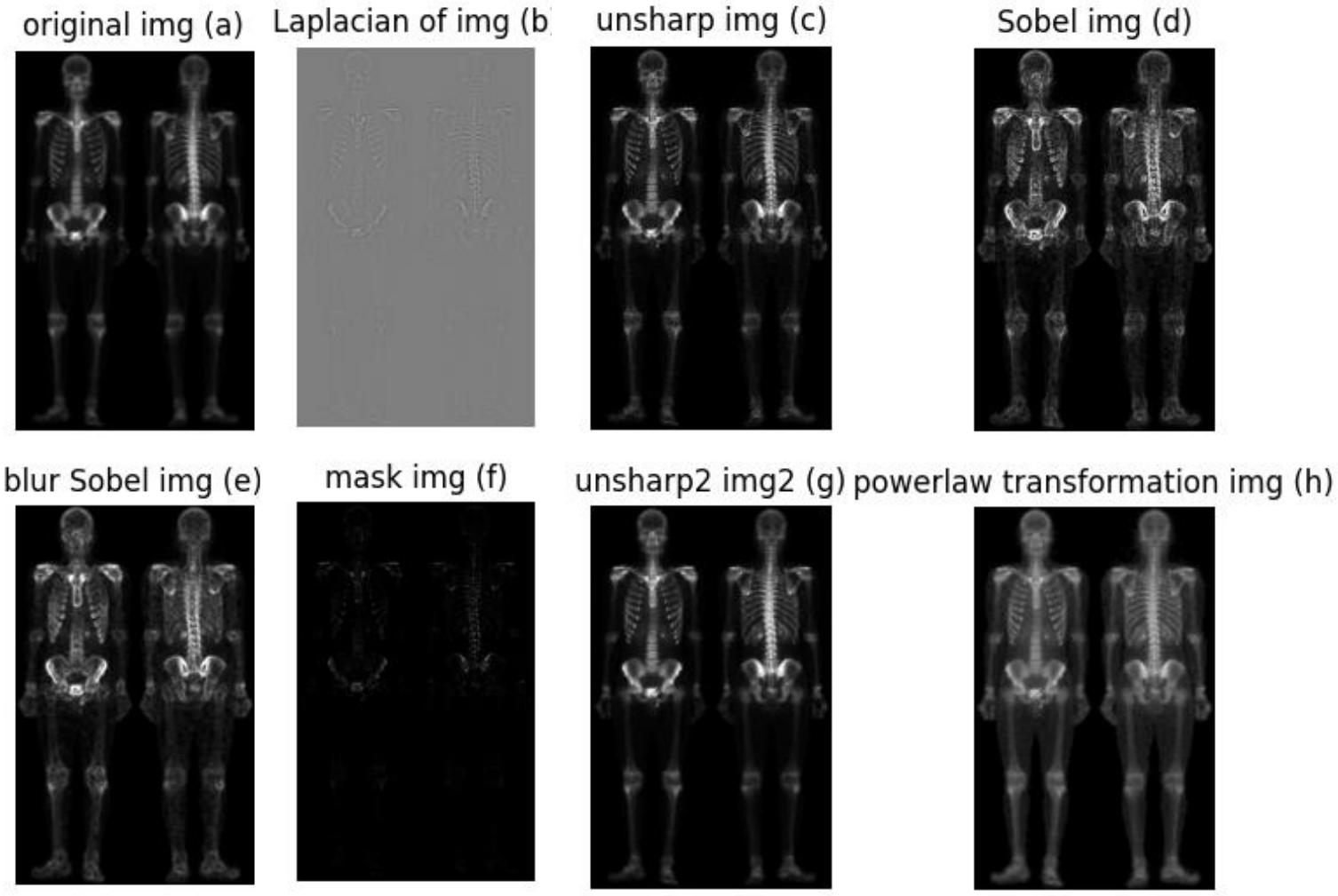
數位影像處理 DIP Homework Chapter 3_4 (100pts)

1. Please design a highboost method including the Sobel and Laplacian filter in pp.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)

使用 digital image processing 官網原圖 :
http://www.imageprocessingplace.com/root_files_V3/image_databases.htm

依照課本 P.193~P.195 的處理方式做出來的 (a)~ (f)圖:



以下對 (a)~(f) 各 procedure 說明:

流程:

(a) Original image → **(b)** Laplacian image → **(c)** unsharp image by (a) + (b)

(a) Original image →

(d) Sobel image by adding abs(Sobelx) and abs(Sobely) →

(e) smoothed Sobel image (by 5*5 box filter) →

(f) mask image by (b)* (f) →

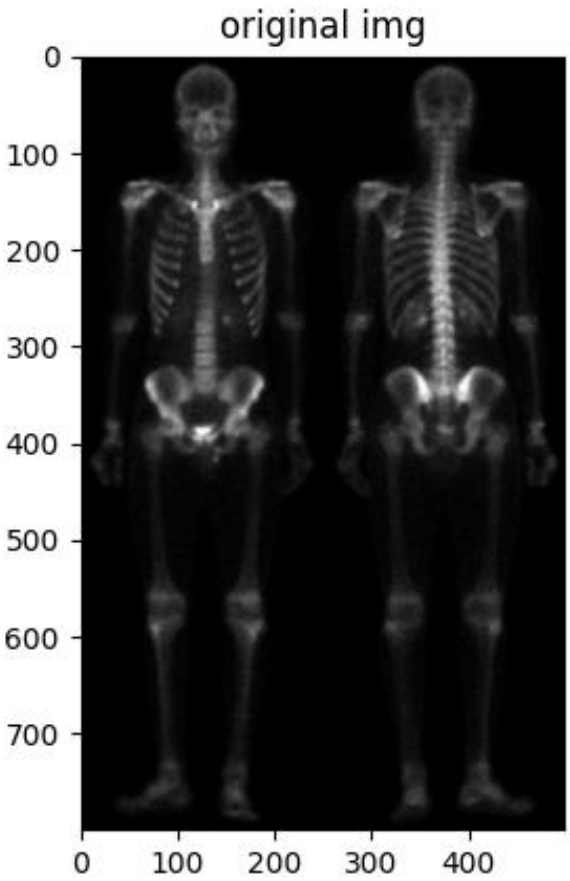
(g) unsharp image2 by (a) + (g) →

(h) Final result obtained by applying a powerlaw transformation on (g)

Code and result

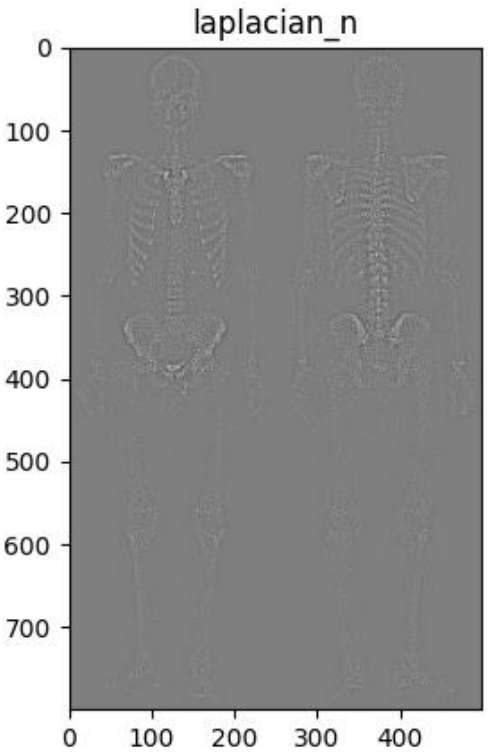
(a) Original image

```
38  img = cv2.imread("Body.tif")
39  print(f"img.dtype = {img.dtype}")
40  print(img.shape)
41  img = img[:,:,::-1]
42  plt.imshow(img), plt.title("original img")
43  plt.show()
```



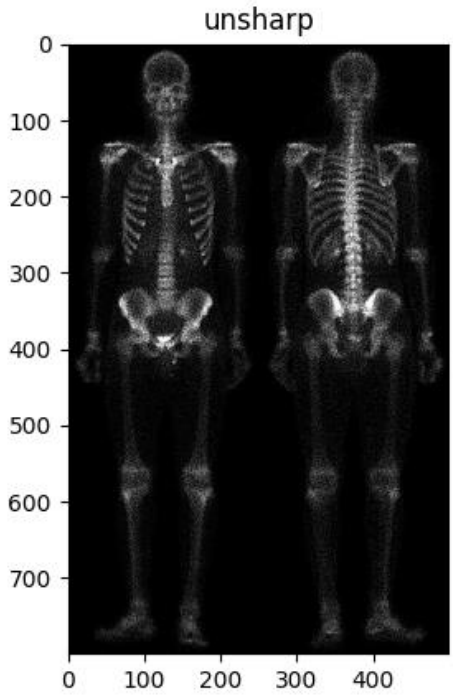
(b) Laplacian image

```
## use self-defined laplacian kernel
laplacian_kernel = np.array([[-1,-1,-1],
                             [-1, 8,-1],
                             [-1,-1,-1]])

laplacian = cv2.filter2D(img, cv2.CV_16S, laplacian_kernel)  # ddepth cv2.CV_16S : 16-bit signed integers

laplacian_n = (((laplacian - laplacian.min())/ (laplacian.max() - laplacian.min()))*255).astype(np.uint8)

plt.imshow(laplacian_n), plt.title("laplacian_n")
plt.show()
```



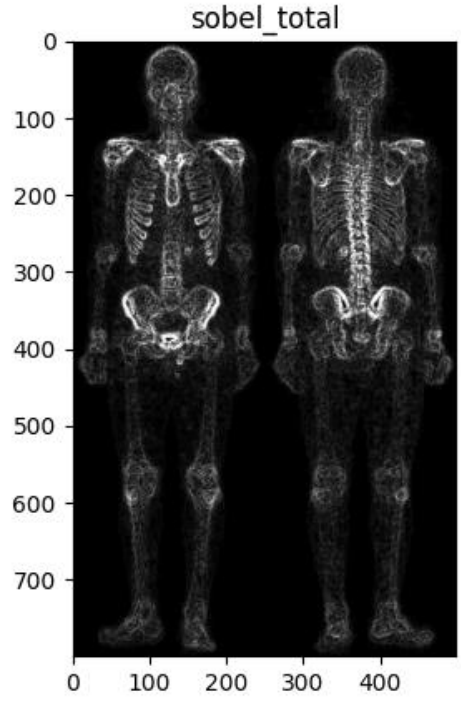
(c) unsharp image by (a) + (b)

```
### unsharp image by img + laplacian (c)
unsharp = img + laplacian

plt.imshow(unsharp), plt.title("unsharp")
plt.show()
```



