影像處理 Lab2 Report 109062110 祝語辰

一、Proj03-01: Image Enhancement Using Intensity Transformations

1. 實作方式：

在 logTransform.m 中，讀取影像的大小，以創建另外一個同大小，且各個 element 的值皆為 1，稱其為 one，然後將影像的 Matrix 和 one 相加，如此一來每個 elements 都不會有 0。最後對整個新 matrix 取 log，也就是將影像的每個 pixels 的 intensity 對應到新的 Intensity。最後乘以 constant C。在我的 main.m 中，我選擇 C = 255/log(256)，目的是為了使轉換後的 image 的 intensity 仍在 0~255 之間。

logTransform.m

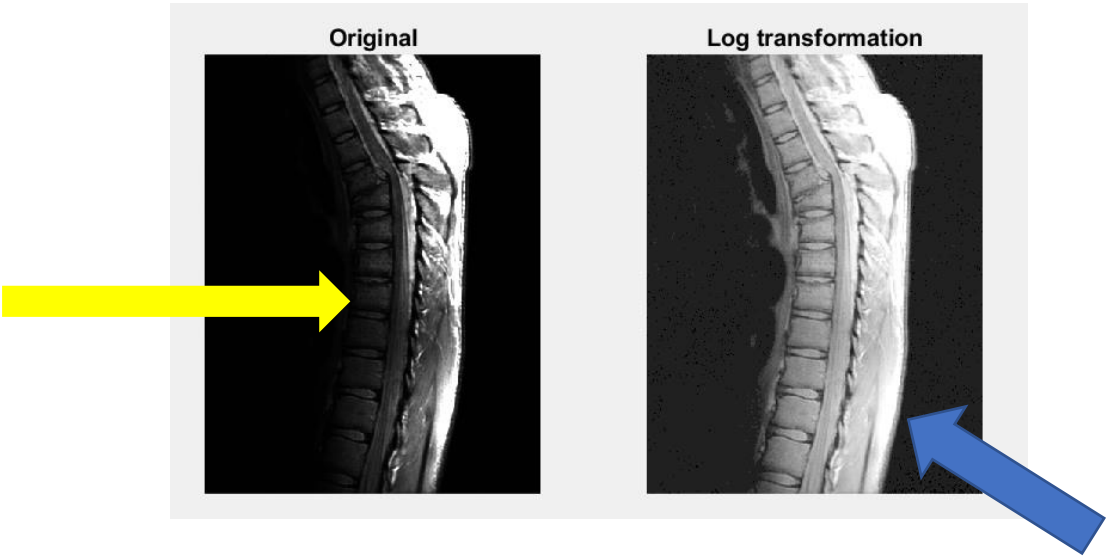
```
function [outputImage] = logTransform(inputImage, c)
    [M,N,C] = size(inputImage);
    one = ones(M,N,C,'double');
    outputImage = c*log(inputImage+one);
end
```

而在 powerlawTransform.m 中，做法也差不多。對影像作 element-wise 的 r 次方，然後在乘上 constant c。

powerlawTransform.m

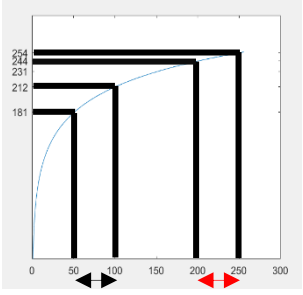
```
function [outputImage] = powerlawTransform(inputImage,c,r)
    temp = c*(inputImage.^r);
    outputImage = temp;
end
```

2. 問題與討論：

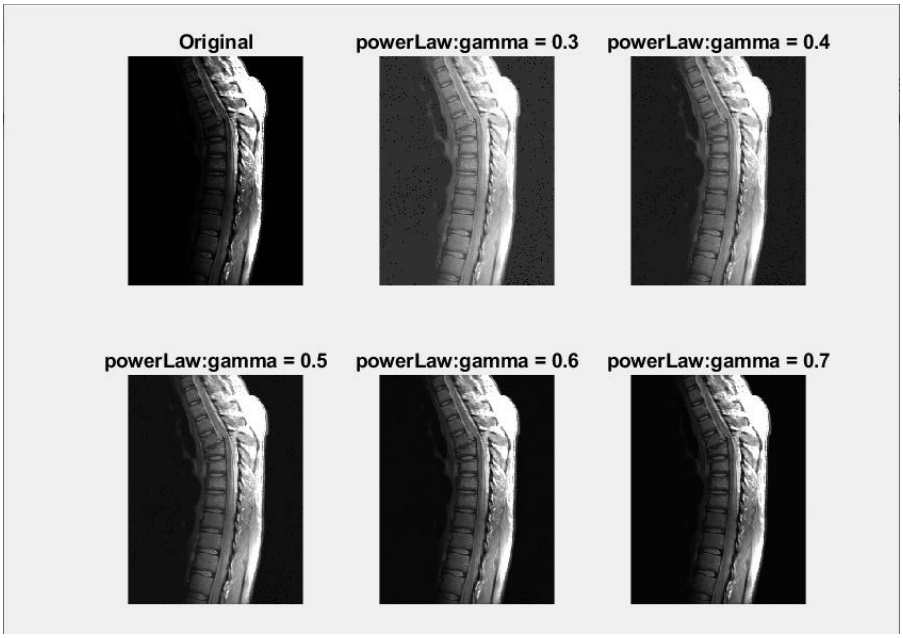


左邊是原圖，右邊是對 intensity 做 log transform 後的圖。

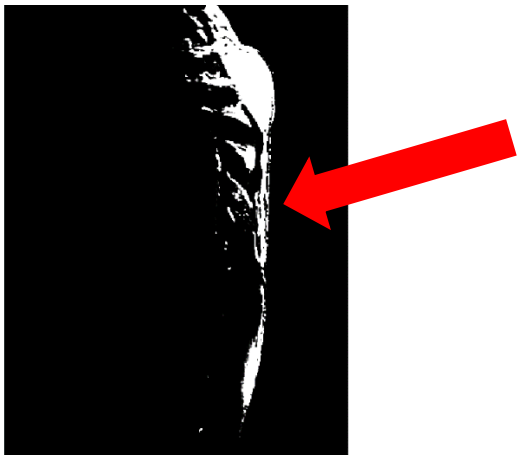
基本上，我就是將 intensity 比較低的 pixels 做明顯的區分。從上圖中，可以看到原圖黃色箭頭處，雖然能看到有東西，但細節上因為太暗不是很清楚。而右圖，就是將這些教案的訊號做出區分，雖然他們都是黑色的，Intensity 可能是 50,54 之類的，在原圖上看起來很暗，但 log transform 就是強化這個差異，所以可以看到右圖中，相同的位置處，能看清楚細節。但相對的，藍色箭頭處，在原圖中，原本是能看清楚細節的，但在處理後，high intensity 的部分被弱化了，變得比較不明顯。下圖可以明顯的看到 high intensity(紅色的部分)，在 log transform 後，被明顯的壓縮。



接下來看 power law transform 的部分。



可以看到，我在上圖的例子中，選用的 gamma 值都小於 1。在這種情形下，power law transform 達到的目的是和 log transform 一樣的，都是要增強原圖偏暗處的 intensity 區別之處。可以看到 gamma 的值越小，對 low intensity 增強的效果就越強，但對 high intensity 的部分，壓縮的程度就更明顯。下圖是一個極端的圖，一樣是使用 power law transform，但是 gamma 值是 20。



在這裡可以看到在圖中紅色的箭頭處被加強，變得明顯。gamma 值大於 1 比較適合用來處理整張影像偏亮時，要增強細節的時候，在這裡是為了要用同樣一張圖說明 power law transform，不然其實不適合對這張圖做 power law transform with gamma > 1。

二、Proj03-02: Histogram Equalization

1. 實作方式：

在 imageHist.m 中，我的作法很直覺，也就是使用 for 迴圈跑，讀取影像的每個 pixel，根據他的 intensity，來決定要對 hist_vector_temp 中哪一個位置加 1。因為 hist_vector_temp 在建的時候，為一個全為零的 1*256 矩陣，所以可以存各個 intensity 出現的次數。

imageHist.m

```
function [hist_vector] = imageHist(input_image)
    hist_vector_temp = zeros(1,256,"uint64");
    [M,N] = size(input_image);
    for i = 1:M
        for j = 1:N
            index = uint8(input_image(i,j));
            hist_vector_temp(index+1) = hist_vector_temp(index+1) + 1;
        end
    end
    hist_vector = single(hist_vector_temp);
end
```

而在 histEqualization.m 中，基本上是實作講義中的這個公式，

- $p_r(r_k) = \frac{n_k}{MN}, k = 0,1,\dots,L-1$
- $s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_k$
  - Or $s_k = s_{k-1} + (L-1)p_r(r_k) = s_{k-1} + \frac{L-1}{MN}n_k, k = 1,2,\dots,L-1$

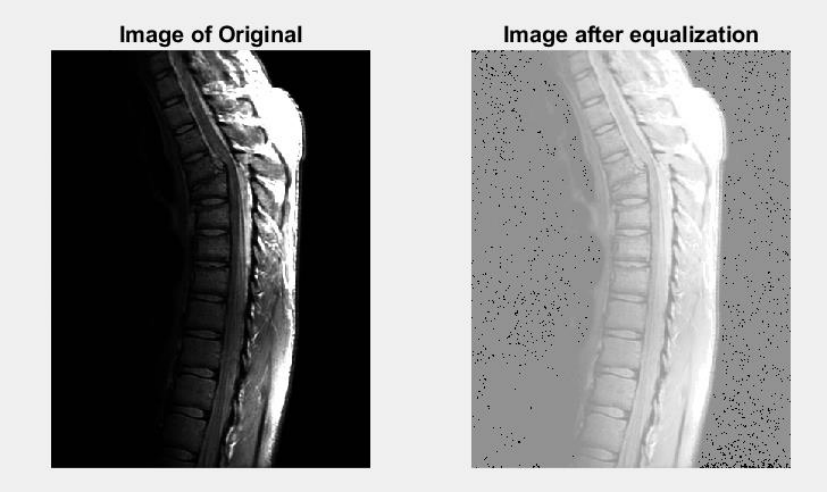
在 histEqualization.m 中，先 call imageHist.m，已取得影像的 histogram，也就是上圖中的 n_k。然後對整個 vector 除以整個影像的

size，就可得到 pr(rk)，也就是以 k 為 index，儲存 pr 的 vector。然後對整個 vector 做處理，使得整個 vector 是以 k 為 index 存 sigma j=0~k,pr(rj)。在我的 program 中，我是另外存在一個 vector S 中。最後將 S 中的每個 element 乘上 255(公式中的 L-1)，就得到 S_k，然後再根據他，利用 for 迴圈，將 image 的 intensity 做轉換。

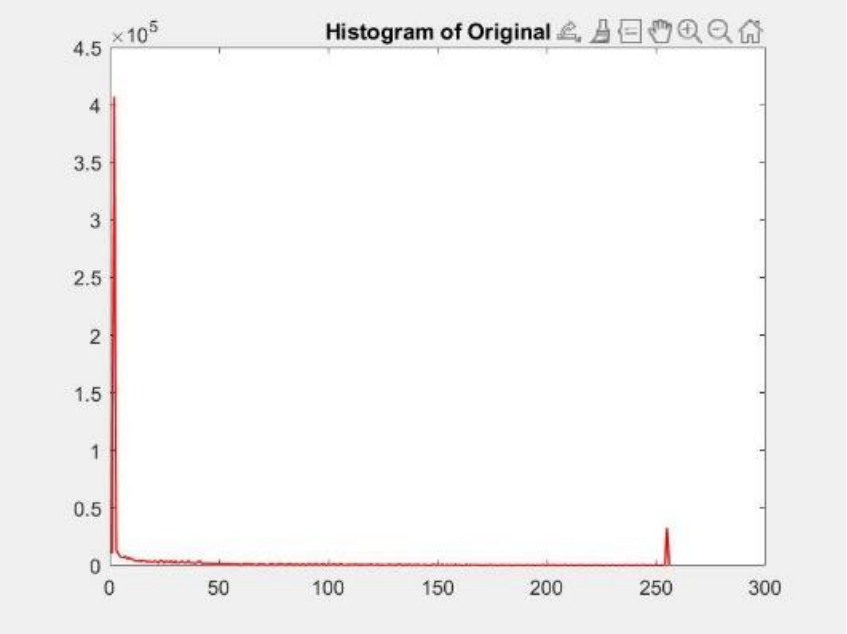
histEqualization.m

```
function [outImage,T] = histEqualization(input_image)
    [M,N] = size(input_image);
    S = zeros(1,256,"single");
    image = zeros(M,N,"uint8");
    [hist_temp] = imageHist(input_image);
    hist_temp = hist_temp/single(M*N);
    for i = 1:256
        if i == 1
            S(i) = hist_temp(i);
        else
            S(i) = S(i-1) + hist_temp(i);
        end
    end
    S = S * 255;
    for i = 1:M
        for j = 1:N
            index = uint8(input_image(i,j));
            image(i,j) = round(S(index+1),0);
        end
    end
    outImage = image;
    T = S;
end
```

