Q8-Q10 DIP HW3 4109029034 蘇柏叡

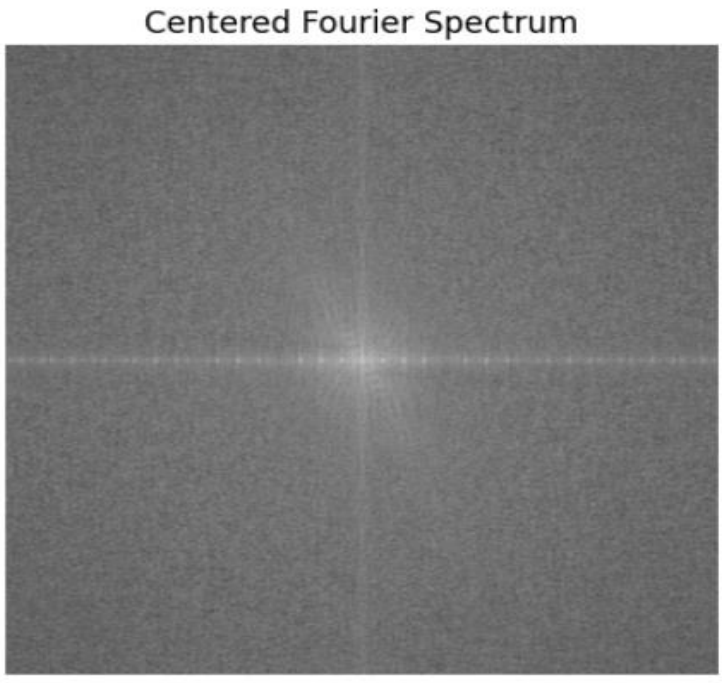
本次題目是使用 colab 環境運行，因小題間環環相扣，故程式碼並沒有分 a、b 小題去撰寫。詳細細節操作寫於 ReadMe_HW3

link:https://colab.research.google.com/drive/1oD5UqthEXv4oiq8k4Pt7pbRu1Unawci7?usp=sharing

Q8(a)、(b)、(c)

