Intro. Image Processing
LAB 3

謝侑哲

112550069

1. Method

A. Sharpening with Spatial Laplacian

(1) Preliminary

Laplacian 是求點對於周圍各點的梯度，也就是各點對於周圍的變化。在邊界等變化大的區域會獲得更大的值。把 Laplacian 值加回去，可以強化變化大的邊界區域等，達到 sharpen 的效果。

(2) Method

Spatial Laplacian 對於空間中的各方向求梯度，在離散的 case 以 x 軸來說就是 f(u-x, v) + f(u+x, v) – 2 * f(u, v)的形式。推廣到 2 維空間，再加上就是 pixel 自己，就可以變成以下 kernel:

```
kernel_1 = np.array([[0, -1, 0],
                     [-1, 5, -1],
                     [0, -1, 0]])
kernel_2 = np.array([[-1, -1, -1],
                     [-1, 9, -1],
                     [-1, -1, -1]])
```

因為希望 kernel 的 pattern 直接應用到 image 上，所以要使用 convolution 的方式來做，convolution 在應用上，就是把 filter 上下、左右翻轉，再套用到圖片上做 weighted-sum，如下:

```
def conv(image, kernel): # assume kernel is odd size
    n, m = kernel.shape
    kernel = np.flip(kernel, axis=(0, 1)) # flip up-down and left-right => convolution (shape m,n)
    image = np.pad(image, ((n//2, n//2), (m//2, m//2)), mode='constant', constant_values=0) # zero-p
    for i in range(image.shape[0]-m+1):
        for j in range(image.shape[1]-n+1):
            image[i][j] = np.clip(np.sum(image[i:i+m, j:j+n] * kernel), 0, 255)
    return image[:-n+1, :-m+1] # remove padding
```

有 convolution 跟設計好的 kernel 之後，就可以直接使用到圖片上

B. Sharpening with Frequency Laplacian

(1) Method

Frequency Laplacian 的概念跟 Spatial Laplacian 一樣，只是把空間中的微分，透過 Fourier Transform 轉到 Frequency domain 去做。

$$
\begin{aligned}
g(x,y) &= \Im^{-1}\{F(u,v) - H(u,v)F(u,v)\} \\
&= \Im^{-1}\{[1 - H(u,v)]F(u,v)\} \\
&= \Im^{-1}\{[1 + 4\pi^2 D^2(u,v)]F(u,v)\}
\end{aligned}
$$

在實作的 code 如下:

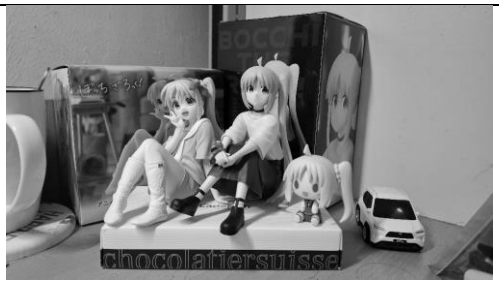
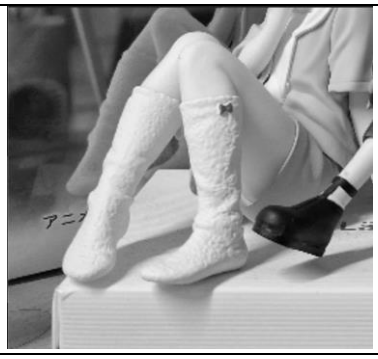
```
def Laplacian_frequency(image):

    f_transform = np.fft.fft2(image)
    # f_transform_shifted = f_transform
    f_transform_shifted = np.fft.fftshift(f_transform) # shift zero frequency to center

    # construct distance map
    u = np.arange(0, image.shape[0])
    v = np.arange(0, image.shape[1])
    u, v = np.meshgrid(u, v, indexing='ij')
    u0 = image.shape[0] // 2
    v0 = image.shape[1] // 2
    dis = np.sqrt((u - u0)**2 + (v - v0)**2)

    H_ = (4 * (np.pi**2) * (dis**2)) # - H
    H_normalized = (H_ - H_.min()) / (H_.max() - H_.min()) # normalize to weight [0,1]
    factor = 10
    Lap_transform = f_transform_shifted * (1 + factor * H_normalized) # high frequency w

    image_ret = np.fft.ifft2(np.fft.ifftshift(Lap_transform)).real
    image_ret = np.clip(image_ret, 0, 255).astype(np.uint8)

    save_image(image_ret, 'Lap_frequency_1')
```

