Cover Page

影像處理作業1 - Image Sharpening

409410019 資工四 王郁誠

HW due : 5/8 00:00

HW handed in : 5/7

程式執行流程：

執行 `python hw2.py` 後，會以blurry_moon.tif作為input，並跳出一個視窗，代表執行結果。接著取消34行註解，再執行一次，即可看到以skeleton_orig.bmp作為input的執行結果。

Technical description

實作拉普拉斯算子

- function

```python
def laplacian_calc_v1(img: np.ndarray):
    height, width = img.shape
    filter_img = np.zeros((height, width), dtype = np.float32)
    for i in range(1,height-1):
        for j in range(1,width-1):
            filter_img[i,j] = ((img[i-1,j] + img[i+1,j] + img[i,j-1] + img[i,j+1]) - 4*img[i,j])
    result_img = filter_img[1:-1, 1:-1]
    return result_img
```

- main

```python
# laplacian v1
padded_img = cv.copyMakeBorder(gray_img, 1,1,1,1, cv.BORDER_DEFAULT)
padded_img = padded_img.astype(np.float32)
laplacian_filter_img = laplacian_calc_v1(padded_img)
laplacian_img = gray_img + laplacian_filter_img
# laplacian_img = laplacian_img.astype(np.uint8)
```

將原圖(img)先轉成gray img，接著再將其邊緣各padding 1個pixel，再將其丟進function。
在function中，先開一個空白圖，接著遍歷所有pixel進行卷積運算，接著捨去掉外圍邊緣得到大小和原圖相同的sharpened image，並回傳。
得到sharpened image後，將其和原圖相加，得到enhanced image。

註：此for迴圈之卷積運算即為套laplacian filter。

```
[ 0, 1, 0]
[ 1,-4, 1]
[ 0, 1, 0]
```

實作high-boosted filtering

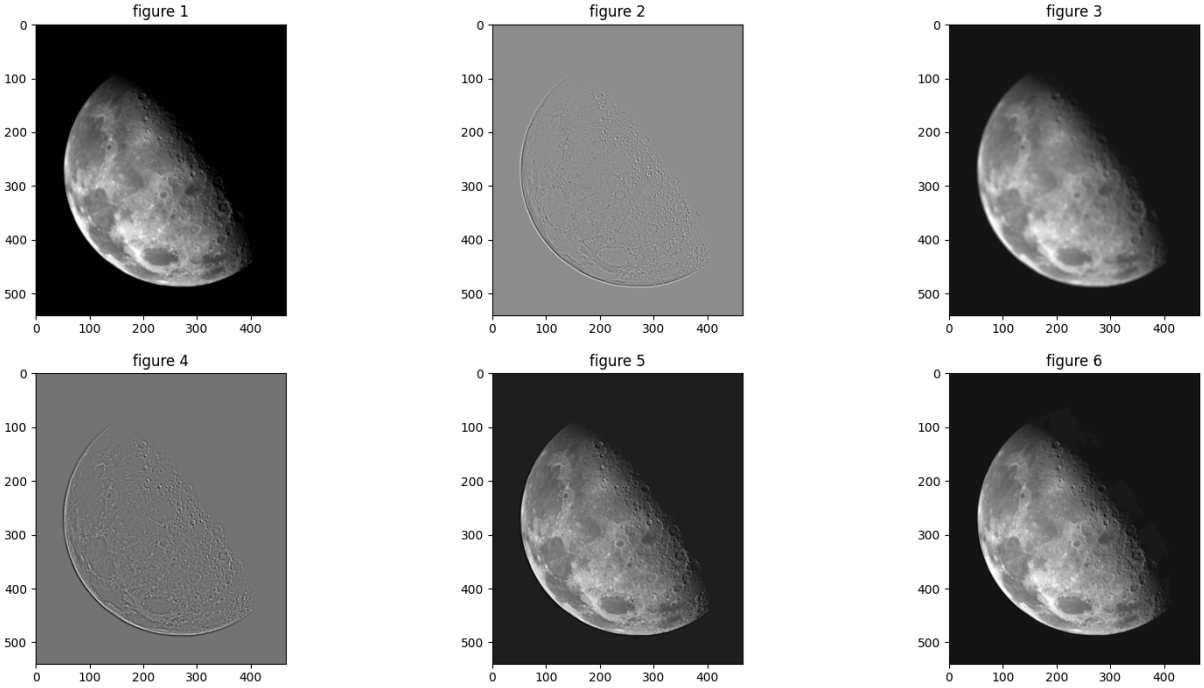
```python
def highboost_calc(img: np.ndarray):
    A = 1.7
    blur_img = cv.GaussianBlur(img, (7,7), 0, 0)
    img2 = img.astype(np.float32)
    sharpened_img = img2 - blur_img
    hb_img_v1 = (A-1) * img2 + sharpened_img
    hb_img_v2 = A * img - laplacian_calc_v1(img)
    return sharpened_img, hb_img_v1, hb_img_v2
```