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

# 讀取圖像
image = cv2.imread('/content/Fig0441(a)(characters_test_pattern).tif', 0)

# 計算中心化傅立葉變換
fft_centered = np.fft.fftshift(np.fft.fft2(image))#這邊是2D所以fft2

# 計算頻譜的幅度譜
spectrum = np.abs(fft_centered)

# 顯示頻譜
plt.imshow(np.log(1 + spectrum), cmap='gray')
plt.title('Centered Fourier Spectrum')
plt.axis('off')
plt.show()

# 計算圖像的平均值
average_value = np.mean(image)
print("圖像的平均值:", average_value)
```

Result:



圖像的平均值: 207.31469924621416

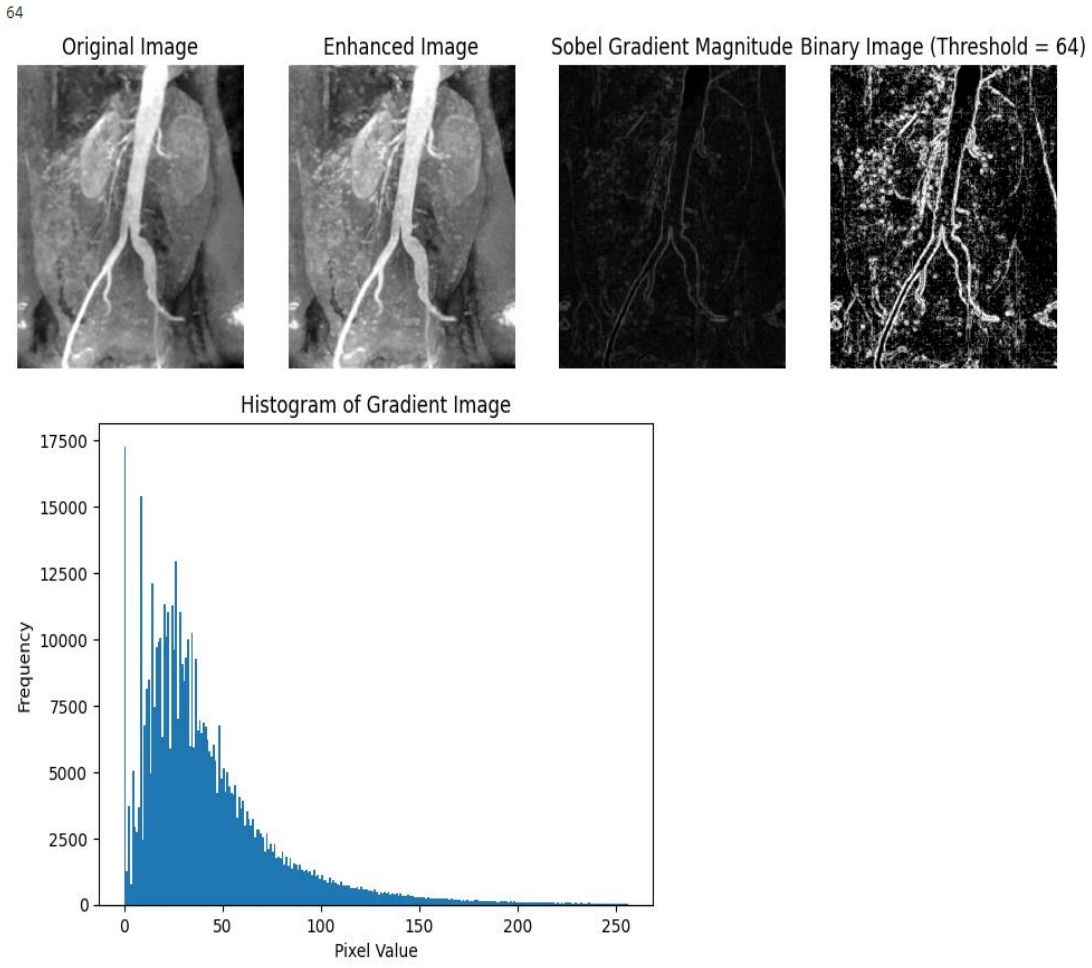
Q9(a)、(b)

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

def spatial_filtering(image, mask):
    filtered_image = cv2.filter2D(image, -1, mask)
    return filtered_image

def sobel_gradient(image):
    sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
    gradient_magnitude = np.hypot(sobel_x, sobel_y)
    return gradient_magnitude

def apply_threshold(image, threshold):
    binary_image = np.where(image >= threshold, 255, 0).astype(np.uint8)
    return binary_image

# Laplacian enhancement function
def laplacian_enhancement(image, laplacian_mask):
    enhanced_image = spatial_filtering(image, laplacian_mask)
    final_image = cv2.add(image, enhanced_image)
    return final_image

# Load the image
image_path = "/content/Fig0235(c)(kidney_original).tif"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
# Define the Laplacian mask
laplacian_mask = np.array([[-1, -1, -1],
                           [-1, 8, -1],
                           [-1, -1, -1]], dtype=np.float32)

# Apply Laplacian enhancement
enhanced_image = laplacian_enhancement(image, laplacian_mask)

# Compute Sobel gradient magnitude
sobel_gradient_magnitude = sobel_gradient(enhanced_image)

# Apply thresholding to isolate the large blood vessel
# You may need to adjust the threshold value based on the image and requirements
threshold = int(input())
binary_image = apply_threshold(sobel_gradient_magnitude, threshold)

# Display the original image, enhanced image, Sobel gradient magnitude, and binary image
plt.figure(figsize=(12, 4))

plt.subplot(1, 4, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 4, 2)
plt.imshow(enhanced_image, cmap='gray')
plt.title('Enhanced Image')
plt.axis('off')
```

```
plt.subplot(1, 4, 3)
plt.imshow(sobel_gradient_magnitude, cmap='gray')
plt.title('Sobel Gradient Magnitude')
plt.axis('off')

plt.subplot(1, 4, 4)
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Image (Threshold = {})'.format(threshold))
plt.axis('off')

plt.show()
# Display histogram of gradient image
plt.hist(sobel_gradient_magnitude.ravel(), bins=256, range=(0, 256))
plt.title('Histogram of Gradient Image')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()

```

Result:

雖然高峰位於 25 上下，但經實測後雜訊極多，故實測將 threshold 拉至 50 以上會逐漸減少白色斑點雜訊，最終我們將閾值門檻設為 64，最後再進行二值化輸出得到圖片，並且將原圖、Enhanced、Sobel、二值化的圖片呈現。



Q10 (a)、(b)

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

def gaussian_highpass_filter(size, center, sigma):
    rows, cols = size
    crow, ccol = center

    # Generate a grid of points
    x = np.arange(-cols/2, cols/2)
    y = np.arange(-rows/2, rows/2)
    X, Y = np.meshgrid(x, y)

    # Create Gaussian function
    gaussian = np.exp(-((X - crow) ** 2 + (Y - ccol) ** 2) / (2 * sigma ** 2))

    # Apply highpass by subtracting from 1
    return 1 - gaussian

# Load the image and convert it to grayscale
image = cv2.imread('/content/Fig0457(a)(thumb_print).tif', cv2.IMREAD_GRAYSCALE)

# Apply Fourier transform
f = np.fft.fft2(image)
fshift = np.fft.fftshift(f)

# Specify parameters for Gaussian highpass filter
size = image.shape
center = (size[0] // 2, size[1] // 2)  # Center of the image
sigma = 160  # Standard deviation of Gaussian
```

```python
# Generate Gaussian highpass filter
highpass_filter = gaussian_highpass_filter(size, center, sigma)

# Apply filter
filtered_fshift = fshift * highpass_filter

# Apply inverse Fourier transform
filtered_image = np.fft.ifftshift(filtered_fshift)
filtered_image = np.fft.ifft2(filtered_image)
filtered_image = np.abs(filtered_image)

# Thresholding
threshold_value = 128
binary_image = np.where(filtered_image > threshold_value, 255, 0)

# Display filtered image
plt.imshow(binary_image, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')
plt.show()
```

Result: