HW3 REPORT

110511010 楊育陞

1 Method

執行指令: python3 hw3.py -i <input image path>

1.1 Laplacian Spatial Filtering

```
# kernel
identity = np.array([[0, 0, 0], [0, 1, 0], [0, 0, 0]])
if kernel_type == 0:
    kernel = identity - np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
elif kernel_type == 1:
    kernel = identity - np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]])

# convolution, padding with edge
pad = kernel.shape[0] // 2
padded_image = np.pad(image, ((pad, pad), (pad, pad)), 'edge')
filtered_image = np.zeros(image.shape, dtype=np.int32)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        filtered_image[i, j] = np.sum(padded_image[i:i+kernel.shape[0], j:j+kernel.shape[1]] * kernel)
filtered_image = np.clip(filtered_image, 0, 255).astype(np.uint8)
```

1. 定義 kernel, 使用 identity kernel 減去 Laplacian kernel 作為 filter
2. 對圖片進行 padding, 以 edge(replicate padding) 填補
3. 以 filter 對圖片進行 convolution, 並將結果 clip 至 0–255

1.2 Laplacian Frequency Filtering

```
# read input image
image = cv.imread(input_image, cv.IMREAD_GRAYSCALE)

# normalize
image_norm = image / 255.0

# fourier transform
f = np.fft.fft2(image_norm)
fshift = np.fft.fftshift(f)

# Laplacian filter
P, Q = fshift.shape
H = np.zeros(fshift.shape)
for u in range(P):
    for v in range(Q):
        H[u, v] = -4 * np.pi**2 * ((u - P//2)**2 + (v - Q//2)**2)

# apply filter
filtered_fshift = fshift * H
```

```
filtered_f = np.fft.ifftshift(filtered_fshift)
filtered_image = np.real(np.fft.ifft2(filtered_f))

# scale down filtered image
scaled_filtered_image = filtered_image / np.max(np.abs(filtered_image))

# result image
result_image = image_norm - scaled_filtered_image
result_image = np.clip(result_image, 0, 1)
result_image = (result_image * 255).astype(np.uint8)
```

我依照課本先將圖片 normalize 至 0–1, 並將 filter scale 至約-1–1

1. 將像素值 normalize 至 0–1, 並進行 Fourier Transform 與 shift
2. 定義 Laplacian filter, 並對 Fourier Transform 後的圖片進行 filter
3. 將 filter 後的 Fourier Transform 進行 inverse shift 與 inverse Fourier Transform
4. 將 filter 後的圖片 scale 至 0–1, 並將原圖片減去 filter 後的圖片
5. 將結果 clip 至 0–1, 並將像素值 scale 至 0–255

2 Result

2.1 Laplacian Spatial Filtering

Figure 1: Source Image (left), Spatial Filtered by Kernel 0 (middle), and Spatial Filtered by Kernel 1 (right)

$$kernel0: \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad kernel1: \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

2.2 Laplacian Frequency Filtering

Figure 2: Source Image (left) and Frequency Filtered (right)

2.3 Comparison

可以明顯看出 Frequency Filter 表現大於 Spatial Filter，細節強化較明顯，而 Spatial Filter 中 kernel 1 又優於 kernel 0。

Figure 3: Spatial Filtered by Kernel 0 (left), and Spatial Filtered by Kernel 1 (middle), Frequency Filtered (right)

3 Feedback

我認為這次作業難度適中，但是若有提供參考圖片與結果，可能會更容易對照自己的結果。