high-boosted filtering原理為："原圖" - "模糊圖" = sharpened image。
所以我用library function進行高斯模糊得到"模糊圖"，再拿原圖扣掉它，得到sharpened image。
最後，透過第三章PPT P.73的公式，得到high-boosted image，其中：
hb_filter_img_v1, hb_filter_img_v2分別代表透過高斯模糊, laplacian filter得到sharpened image後，再進行後續處理(對應第三章PPT P.73，公式3.7-10, 3.7-11)

Experimental results

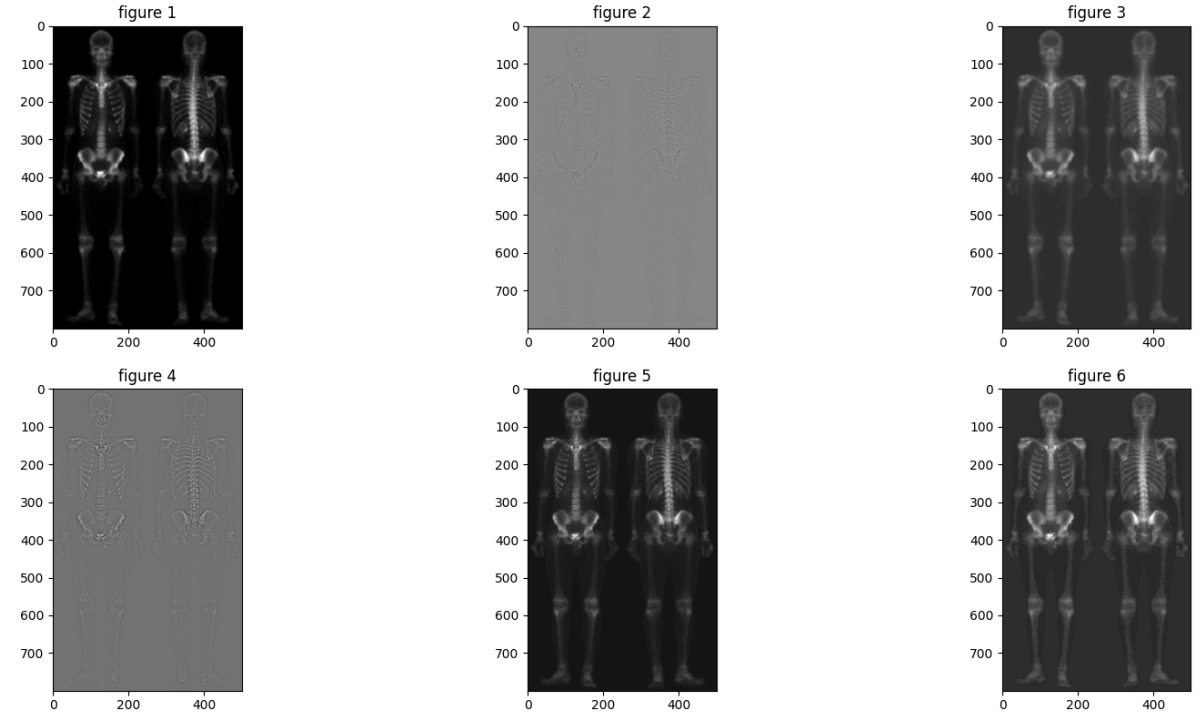
執行程式後，會看到一個視窗，其中有6個figure，分別代表：
figure 1：原圖
figure 2：套完laplacian filter後的sharpened image
figure 3：將原圖加上figure 2的sharpened image
figure 4：將原圖套上高斯模糊得到"模糊圖"後，原圖減去"模糊圖"後得到得sharpened image
figure 5：用高斯模糊得到之sharpened image，進行操作得到的high-boost image
figure 6：用laplacian filter得到之sharpened image，進行操作得到的high- boost image

