

影像處理 HW3-4

312512049 電控碩 顏志憲

1. Self-designed Lowpass Gaussian Filter Kernels:

```
import cv2
import numpy as np
import os

def sobel_kernel():
    kernel_x = np.array([[ -1,  0,  1], [ -2,  0,  2], [ -1,  0,  1]])
    kernel_y = np.array([[ 1,  2,  1], [ 0,  0,  0], [ -1, -2, -1]])
    return kernel_x, kernel_y

def laplacian_kernel():
    kernel = np.array([[ -1, -1, -1], [ -1,  9, -1], [ -1, -1, -1]])
    return kernel

def kernel_filter(img, kernel, img_pad):
    img_filter = cv2.filter2D(img_pad, -1, kernel)
    img_filter = img_filter[kernel.shape[0] // 2 : kernel.shape[0] // 2 + img.shape[0] ,
                           kernel.shape[1] // 2 : kernel.shape[1] // 2 + img.shape[1]]
    return img_filter

def filter(img, kernel):
    kernel_size = kernel.shape[0]
    img_pad = cv2.copyMakeBorder(img, kernel_size // 2, kernel_size // 2, kernel_size // 2,
                                   kernel_size // 2, cv2.BORDER_REPLICATE)
    filtered_img = kernel_filter(img, kernel, img_pad)
    return filtered_img

def sobel (img):
    kernel_x, kernel_y = sobel_kernel()
    sobel_x = filter(img, kernel_x)
    sobel_x = cv2.convertScaleAbs(sobel_x)
    sobel_y = filter(img, kernel_y)
    sobel_y = cv2.convertScaleAbs(sobel_y)

    sobel_xy = cv2.add(sobel_x, sobel_y)
    sobel_xy = cv2.normalize(sobel_xy, None, 0, 255, cv2.NORM_MINMAX)
    return sobel_xy

def laplacian (img):
    kernel = laplacian_kernel()
    laplacian_img = filter(img, kernel)
    laplacian_img = cv2.normalize(laplacian_img, None, 0, 255, cv2.NORM_MINMAX)
    return laplacian_img
```

sobel_kernel 和 laplacian_kernel 分別定義了 Sobel 和 laplacian 的 filter kernel，而 filter 對影像進行填充以解決邊界問題後，調用 kernel_filter 對影像進行卷積，並去除掉邊界外的填充。

