# DIP HW 3-4

311505011 人工智慧碩二 黃丰楷

## 1. Please design a highboost method including the Sobel and Laplacian filter in p.183-195 to enhance the image, ‘bodybone.bmp’ as Fig. 3.49 (e). Please describe the your highboost filter, procedures, final enhanced image and print out the source code? (40)

The image processing I designed can be described in 8 steps:

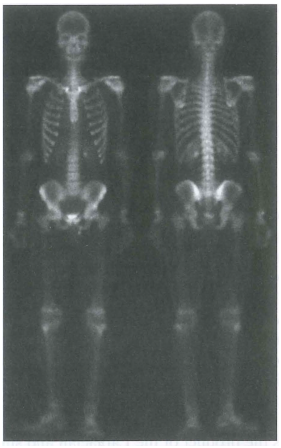
1. Load the original image **(a)**
2. Apply Laplacian filter with kernel $K$ on the original image to get the laplacian **(b)** , where

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

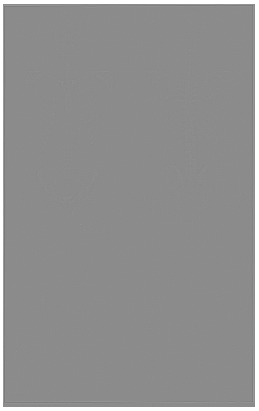
3. Add the original image **(a)** and the Laplacian **(b)** to get the sharpened image 1 **(c).**
4. Compute Sobel gradient **(d)** with component with $g_x$ and $g_y$, where

$$g_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix},\ g_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

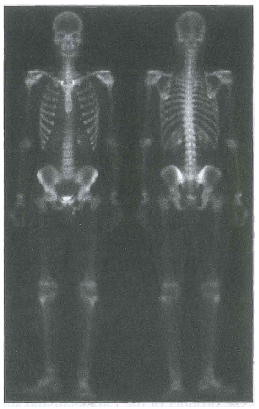
5. Smooth the Sobel gradient **(d)** using a box filter of size 5x5 to get **(e).**
6. Obtain the mask **(f)** by the product of Laplacian **(b)** and Smoothed Sobel **(e)**
7. Get sharpened image 2 **(g)** by adding the original image **(a)** and the mask **(f)**
8. Applying a power-law transformation with $\gamma = 0.5, c = 1$ to sharpened image 2 **(g)** to get the final result **(h).**