- blurry_moon.tif



此case中，figure 3看起來有點糊糊的，邊緣感覺也沒有被銳化，不算是成功。而figure 5、6感覺好像有銳利些，應該算成功吧。

- skeleton_orig.bmp



此case也和上面類似，figure 3可能失敗了，而figure5、6感覺有比較銳化，大小腿的部分感覺有變清楚。

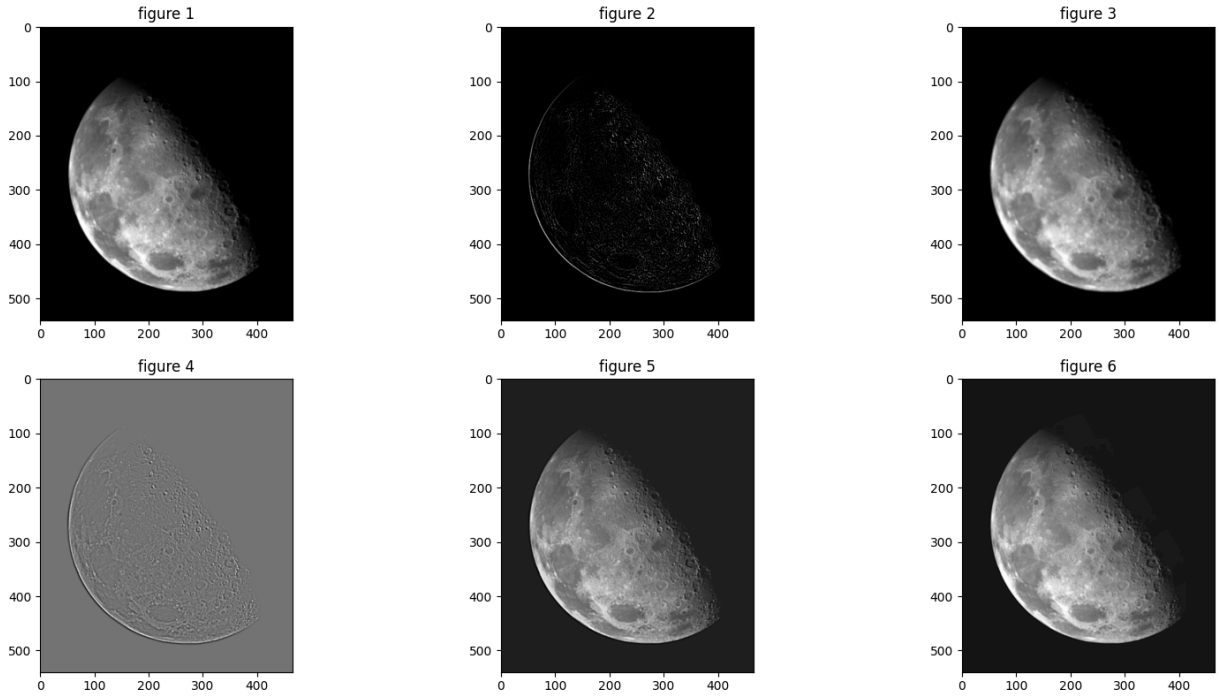
由於我的laplacian filter實作成果不理想，所以我嘗試使用已有的函式來進行卷積運算。(可將code 45~46行取消註解重現)

```
laplacian_filter = np.array([[ 0, 1, 0],
                             [ 1,-4, 1],
                             [ 0, 1, 0]])

def laplacian_calc_v2(img: np.ndarray):
    global laplacian_filter
    filter_img = cv.filter2D(img, -1, laplacian_filter)
    return filter_img
```

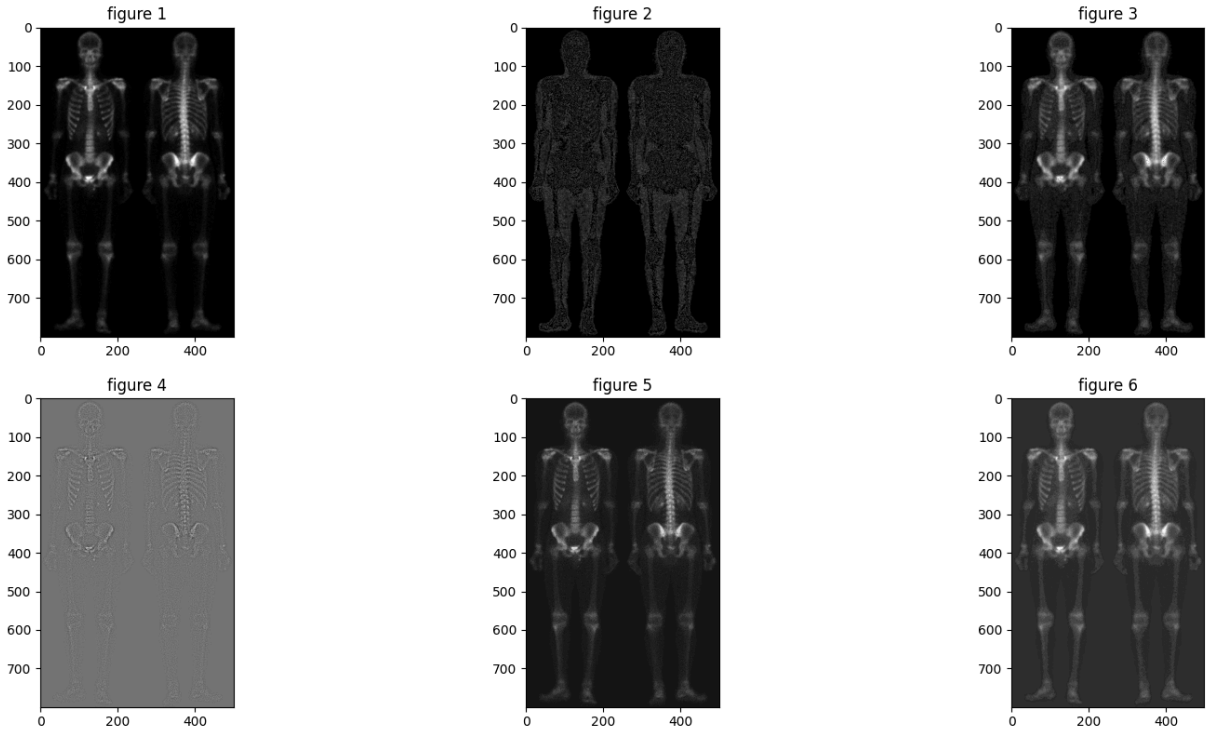
直接使用filter2D進行卷積運算，重新執行一次。

- blurry_moon.tif



這次figure 2和之前的相差甚大，整個顏色都變了。此外，figure 3就沒有比較模糊的問題，邊緣也稍微銳利了些。

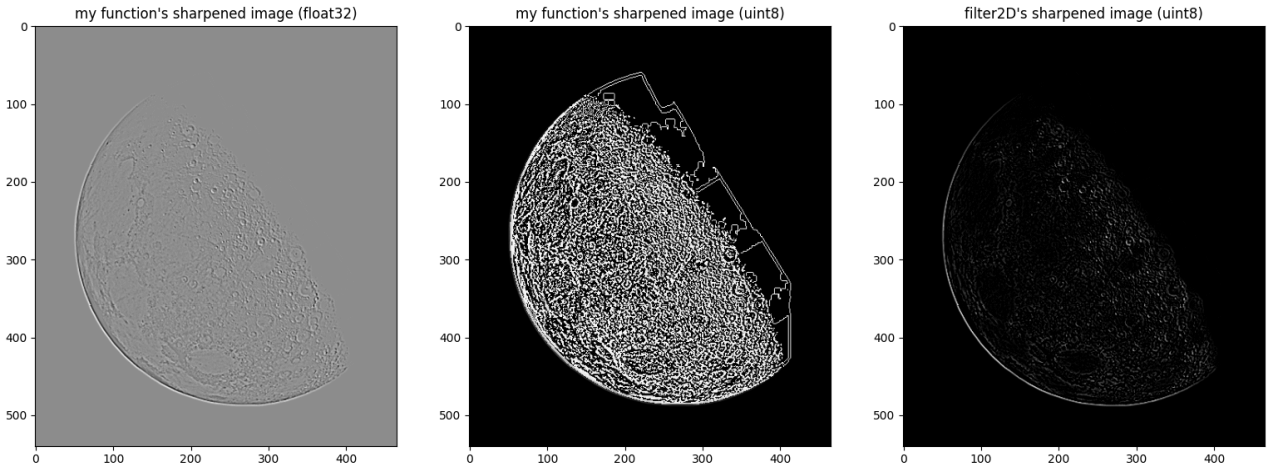
- skeleton_orig.bmp



同樣地，這次figure 2、3和之前也相差很大，這次多了很多輪廓，感覺確實銳化了邊緣。

Discussions & Appendix

這次作業我原本很簡單，就套套filter，然後根據公式將影像進行銳化而已，結果...沒想到比我想像中的難，我不知道為什麼我的filter會跟filter2D的差那麼多。我在計算拉普拉斯算子時，先把原圖給轉成float32，得到的sharpened image並沒有轉成uint8，而filter2D轉出來的資料型態直接是uint8。因此，我也把我的sharpened image型態轉為uint8，比較圖為(code 72~83行)：



可以看出我的sharpened image轉出去就是一坨答辯，而filter2D轉出來的就很好看，而且感覺做完後要轉到uint8它的值域才會是對的，但我的版本一轉過去就變超醜，哀。這是我這次作業遇到最大的挫折，希望能在下次作業前釐清原因，並順利完成下次作業。