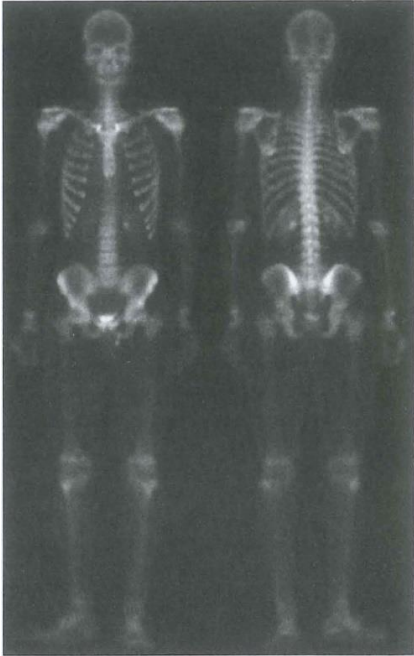
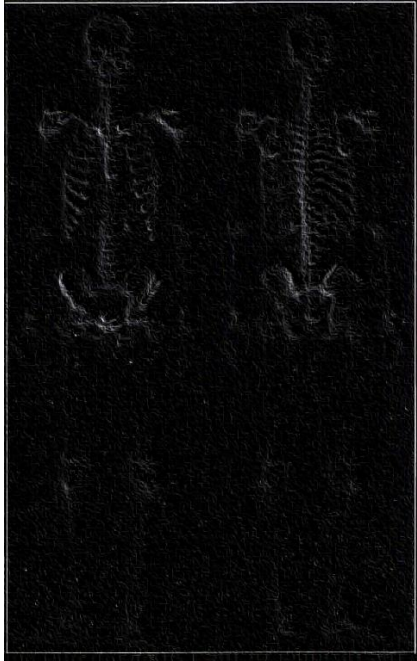
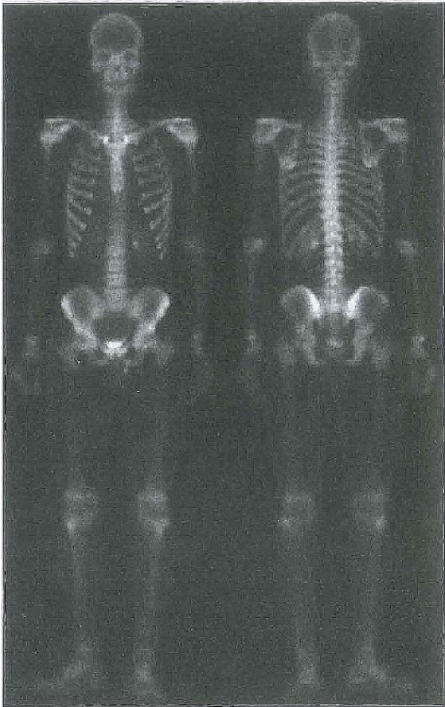
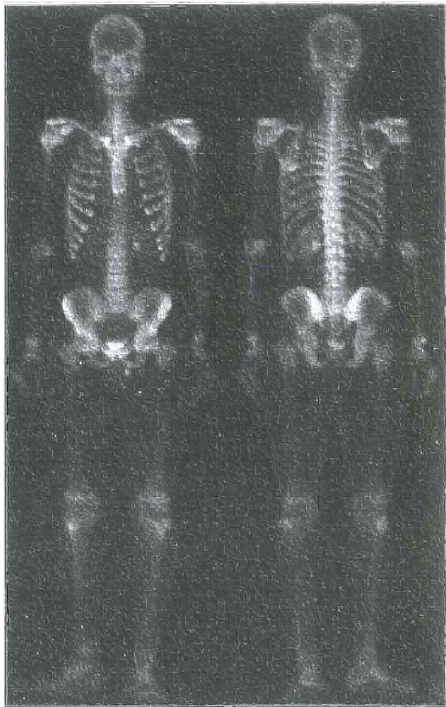
Sobel 實現了 Sobel 濾波器，首先分別對圖像的 X 和 Y 方向利用定義的 sobel kernel 和 filter 進行 Sobel filter 後進行絕對值操作，最後將兩個結果進行加權合併並進行歸一化得到最終的結果；laplacian 則實現了 laplacian filter，利用定義好的 laplacian kernel 和 filter 進行 laplacian filter 後，進行歸一化得到最終的結果。

```
if __name__ == '__main__':
    img_1 = cv2.imread('Bodybone.bmp')
    img_2 = cv2.imread('fish.jpg')
    # sobel
    sobel_xy_1 = sobel(img_1)
    sobel_xy_2 = sobel(img_2)
    # laplacian
    laplacian_img_1 = laplacian(img_1)
    laplacian_img_2 = laplacian(img_2)
    # combine two filters
    # bodybone
    result_1 = cv2.addWeighted(sobel_xy_1, 1, laplacian_img_1, 1, 0)
    result_1 = cv2.normalize(result_1, None, 0, 255, cv2.NORM_MINMAX)
    # fish
    result_2 = cv2.addWeighted(sobel_xy_2, 1, laplacian_img_2, 1, 0)
    result_2 = cv2.normalize(result_2, None, 0, 255, cv2.NORM_MINMAX)
    # save bodybone images
    output_dir = 'img1_output'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    cv2.imwrite(os.path.join(output_dir, 'original_image_1.png'), img_1)
    cv2.imwrite(os.path.join(output_dir, 'sobel_image_1.png'), sobel_xy_1)
    cv2.imwrite(os.path.join(output_dir, 'laplacian_image_1.png'), laplacian_img_1)
    cv2.imwrite(os.path.join(output_dir, 'final_image_1.png'), result_1)
    # save fish images
    output_dir = 'img2_output'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    cv2.imwrite(os.path.join(output_dir, 'original_image_2.png'), img_2)
    cv2.imwrite(os.path.join(output_dir, 'sobel_image_2.png'), sobel_xy_2)
    cv2.imwrite(os.path.join(output_dir, 'laplacian_image_2.png'), laplacian_img_2)
    cv2.imwrite(os.path.join(output_dir, 'final_image_2.png'), result_2)
```

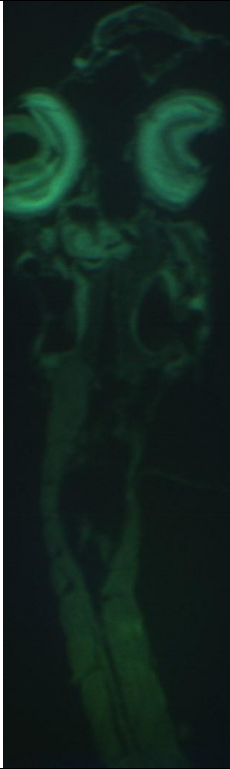



整個流程做法如下：讀取圖片後，首先對圖像進行 Sobel filter，接著進行 Laplacian filter，隨後將兩個結果進行合併後進行歸一化得到最後得銳化結果。在經過手調參數後，最終 sobel kernel 使用與課本相同的方式，而 laplacian 則使用鄰域為負一而中間為 9 的三乘三的 kernel。

銳化結果：

1. Checkerboard1024-shaded.tif:

Origin	Sobel
	
Laplacian	Final
	

2. N1.bmp:

Origin		Sobel	
			
Laplacian		Final	
			

2. Comment:

Sobel 運算子是離散型的差分算子，用於計算圖像亮度梯度的近似值，Laplacian 運算子是二階微分算子，對於階躍的邊緣，會有 zero-crossing 發生，也就是邊緣點兩旁的像素異號，在不考慮周圍灰度差時可以用來進行邊緣檢測，在結果可以發現，sobel 能夠進行更精確的邊緣偵測但忽略了較模糊的邊緣，結果較不受噪聲影響，而 Laplacian 能夠得到更廣泛的邊緣，同時保留了較細節的原圖特徵，但容易受到噪聲影響，因此將兩結果進行合併後能夠消除彼此的缺點得到更好被邊緣銳化的圖像。