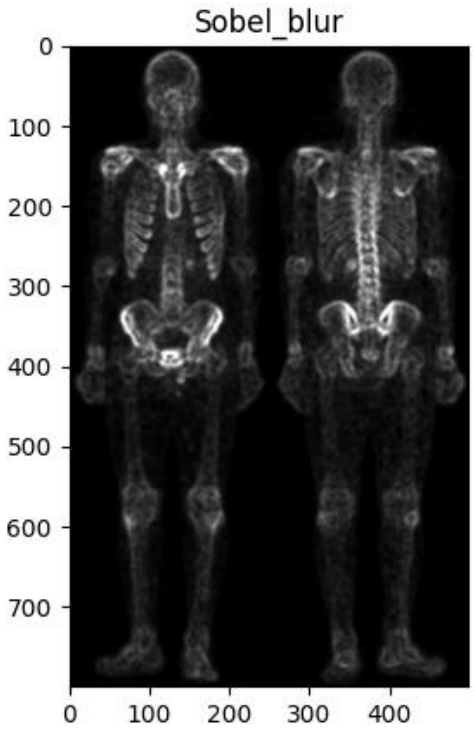
(d) Sobel image by adding abs(Sobelx) and abs(Sobely)

```
## sobelx
sobelx = cv2.Sobel(img, cv2.CV_16S, 1, 0, ksize=3)  # x
## sobely
sobely = cv2.Sobel(img, cv2.CV_16S, 0, 1, ksize=3)

sobel_total = np.abs(sobelx) + np.abs(sobely)
#print(sobel_total.dtype)  #int16
#print(f"sobel_total.max(), sobel_total.min() = {sobel_total.max(), sobel_total.min()}")
plt.imshow(sobel_total), plt.title("sobel_total")
plt.show()
```



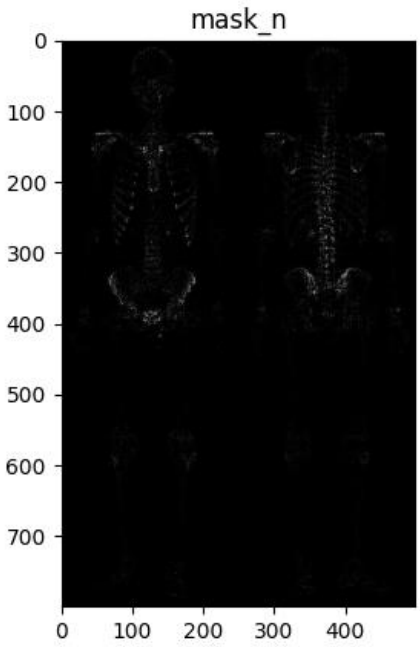
(e) smoothed Sobel image (by 5*5 box filter)

```
box_filter = np.ones((5,5))*(1/25)
Sobel_blur = cv2.filter2D(sobel_total, cv2.CV_16S, box_filter)
#print(f"Sobel_blur.max(), Sobel_blur.min() = {Sobel_blur.max(), Sobel_blur.min()}")
plt.imshow(Sobel_blur), plt.title("Sobel_blur")
plt.show()
```



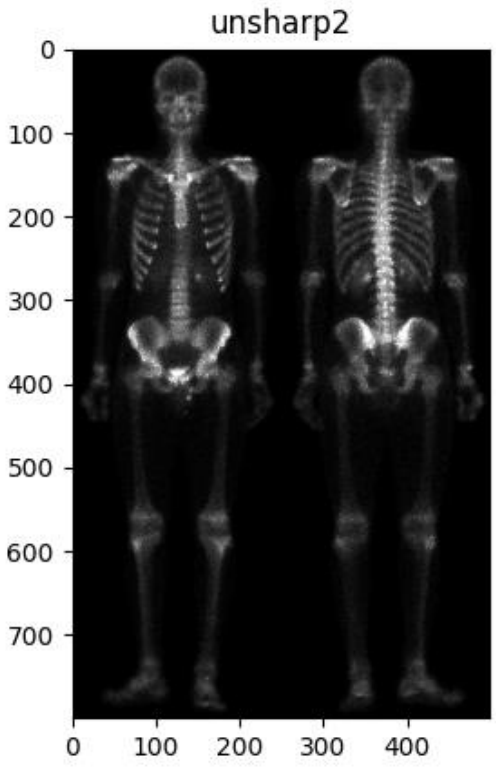
(f) mask image by (b)* (f)

```
5   # Mask image formed by the product of Laplacian and Sobel_blur. (f)
6   mask = Sobel_blur * laplacian
7   #print(mask.dtype)  #int16
8   #print(f"mask.max(), mask.min() = {mask.max(), mask.min()}")
9   mask[mask<0] = 0
10
11  mask_n = (((mask - mask.min()) / (mask.max() - mask.min()))*255).astype("uint8")
12  # print(f"mask_n.max(), mask_n.min() = {mask_n.max(), mask_n.min()}")
13  plt.imshow(mask_n), plt.title("mask_n")
14  plt.show()
```



(g) unsharp image2 by (a) + (g)

```
# unsharp2 image obtained by the adding images img and mask_n. (g)
unsharp2 = img.astype("uint16") + mask_n.astype("uint16")
print(f"unsharp2.max(), unsharp2.min() = {unsharp2.max(), unsharp2.min()}")

plt.imshow(unsharp2), plt.title("unsharp2")
plt.show()
```



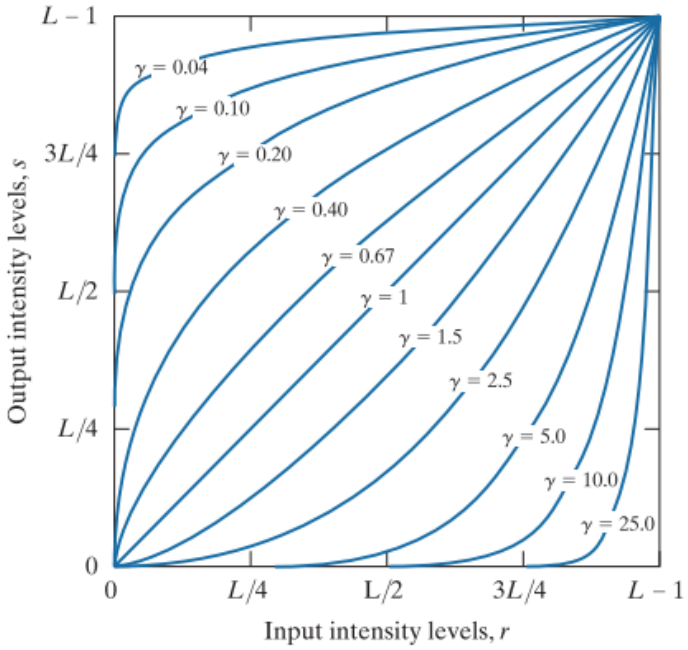
(h) Final result obtained by applying a powerlaw transformation on (g)

POWER-LAW (GAMMA) TRANSFORMATIONS

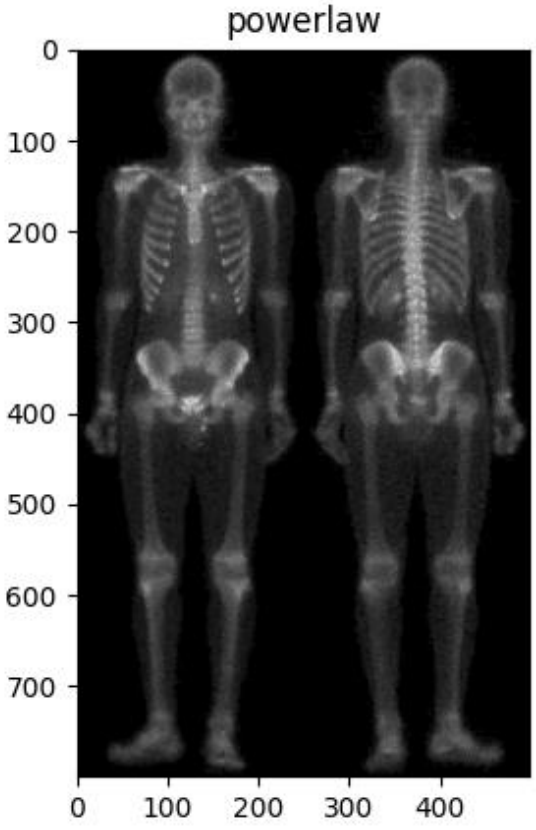
Power-law transformations have the form

$$s = cr^{\gamma} \qquad (3\text{-}5)$$

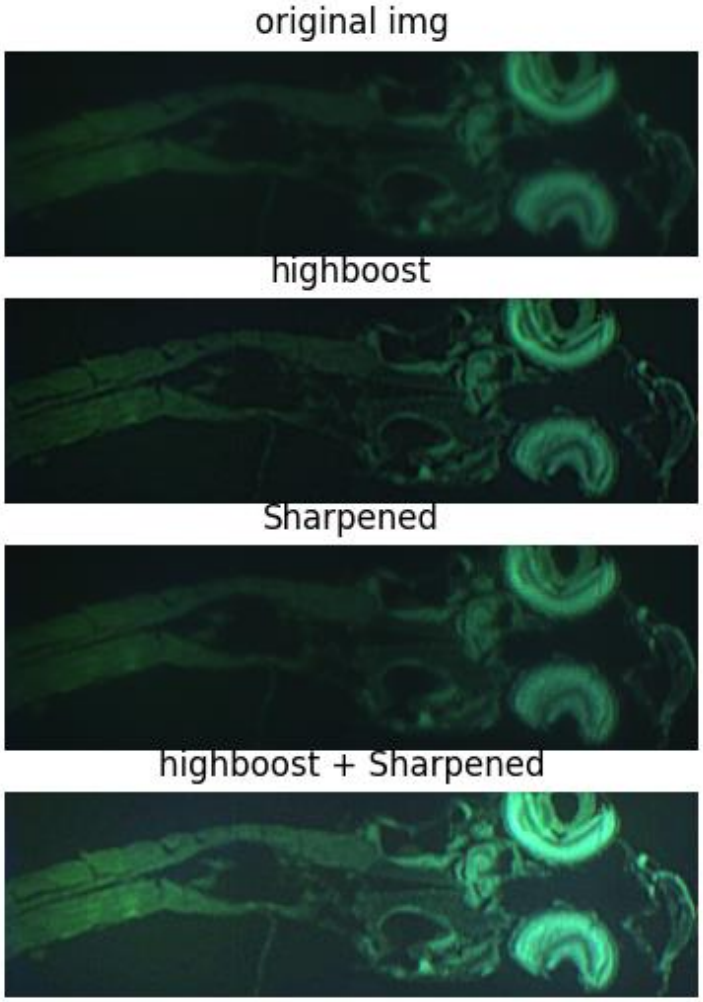
FIGURE 3.6
Plots of the gamma equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). Each curve was scaled *independently* so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.




```
# Final result obtained by applying a powerlaw transformation (h)
c = 1
gamma = 0.5
powerlaw = unsharp2**gamma*c
print(f"powerlaw.max(), powerlaw.min() = {powerlaw.max(), powerlaw.min()}")

powerlaw = (((powerlaw - powerlaw.min()) / (powerlaw.max() - powerlaw.min()))*255).astype("uint8")
plt.imshow(powerlaw), plt.title("powerlaw")
plt.show()
```

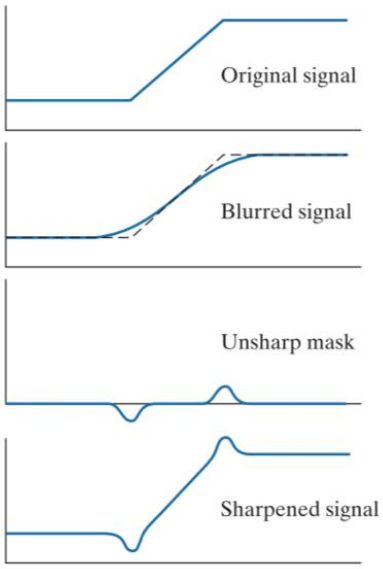

powerlaw
0
100
200
300
400
500
600
700
0
100
200
300
400

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



結果如上，最下面一張為最終結果圖，procedures 如下說明:
(圖片由上往下為**(a)~(d)**)

