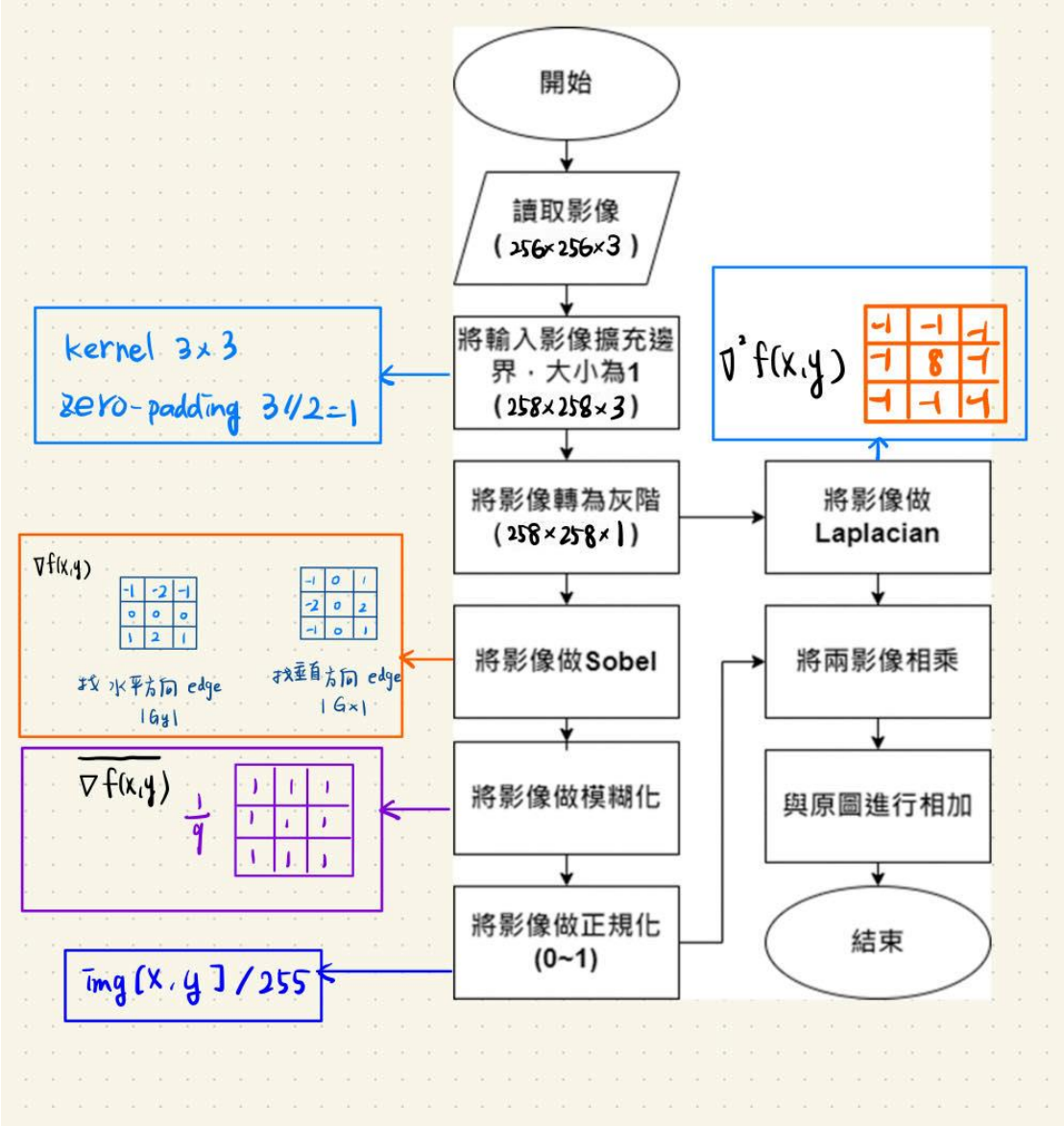
影像處理 HW02

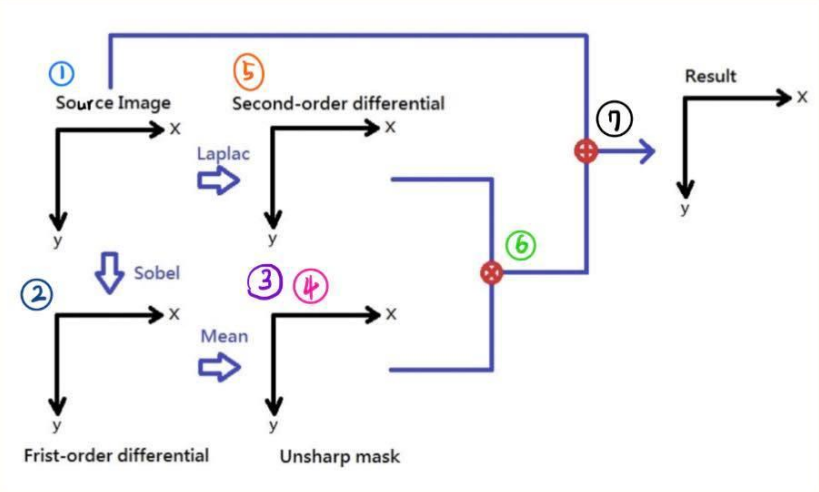
影像銳化 Sharpening Unsharp Masking

資工碩一 7111056426 蘇亭云

流程：



HW 02 影像銳化 用 Unshape mask



① Source Image 彩色轉灰階

② 對 ① 做 Sobel filter 得一階微分 Edge Image

$\nabla f(x,y)$

-1	-2	-1
0	0	0
1	2	1

找水平方向 edge

$|G_y|$

-1	0	1
-2	0	2
-1	0	1

找垂直方向 edge

$|G_x|$

③ 對 ② 一階微分 Edge Image 做 Mean filter 得模糊影像

$\overline{\nabla f(x,y)}$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

---

④ 對 ③ 模糊影像做正規化 [0, 1]

Img [x, y] / 255 → 加速運算

⑤ 對 ① 做 Laplacian 得二階微分 Edge Image

$\nabla^2 f(x,y)$

-1	-1	-1
-1	8	-1
-1	-1	-1

⑥ 將 ④ 正規化後的模糊一階 Edge Image

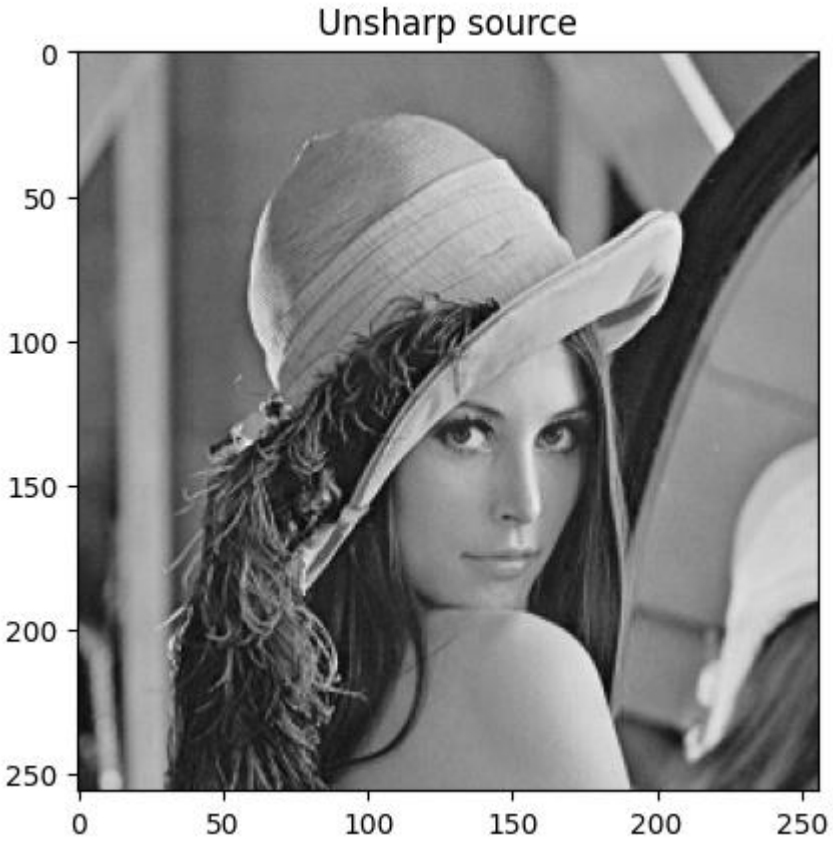
⊗ 乘上 ⑤ 二階微分 Edge Image

⑦ 原灰階圖 ① ⊕ 加上 ⑥ 得結果圖

(銳利化且無雜訊)

