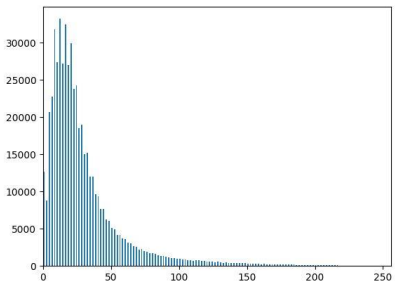


(Centered image in Fourier Spectrum)

Problem 9：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt

IMAGE_PATH = "homework_3/src/Fig0235(c)(kidney_original).tif"
TARGET_FOLDER = "homework_3/result"

T = 115

def filtering(img: np.ndarray, kernel: np.ndarray, coeff = 9) -> np.ndarray:
    """ Filtering an image with symmetric padding """
    s, pad = kernel.shape[0], kernel.shape[0] // 2
    new_img = np.empty_like(img, dtype=np.float32)
    img = np.pad(img, ((pad, pad), (pad, pad)), "symmetric")

    for r in tqdm(range(new_img.shape[0])):
        for c in range(new_img.shape[1]):
            new_img[r, c] = round(coeff * np.average(kernel * img[r : r+s, c : c+s]))
    return new_img

def Edge_detection(img: np.ndarray, threshold: int) -> np.ndarray:
    """ Problem (a): Edge detection using sobel filters """
    mask_x = np.array([
        [-1,  0,  1],
        [-2,  0,  2],
        [-1,  0,  1]
    ])
    mask_y = np.array([
        [-1, -2, -1],
        [ 0,  0,  0],
        [ 1,  2,  1]
    ])
    Gx, Gy = filtering(img, mask_x), filtering(img, mask_y)
    grad_sum = abs(Gx) + abs(Gy)

    # Histogram
    plt.hist(grad_sum.ravel(), 256, (0, 256)), plt.xlim(left=0, right=256)
    plt.savefig(f"{TARGET_FOLDER}/P9_Grad_Hist.jpg"), plt.clf()

    # Binarization
    grad_sum[grad_sum < threshold] = 0
    grad_sum[grad_sum > 0] = 255
    return grad_sum.astype(np.uint8)

if __name__ == "__main__":
    # Problem (b): smoothing with 3x3 mask & use program from (a)
    img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)

    print(f">> Smoothing..")
    img = filtering(img, np.ones((3, 3)), 1)

    print(f">> Computing Gradients (Gx, Gy)..")
    edge = Edge_detection(img, T)

    cv2.imwrite(f"{TARGET_FOLDER}/P9_Result_T{T}.jpg", edge)
    print(f">> Threshold T: {T}")
```

(Original image)

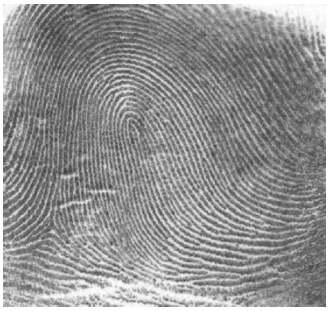


(Gradient image histogram)

(Edge T=115)

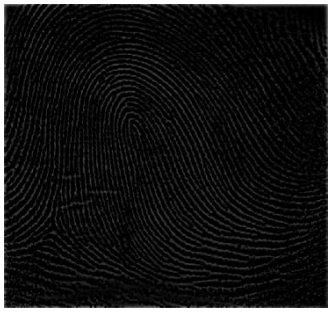
Problem 10：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np

IMAGE_PATH = "homework_3/src/Fig0457(a)(thumb_print).tif"
TARGET_FOLDER = "homework_3/result"

D0 = 50
T = 10

def Gauss_Highpass_Filter(shape: tuple, cutoff: int, center: tuple) -> np.ndarray:
    """ Problem (a): Gaussian highpass filter """
    x = np.linspace(0, shape[1]-1, shape[1])
    y = np.linspace(0, shape[0]-1, shape[0])
    xv, yv = np.meshgrid(x, y)

    xv = xv - center[1]
    yv = yv - center[0]
    D2_uv = np.square(xv) + np.square(yv)

    H_uv = np.ones_like(D2_uv) - np.exp(-D2_uv / (2 * cutoff**2))
    return H_uv

if __name__ == "__main__":
    # Problem (b)
    img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)
    m, n = img.shape[0], img.shape[1]
    p, q = 2*m - 1, 2*n - 1

    ft_img = np.zeros((p, q))
    ft_img[:m, :n] = img
    hp = Gauss_Highpass_Filter((p, q), D0, (m, n))
    cv2.imwrite(f"{TARGET_FOLDER}/P10_Highpass_Filter.jpg", hp * 255)

    ft_img = np.fft.fft2(ft_img)
    ft_img = np.fft.fftshift(ft_img)
    ft_img = ft_img * hp
    ft_img = np.fft.ifftshift(ft_img)
    img = np.fft.ifft2(ft_img)

    img = img.real[:m, :n]
    cv2.imwrite(f"{TARGET_FOLDER}/P10_Filtered_image.jpg", img)

    img[img < T] = 0
    img[img > 0] = 255
    cv2.imwrite(f"{TARGET_FOLDER}/P10_Result.jpg", img)
```



(Source)

(Gaussian high-pass filter)



(Filtered image)

(Result image)