Hw2_Image Sharpening (Due. 5/9)

409410014 資工三 柯柏旭 (hand in 5/7)

Technical description

測試環境

```
❯ uname -a
Linux hentci-Aspire-A515-52G 5.15.0-69-generic #76-Ubuntu SMP Fri Mar 17 17:19:29 UTC 2023
 x86_64 x86_64 x86_64 GNU/Linux
```

使用語言:

`python 3.10.6`

library-requirement:

```
opencv_python==4.7.0.72
```

如何執行

```
python3 Image_Sharping.py
```

或是

```
python Image_Sharping.py
```

按`Esc`鍵便會依序顯示以下內容:

```
共8個parts
1. 原始moon
2. 經過laplacian後的moon
3. 經過使用參數A = 1.2, high-boost後的moon
4. 經過使用參數A = 1.7, high-boost後的moon
5. 原始的skeleton
6. 經過laplacian後的skeleton
7. 經過使用參數A = 1.2, high-boost後的skeleton
8. 經過使用參數A = 1.7, high-boost後的skeleton
```

程式碼解釋

1. Laplacian Operator

```python
def Laplacian(img):
    # Create a new image with the same shape as the input image
    lap_img = img.copy()
    # Get the height, width, color channels of the input image
```

```
    height, width, channel = img.shape
    # Loop through every pixel in the image
    for i in range(height):
        for j in range(width):
            cur_sum = 0
            # Loop through the four neighbors of the current pixel
            for k in range(4):
                x = i + dx[k]
                y = j + dy[k]
                # Check if the neighbor is within the image boundaries
                if 0 <= x < height and 0 <= y < width:
                    # Subtract the neighbor's value from the \
                    # current pixel's value
                    cur_sum -= img[x, y, 0]
            # Add 5 times the current pixel's value to the sum
            cur_sum += 5 * img[i, j, 0]
            # standardize the sum to keep it within 0-255 range
            cur_sum = standardize(cur_sum)
            # Set the pixel value in the new image to the standardizeed sum
            lap_img[i, j] = [cur_sum] * 3

    return lap_img
```

經由公式的推導後，我們可以得到`g(x, y)`這個mask:

0	-1	0
-1	5	-1
0	-1	0

接著便使用迴圈對於每一個pixel按照這個mask做image sharping。
其中需要注意的是，在更新pixel value時，不管是邊界還是value都可能會超出範圍外，因此需要特別約束範圍，其中`standardize`這個function就是在做pixel value的constraint:

```
# Define a function to standardize the value and keep it within 0-255 range
def standardize(value):
    return max(0, min(value, 255))
```

2. High-Boost filtering

```
def High_Boost(img, A):
    new_img = img.copy()
    height, width, channel = img.shape
    # Loop through every pixel in the image
    for i in range(height):
        for j in range(width):
            cur_sum = 0
```

```python
            # Loop through the four neighbors of the current pixel
            for k in range(4):
                x = i + dx[k]
                y = j + dy[k]
                # Check if the neighbor is within the image boundaries
                if 0 <= x < height and 0 <= y < width:
                    # Subtract the neighbor's value from the current\
                    # pixel's value
                    cur_sum -= img[x, y, 0]
            # Add (4 + A) times the current pixel's value to the sum
            cur_sum += (4 + A) * img[i, j, 0]
            # standardize the sum to keep it within 0-255 range
            cur_sum = standardize(cur_sum)
            # Set the pixel value in the new image to the standardizeed sum
            new_img[i, j] = [cur_sum] * 3
    # Return the new image
    return new_img
```

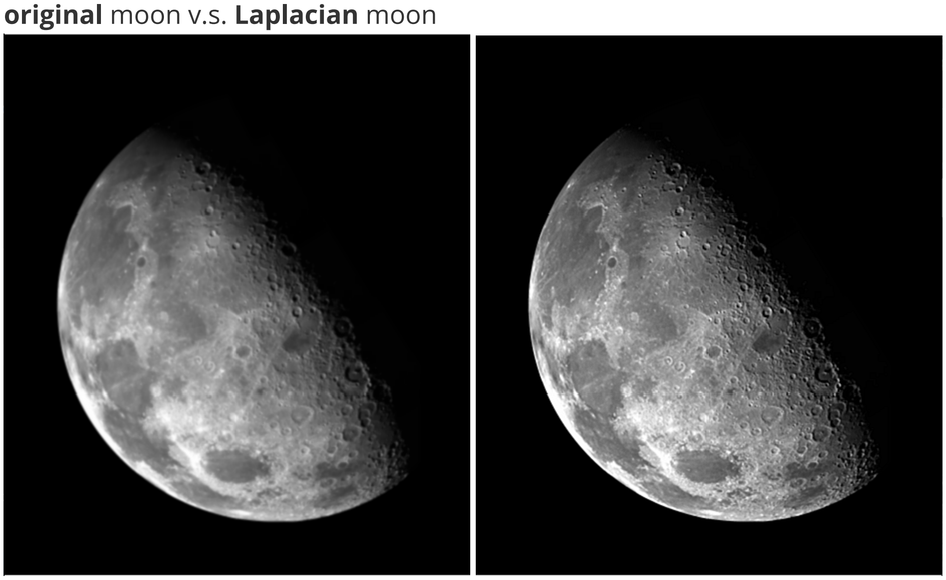
做法和`Laplacian`大同小異，主要的差異在於mask中間值的倍率。
舉例來說，如果`A = 1`，那麼結果就會和laplacian相同。這次的實驗中，則使用了`A = 1.2`和`A = 1.7`來實做High-Boost filtering。

0	**-1**	**0**
-1	A + 4	-1
0	-1	0

Experimental results

blurry_moon.tif

- **original** moon v.s. **Laplacian** moon

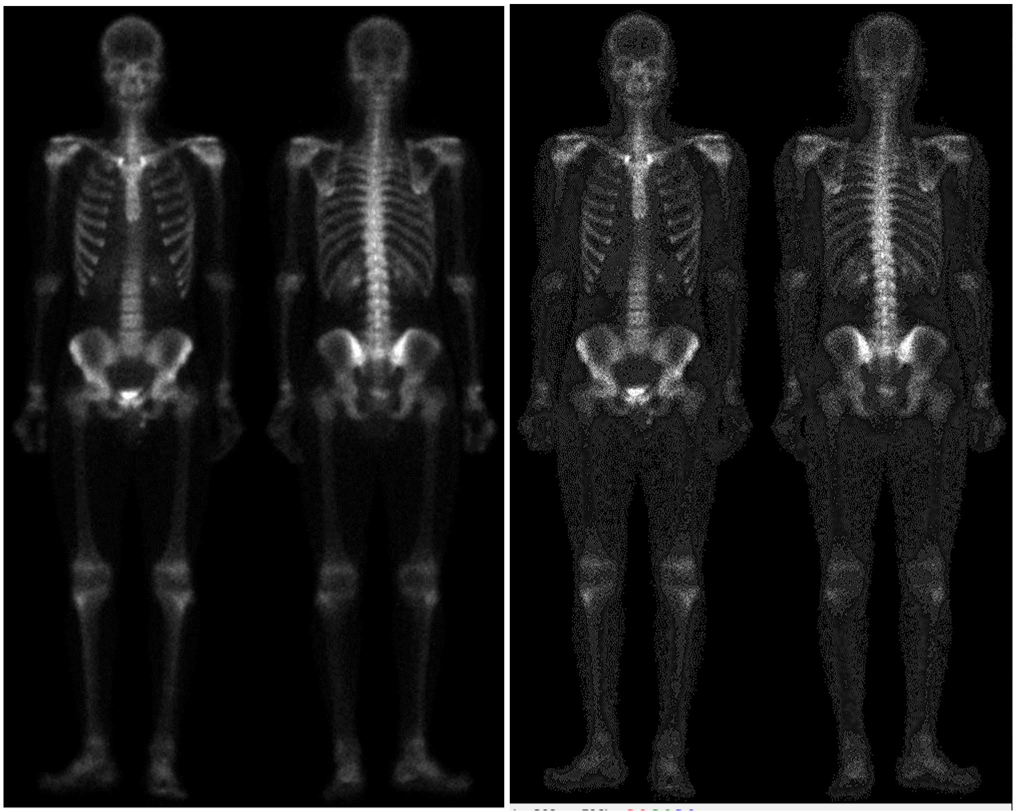


- **original** moon v.s. **High-Boost filtering with A = 1.2** moon

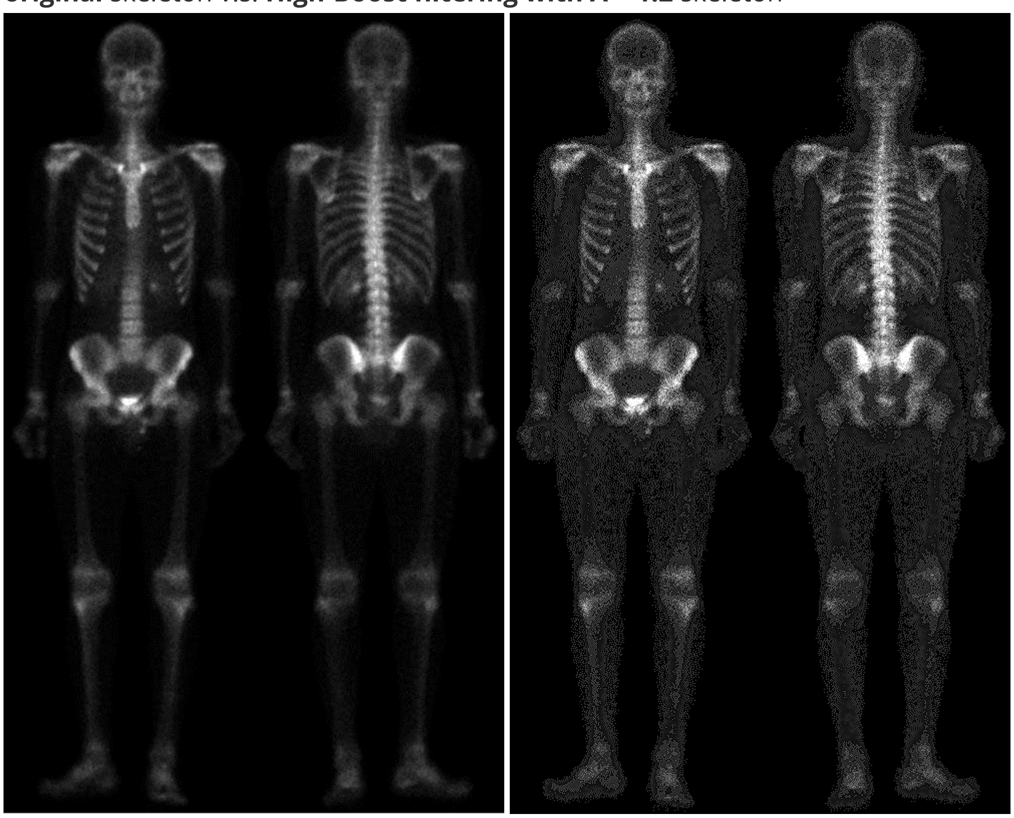
- **High-Boost filtering with A = 1.2** moon v.s. **High-Boost filtering with A = 1.7** moon

skeleton_orig.bmp

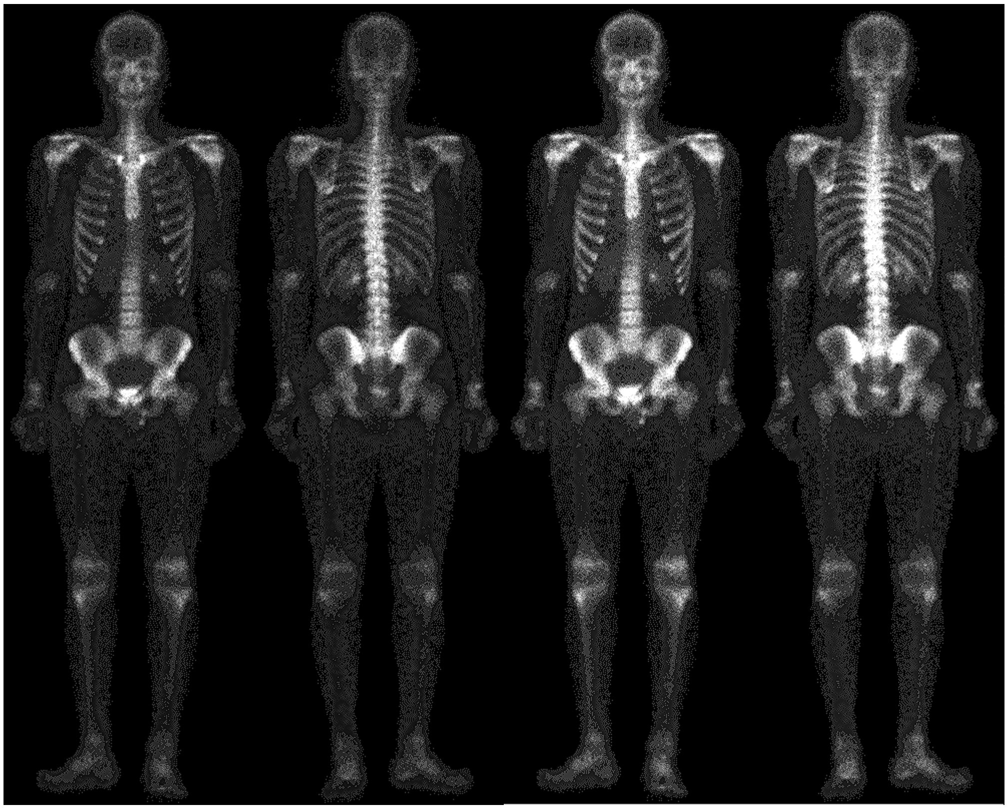
- **original** skeleton v.s. **Laplacian** skeleton



- **original** skeleton v.s. **High-Boost filtering with A = 1.2** skeleton



- **High-Boost filtering with A = 1.2** skeleton v.s. **High-Boost filtering with A = 1.7** skeleton



Discussions

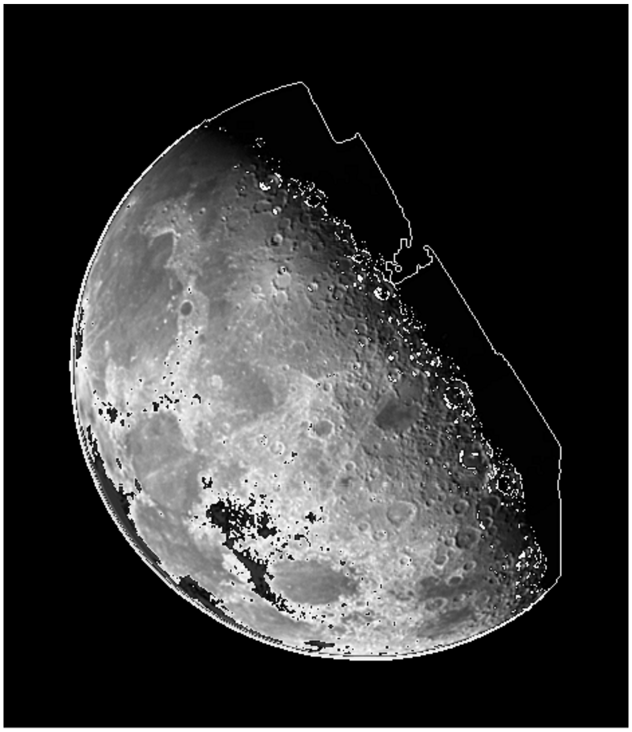
從上方的實驗結果與比較可以發現:

對於moon這張圖，image sharping的結果非常成功。銳利過後整張圖的變得很清楚。其實單做Laplacian效果就很好了，但在High-Boost filtering調整參數A到1.7附近就有點過度了，整個月亮直接閃閃發光。

對於skeleton這張圖，image sharping的結果也非常成功。和moon做比較後可以發現，雖然moon很不適合過高的參數A，但是把這個方法應用在X光圖的時候，高一點的參數A反而使整張圖更明顯。例如參數A = 1.7High-Boost filtering的skeleton這張圖，我個人認為便是裡面最合適的。

心得

這次的作業相比第一次的histogram反而變得簡單許多，不過在實做的過程還是有遇到許多bug，沒說那麼順。像是在update完pixel的值時，有機會把他更新到超過255或小於0。這時候就會直接溢位讓整張圖爆炸，像這樣:



變成還蠻有科幻風的月亮

總而言之，這次的作業真的很令人眼睛為之一亮，沒想到這麼簡單的mask就可以讓圖變得這麼清楚。

References and Appendix

References的部份主要為老師ppt的Ch03, Ch04