(b) Highboost :
首先使用 sigma 為 5,kernel size 為(31*31)的 gaussian filter 得到模糊後的原圖，接著將原圖與模糊過的圖相減，得到邊緣，再將邊緣乘以一個大於 1 的 k 值(=1 為 Sharpened image)得到 **highboost** image.




```
14  blur = cv2.GaussianBlur(img, (31, 31), 5, cv2.BORDER_REFLECT)
15
16  mask = img - blur
17  print(mask.max(), mask.min())
18  plt.imshow(mask), plt.title("mask")
19  plt.show()
20
21  unsharp = img + mask
22  plt.imshow(unsharp), plt.title("unsharp")
23  plt.show()
24
25  k = 2
26  highboost = img + k*mask
27  plt.imshow(highboost), plt.title("original highboost")
28  plt.show()
```

(c) Sharpened :

如同課本 p.197 FIGURE 3.57 (g)所做的處理

Original image -> Laplacian image

Original image -> Sobel image -> smoothed Sobel image (by 3*3 box filter) -> Mask image (Laplacian image* smoothed Sobel image) -> **Sharpened** image original image + mask image)

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread("fish.jpg")
img = img[:,:,::-1]

## use self-designed laplacian kernel
laplacian_kernel = np.array([[-1,-1,-1],
                             [-1, 8,-1],
                             [-1,-1,-1]])

laplacian = cv2.filter2D(img, cv2.CV_16S, laplacian_kernel)  # ddepth = -1

## sobelx
sobelx = cv2.Sobel(img, cv2.CV_16S, 1, 0, ksize=3)  # x
## sobely
sobely = cv2.Sobel(img, cv2.CV_16S, 0, 1, ksize=3)

sobel_total = np.abs(sobelx) + np.abs(sobely)

# Sobel image smoothed with a 3×3 box filter.
box_filter = np.ones((3,3))*(1/9)
Sobel_blur = cv2.filter2D(sobel_total, cv2.CV_16S, box_filter)

# Mask image formed by the product of Laplacian and Sobel_blur.
mask = Sobel_blur * laplacian
mask[mask<0] = 0
mask2 = (((mask - mask.min()) / (mask.max() - mask.min()))*255).astype("uint8")
print(f"mask2.max(), mask2.min() = {mask2.max(), mask2.min()}")
```

```python

# Sharpened image obtained by the adding images img and mask2.
Sharpened = img.astype("uint16") + mask2.astype("uint16")

plt.imshow(Sharpened), plt.title("Sharpened")
plt.show()
```

(d)最後將上述的 Highboost 及 Sharpened image 加起來，可得到最終結果圖。

```python
plt.subplot(411), plt.imshow(img), plt.axis("off"), plt.title("original img")
plt.subplot(412),plt.imshow(highboost), plt.axis("off"), plt.title("highboost")
plt.subplot(413),plt.imshow(Sharpened), plt.axis("off"), plt.title("Sharpened")
plt.subplot(414),plt.imshow(highboost + Sharpened), plt.axis("off"), plt.title("highboost + Sharpened")
plt.show()
```

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

這兩個影像分別用了許多方法得到最終結果，過程中都有使用到的是兩個 highpass filter : Laplacian filter, sobel filter，兩者皆用於取出邊緣，不過使用上需要特別注意，由於 gradient 有從負到正及正到負兩個方向，因此須特別注意經過 filter 出來的影像，pixel value 會超過 0~255 的範圍，在顯示及與其他圖相加上，須根據用途與最終希望呈現的結果去決定，是否做 normalize 或 clipping.

Laplacian filter 與 sobel filter 的比較，可以比較(b) Laplacian 和(d) Sobel total，Sobel total 是結合 Sobelx 及 Sobely 的結果，在邊緣顯示上，比 Laplacian filter 更明顯，但也相對得更易凸顯雜訊，所以(e)才另用 box filter 將其模糊。