STEP 1、讀取彩色圖片轉為灰階圖片並做 zero-padding

```
# STEP 1、讀取彩色圖片轉為灰階圖片並做zero-padding
img_path="source\Lena_256x256.png"
# 設 mask 為3*3
kernel_size=3
# 來源圖讀取成灰階影像
img_gray = cv2.imread(img_path ,cv2.IMREAD_GRAYSCALE)
# zero-padding
pad=kernel_size//2
img_padded = np.pad(img_gray, pad)
```



STEP 2、對 STEP1 灰階圖片做 sobel filter 得一階微分 Eage 影像

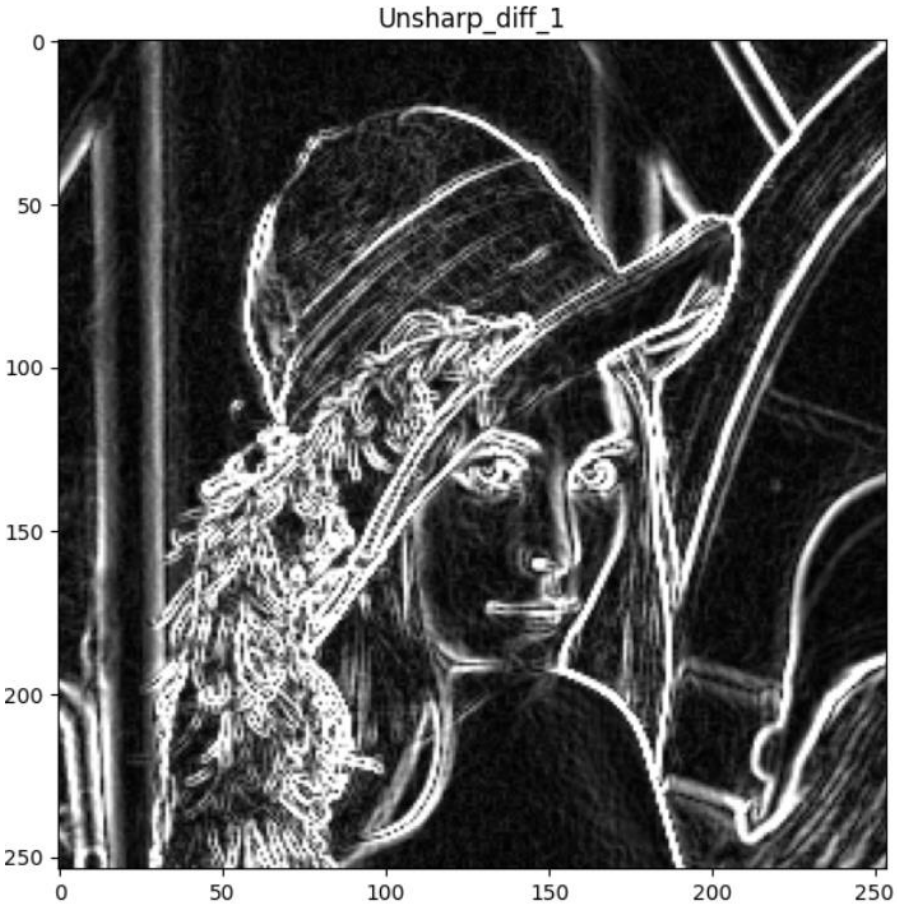
一階微分 套 sobel 公式

f=|(z7+2z8+z9)-(z1+2z2+z3)| + |(z3+2z6+z9)-(z1+2z4+z7)|

```
# mask
# [(i-1,j-1),(i,j-1),(i+1,j-1),
#  (i-1,j),  (i,j),  (i+1,j),
#  (i-1,j+1),(i,j+1),(i+1,j+1)
# ]
def sobel(img,pad):
    h,w=img.shape
    out = np.zeros(img.shape)
    # pad=1 ~ pad-w=257
    for i in  range(pad,h-pad):
        for j in range(pad,w-pad):
            out[i,j]= np.abs((img[i-1,j+1] + 2*img[i,j+1] + img[i+1,j+1])-(img[i-1,j-1] + 2*img[i,j-1] + img[i+1,j-1]))\
                + np.abs((img[i+1,j-1] + 2*img[i+1,j] + img[i+1,j+1])-(img[i-1,j-1] + 2*img[i-1,j] + img[i-1,j+1]))
            # boundary check 超過邊界拉回在邊界上
            out[i][j] = int(np.clip(out[i][j], 0, 255))
    return out
```

✓ 0.6s

```
# STEP 2、對STEP1 灰階圖片做sobel filter 得一階微分Eage影像
img_diff1=sobel(img_padded,pad)
```



Convolution function

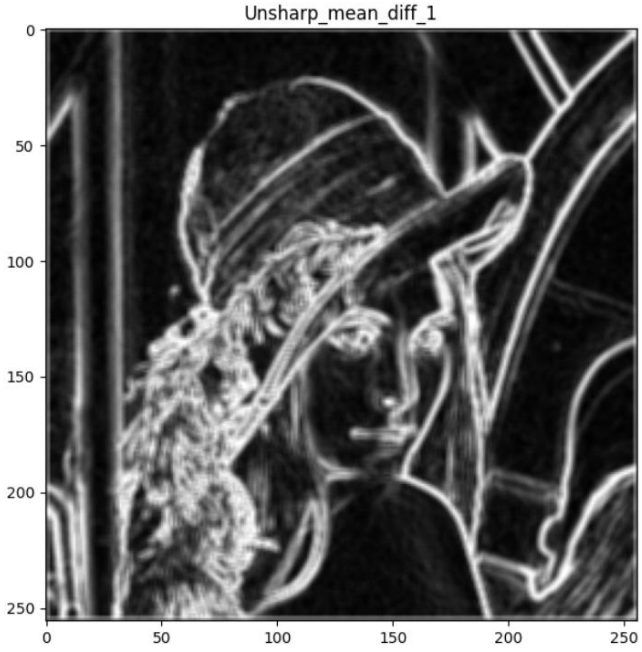
卷積 並超過0~255 需拉回

```
# mask
# [(i-1,j-1),(i,j-1),(i+1,j-1),
#  (i-1,j),  (i,j),  (i+1,j),
#  (i-1,j+1),(i,j+1),(i+1,j+1)
# ]
def convolution(img,mask,pad):
    h,w=img.shape
    out = np.zeros(img.shape)

    # pad=1 ~ pad-w=257
    for i in  range(pad,h-pad):
        for j in range(pad,w-pad):
            out[i,j]=mask[0]*img[i-1,j-1]\
                        + mask[1]*img[i,j-1]\
                        + mask[2]*img[i+1,j-1]\
                        + mask[3]*img[i-1,j]\
                        + mask[4]*img[i,j]\
                        + mask[5]*img[i+1,j]\
                        + mask[6]*img[i-1,j+1]\
                        + mask[7]*img[i,j+1]\
                        + mask[8]*img[i+1,j+1]
            # boundary check 超過邊界拉回在邊界上
            out[i][j] = int(np.clip(out[i][j], 0, 255))
    return out
```

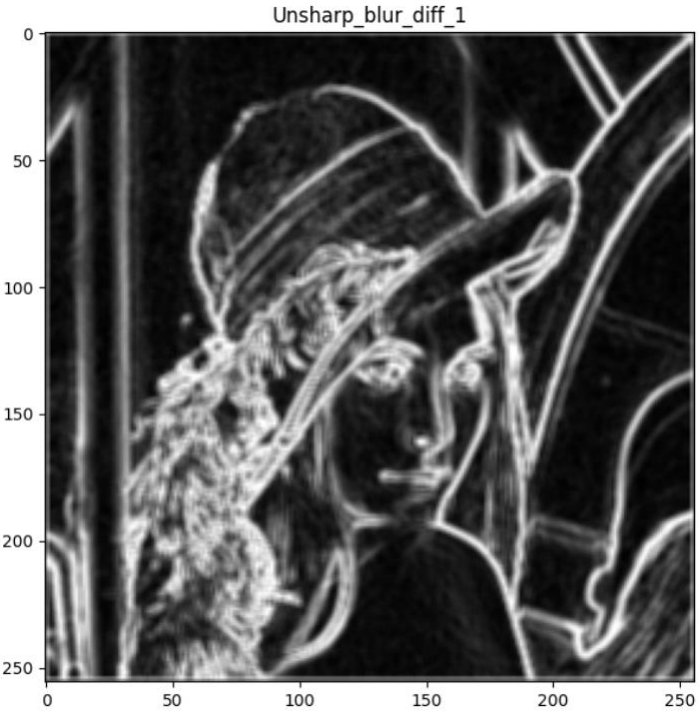
STEP 3、對 STEP2 一階微分 Edge 影像 做 Mean filter 得模糊影像 (去雜訊)

```
mean_mask = [1/9, 1/9, 1/9,
             1/9, 1/9, 1/9,
             1/9, 1/9, 1/9]
```



STEP 4、對 STEP3 模糊影像 做正規化 得正規化影像 (加速後面運算)

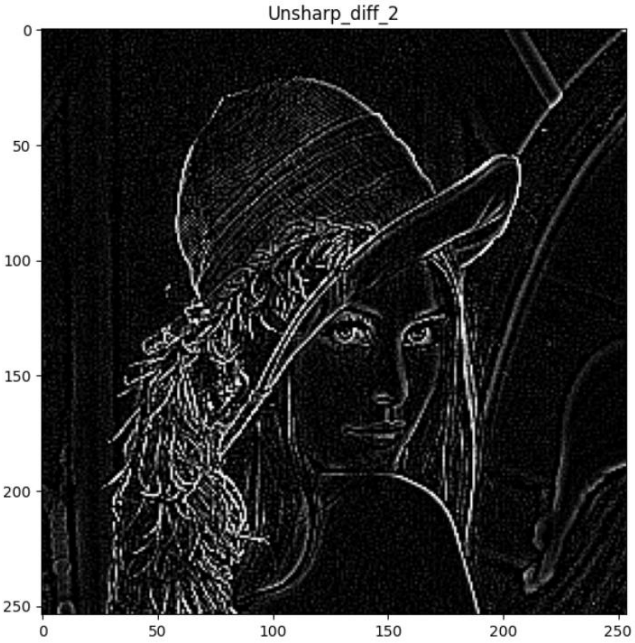
```
# STEP 4、對STEP3模糊影像 做正規化 得正規化影像 (加速後面運算)
img_blur_diff1=np.zeros(img_padded.shape,dtype=np.uint8)
img_blur_diff1=img_mean_diff1/255
img_blur_diff1=np.clip(img_blur_diff1,0,255) # boundary check 超過邊界拉回在邊界上
```



STEP 5、對 STEP1 灰階圖片做 Laplacian 得二階微分 Edge 影像

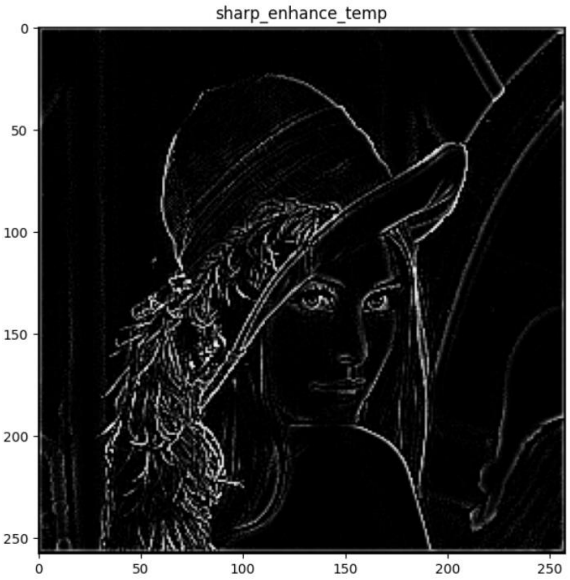
```
laplace_mask = [-1, -1, -1,
                -1, 8, -1,
                -1, -1, -1]
```

```
# STEP 5、對STEP1灰階圖片做Laplacian 得二階微分Edge影像
img_diff2=convolution(img_padded,laplace_mask,pad)
```



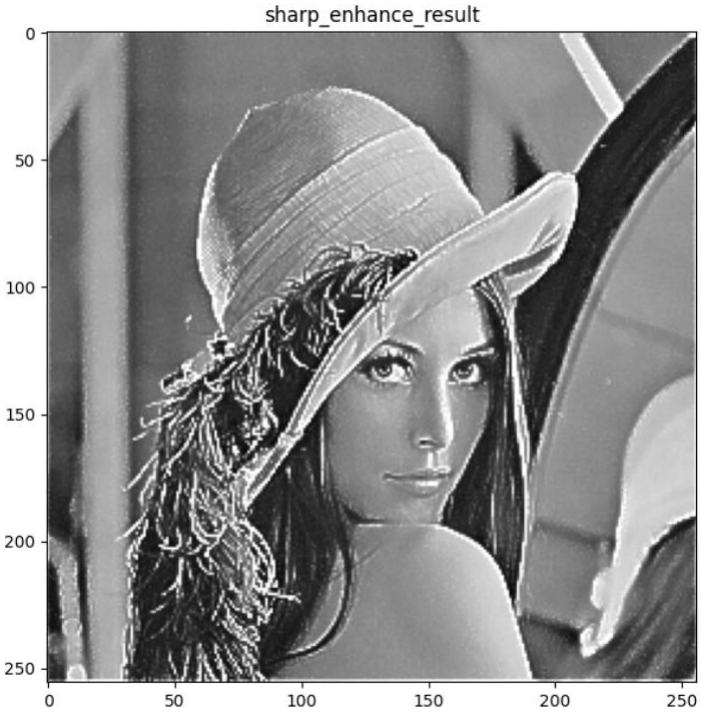
STEP6、將 STEP 4 正規化影像 乘以 STEP 5 二階微分 Edge 影像

```
# STEP6、將 STEP 4 正規化影像 乘以 STEP 5 二階微分Edge影像
img_enhance_temp=np.zeros(img_padded.shape)
img_enhance_temp=img_blur_diff1*img_diff2
img_enhance_temp=np.clip(img_enhance_temp,0,255)# boundary check 超過邊界拉回在邊界上
```



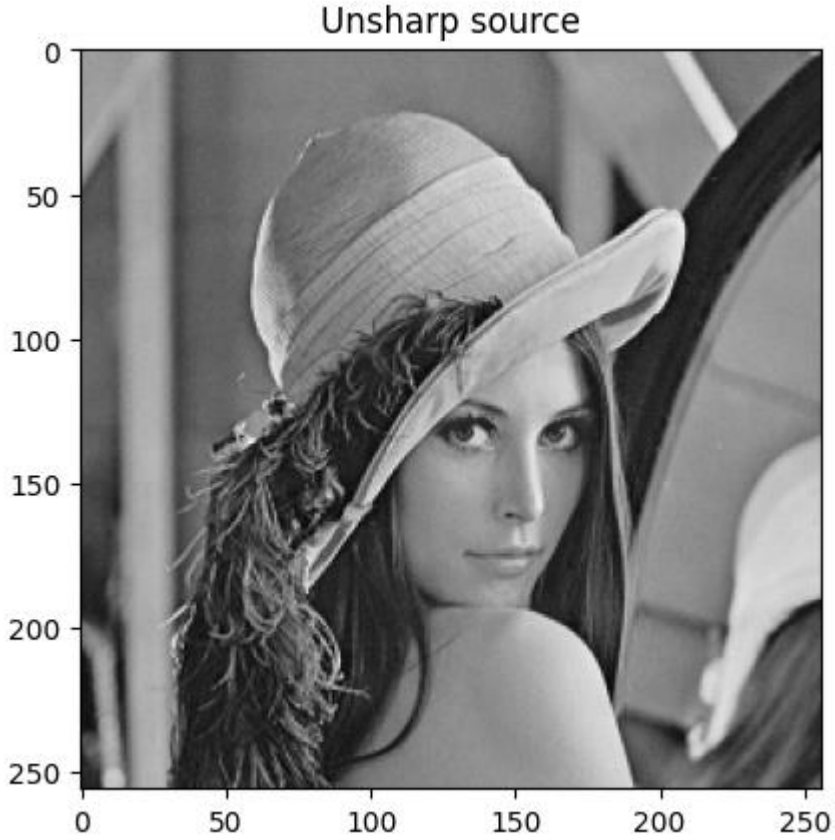
STEP7、STEP6 加上 STEP1 原圖得銳化無雜訊結果

```
# STEP7、STEP6 加上 STEP1原圖得銳化無雜訊結果
img_enhance=np.zeros(img_padded.shape)
img_enhance=img_enhance_temp+img_padded
img_enhance=np.clip(img_enhance,0,255)# boundary check 超過邊界拉回在邊界上
```

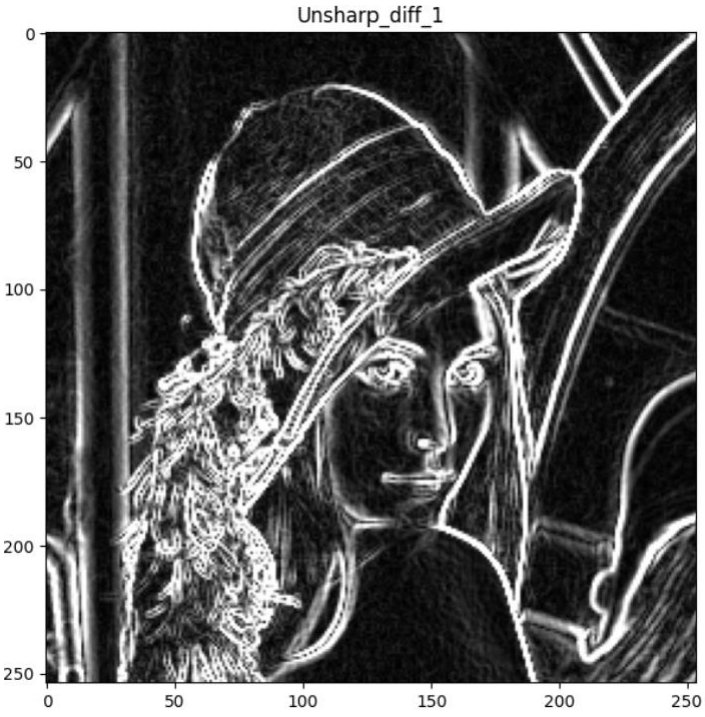


結果

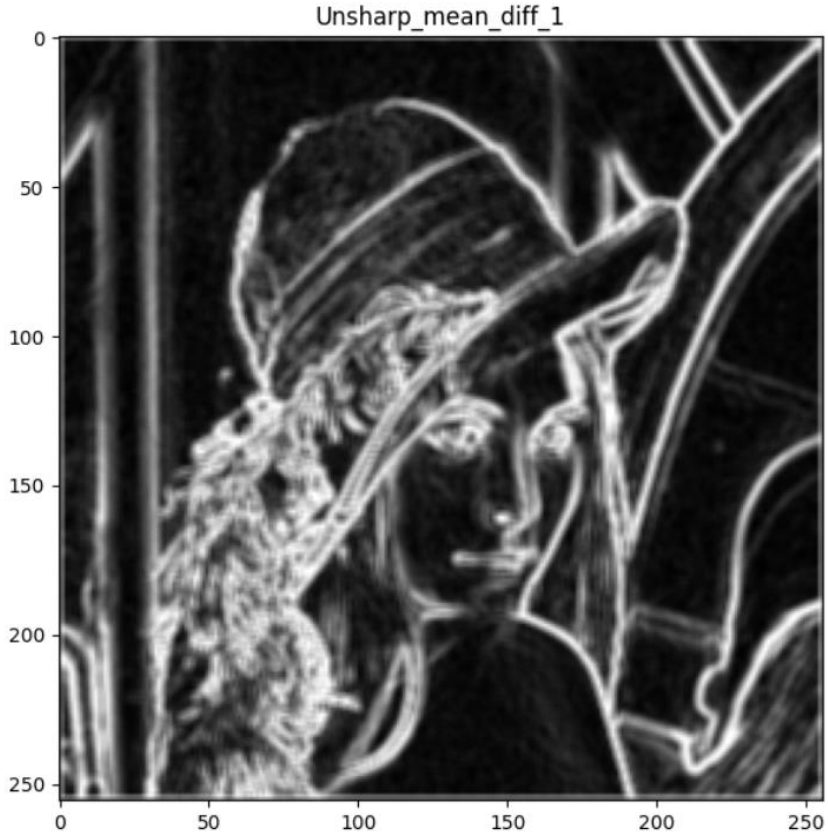
(1) 原灰階圖



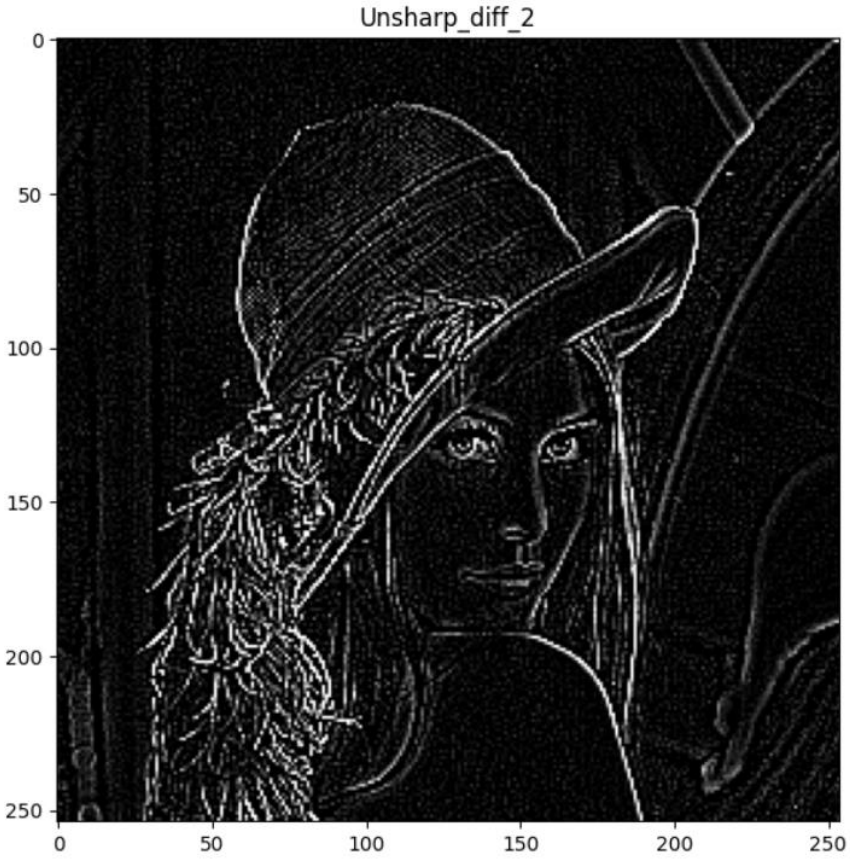
(2) 一階微分 Edge 影像