**(a)** Original image



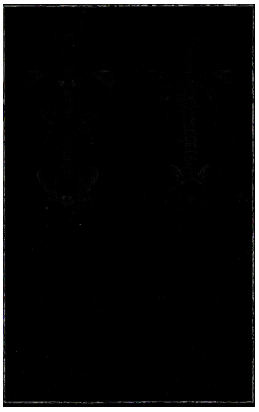
**(b)** Laplacian



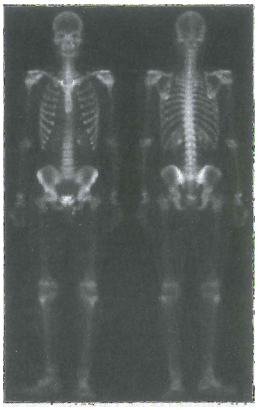
**(c)** Sharpened 1

(**d)** Sobel gradient

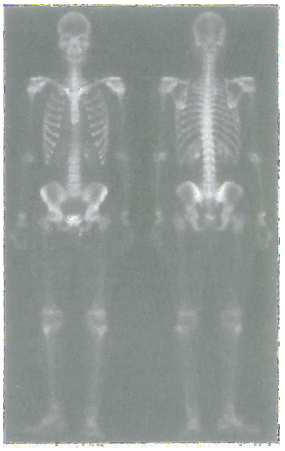
**(e)** Smoothed Sobel



**(f)** Mask



**(g)** Sharpened 2



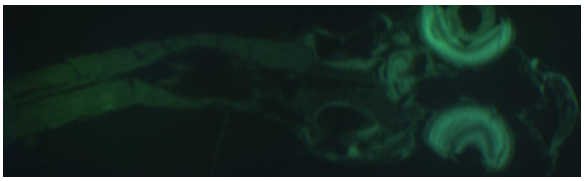
**(h)** Final result

- code

```
import cv2
import numpy as np

img = cv2.imread("./Bodybone.bmp")


laplacian_kernel = np.array([[-1,-1,-1],
                             [-1, 8,-1],
                             [-1,-1,-1]])

laplacian = cv2.filter2D(img, cv2.CV_16S, laplacian_kernel)
laplacian_save = (255*((laplacian-np.min(laplacian))/ (np.max(laplacian)-np.min(laplacian)))).astype(np.uint8)

lap_sharpend = img + laplacian
sobel_x = cv2.Sobel(img , cv2.CV_16S, 1, 0, ksize=3)
sobel_y = cv2.Sobel(img , cv2.CV_16S, 0, 1, ksize=3)
sobel = np.abs(sobel_x) + np.abs(sobel_y)

box_kernel = np.ones((5,5),np.float32)/25
sobel_blur = cv2.filter2D(sobel, cv2.CV_16S, box_kernel)

mask = (laplacian * sobel_blur)
mask[mask<0] = 0
mask = (255*((mask - np.min(mask)) / (np.max(mask) - np.min(mask)))).astype(np.uint8)
soblap_sharpened = img + mask


c = 1
gamma = 0.5

powerlaw = soblap_sharpened**gamma*c
powerlaw = (255*((powerlaw - np.min(powerlaw)) / (np.max(powerlaw) - np.min(powerlaw)))).astype(np.uint8)


titles = ['original_1', 'lap_1', 'lap_sharp1', 'sobel1', 'sobel_blur1', 'mask1', 'soblap_sharpened1', 'power_law1']

outputs = [img, laplacian_save, lap_sharpend, sobel, sobel_blur, mask, soblap_sharpened, powerlaw]

for idx, title in enumerate(titles):
    cv2.imwrite(f"./{title}.png", outputs[idx])
```

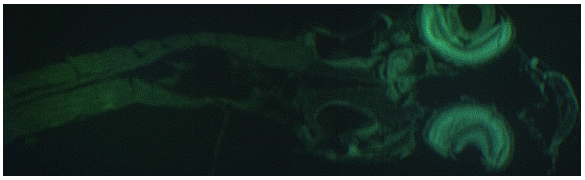
## 2. Repeat (1) steps in the image ‘fish.jpg’? (40)

The size of processing steps (5) is changed to 3x3, and the $\gamma$ and $c$ are set to 2 and 0.6, respectively. The other steps are same as those in question 1. The result images are showed as follow:

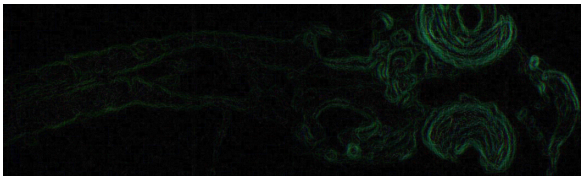


**(a)** Original image

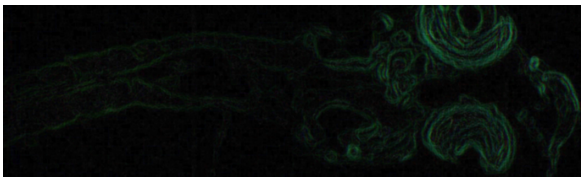
**(b)** Laplacian



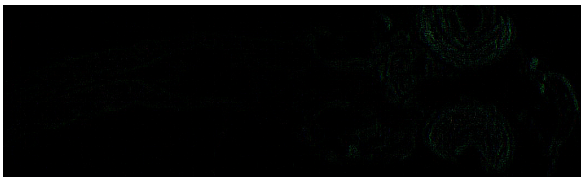
**(c)** Sharpened 1



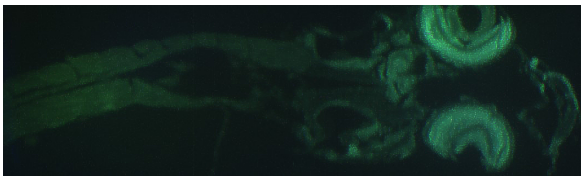
(**d)** Sobel gradient



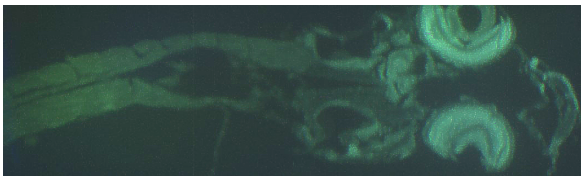
**(e)** Smoothed Sobel



**(f)** Mask



**(g)** Sharpened 2



**(h)** Final result

- code

```
import cv2
import numpy as np


img = cv2.imread("./fish.jpg")


laplacian_kernel = np.array([[-1,-1,-1],
                             [-1, 8,-1],
                             [-1,-1,-1]])

laplacian = cv2.filter2D(img, cv2.CV_16S, laplacian_kernel)
laplacian_save = (255*((laplacian-np.min(laplacian))/ (np.max(laplacian)-np.min(laplacian)))).astype(np.uint8)

lap_sharpend = img + laplacian
sobel_x = cv2.Sobel(img , cv2.CV_16S, 1, 0, ksize=3)
sobel_y = cv2.Sobel(img , cv2.CV_16S, 0, 1, ksize=3)
sobel = np.abs(sobel_x) + np.abs(sobel_y)

box_kernel = np.ones((3,3))*(1/9)
```

```
sobel_blur = cv2.filter2D(sobel, cv2.CV_16S, box_kernel)

mask = (laplacian * sobel_blur)
mask[mask<0] = 0
mask = (255*((mask - np.min(mask)) / (np.max(mask) - np.min(mask)))).astype(np.uint8)
soblap_sharpened = img + mask



c = 2
gamma = 0.6
powerlaw = soblap_sharpened**gamma*c
powerlaw = (255*((powerlaw - np.min(powerlaw)) / (np.max(powerlaw) - np.min(powerlaw)))).astype(np.uint8)


titles = ['original_2', 'lap_2', 'lap_sharp2', 'sobel2', 'sobel_blur2', 'mask2', 'soblap_sharpened2', 'power_law2']

outputs = [img, laplacian_save, lap_sharpend, sobel, sobel_blur, mask, soblap_sharpened, powerlaw]

for idx, title in enumerate(titles):
    cv2.imwrite(f"./{title}.png", outputs[idx])
```

## 3. Please comment and compare your two designed filters and results ?(20)

In the process of image manipulation, a fusion of the Laplacian filter and the Sobel filter is orchestrated. Within the skeletal imagery, the regions encompassing the wrists, hands, ankles, and feet serve as exemplary instances of this phenomenon. The skeletal framework, in particular, showcases a heightened prominence, notably in the structure of the arm and leg bones. An observation worth noting is the subtle demarcation of the body's outline and its underlying tissue. The revelation of such intricacies, through the expansion of the dynamic spectrum of intensity levels, begets an amplification of noise. Nevertheless, the ultimate outcome culminates in a markedly substantial visual spectacle. However, the portrayal of the fish image manifests a deeper shade in contrast to its rendition in the initial inquiry, and its texture exudes a heightened delicacy. It is discernible that the Sobel filter yields a gradient output featuring slender lines delineating the contours. Consequently, in formulating the box filter for the purpose of mollifying the Sobel gradient, a reduction in the kernel size is deemed necessary. Additionally, owing to the luminosity discrepancy, adjustments are made to the parameters governing the power-law transformation in the final phase.