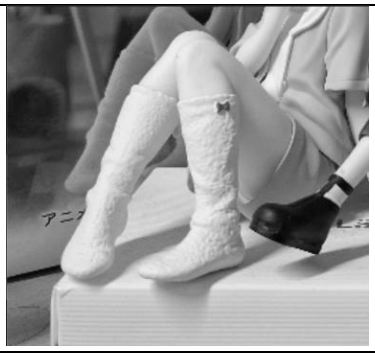
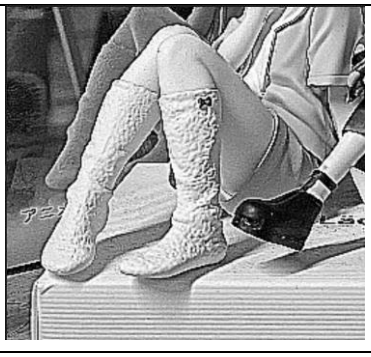
步驟為:

1. 把圖片用 FFT 轉成 frequency domain 並利用 fftshift 把 frequency=0 移到中間
2. 計算在 frequency domain 的 distance map
3. 利用 distance map 去計算出 H(u,v)，因為希望 H(u,v)是作為權重的存在，所以要 normalize 到[0,1]
4. 把得到的權重乘以一個 factor 來做為放大效果，乘上原本圖片的 frequency image
5. 把 frequency image 用 inverse FFT 轉回 spatial domain 並限縮至合理的 RGB 範圍

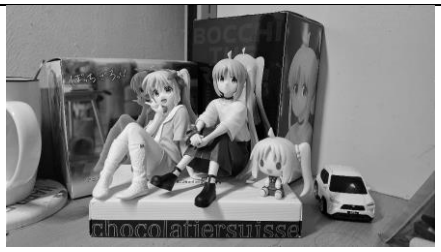
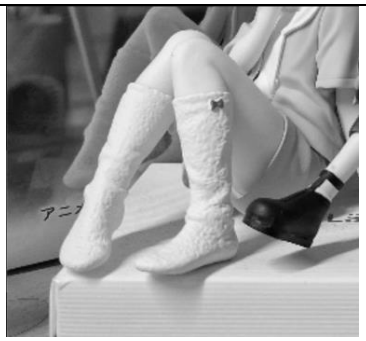
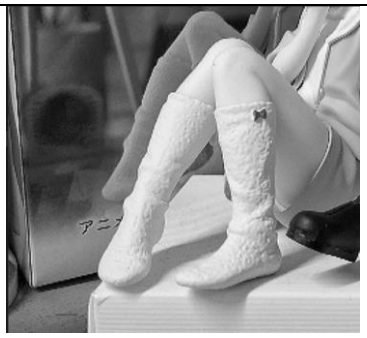
2. Result

A. Sharpening with Spatial Laplacian

Origin	Kernel 1
	
	

Origin	Kernel 2
	
	

B. Sharpening with Frequency Laplacian

Origin	Frequency Laplacian
	
	

C. Time Consume

```
Laplacian_frequency Time: 0.096448 seconds
```

```
Laplacian_spatial Time: 19.189555 seconds
```

從 Spatial Laplacian 的圖中可以觀察到，在 sharpen 之後，圖片變得有點 noisy，這是因為 spatial filter 只能看到 local 資訊，導致對於變化的計算結果會更加浮動，使得圖片更 noisy。而 Frequency Laplacian 的做法則因為分析了整張圖的 Frequency 並強化高頻的部分，擁有 global 資訊，因此得到的圖片會比較不這麼 noisy。

從時間上分析，可以發現在 1020*574 的圖片上，Spatial Laplacian 所消耗的時間大概是 Frequency Laplacian 的 200 倍，運算的時間久非常多。

3. Feedback

本次課程中，我了解到怎麼透過 filter 來對圖片進行想要的修改，更重要的是，如何實作透過 frequency 去更動圖片的作法，對於我影像處理的技能開拓了新且重要的一個部分，也十分有趣。希望未來做專題時，可以很好的利用 FFT 之類，能夠做到頻率域操作的方式來對於 project 做改進