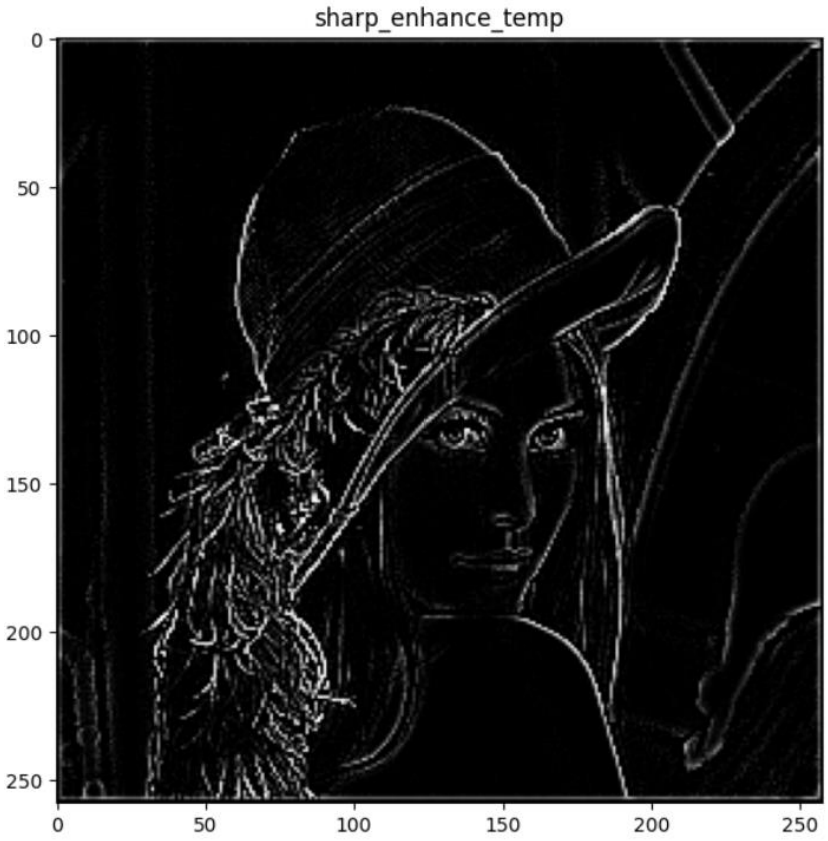
模糊化(去雜訊結果)



(3) Laplacian 二階微分 Edge 影像



(4) 正規化 0.0~1.0 和 Laplacian 二階微分 Edge 影像 相乘



(5) (4) 加上原始影像 (1) 得銳化無雜訊結果