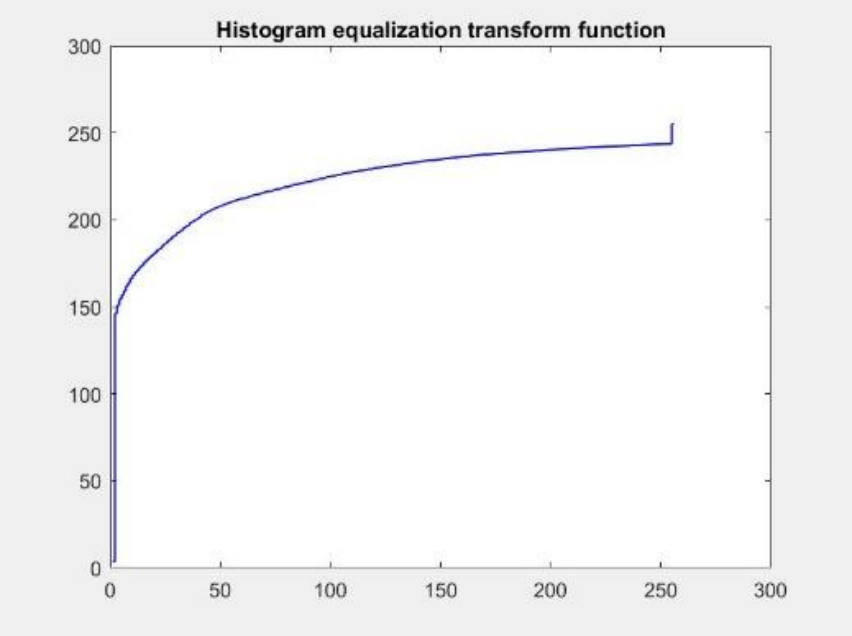
2. 問題與討論：



上圖中分別是原圖以及對他做 intensity 均衡處理後的圖片。

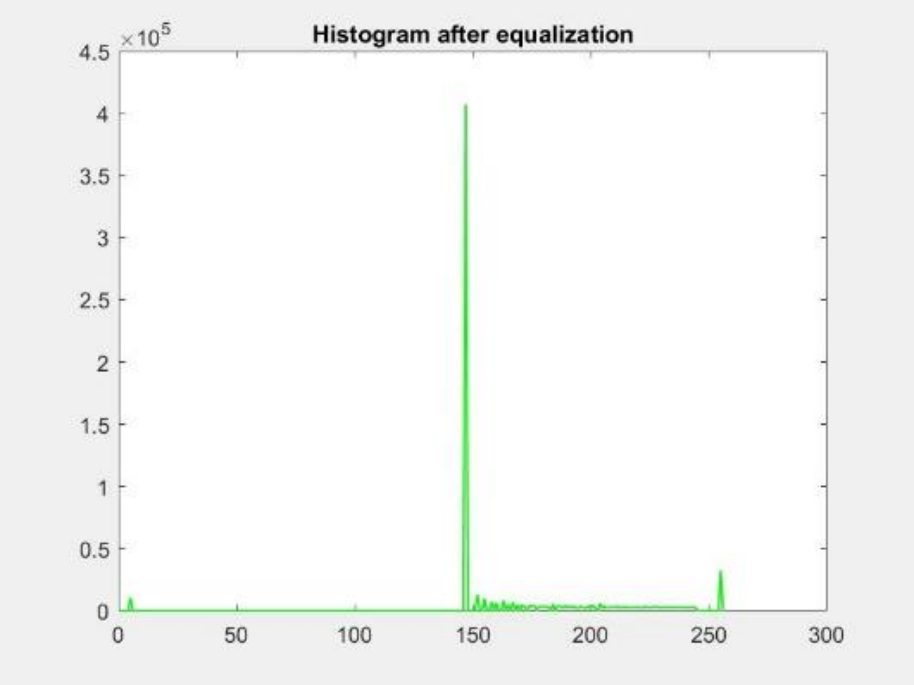


上圖為原圖的 Histogram，基本上所寫的就是不同的 intensity 各自的 pixel 數，可以看到因為原圖偏暗，所以整個的 Histogram dominate by low intensity 的部分。



接下來這張圖可以看到，為了要使整理的亮度一致，所以會把 low

intensity 的部分拉高。



所以最後處理過後的影像，其 Histogram 會集中在中間。

三、Proj03-03 ~ 03-04: Spatial Filtering, Enhancement Using the Laplacian

1. 實作方式：

在 spatialFiter.m 中，會先將原影像作 zero padding，然後將 mask 作 rotate。接下來，因為對 mask 作處理了，所以可以直接點乘，也就是對應的 element 相乘。最後在輸出到對應的位置。

spatialFiltering.m

```
function [output] = spatialFiltering(input, mask)
    [M,N] = size(input);
    temp = zeros(M,N,"uint8");
    image = single(padarray(input,[1,1],0,'both'));
    X = [0,0,1;0,1,0;1,0,0];
    w = single(X*mask*X);
    for i = 2:(M+1)
        for j = 2:(N+1)
            Y = [image(i-1,j),image(i-1,j),image(i-1,j+1);image(i,j-1),image(i,j),image(i,j+1);image(i+1,j-1),image(i+1,j),image(i+1,j+1)];
            temp(i-1,j-1) = sum(sum(w .* Y));
        end
    end
    output = temp;
end
```

LaplacianFilter 基本上就是 call special Filter，只不過處理過的影像還會

在與原圖作疊圖。

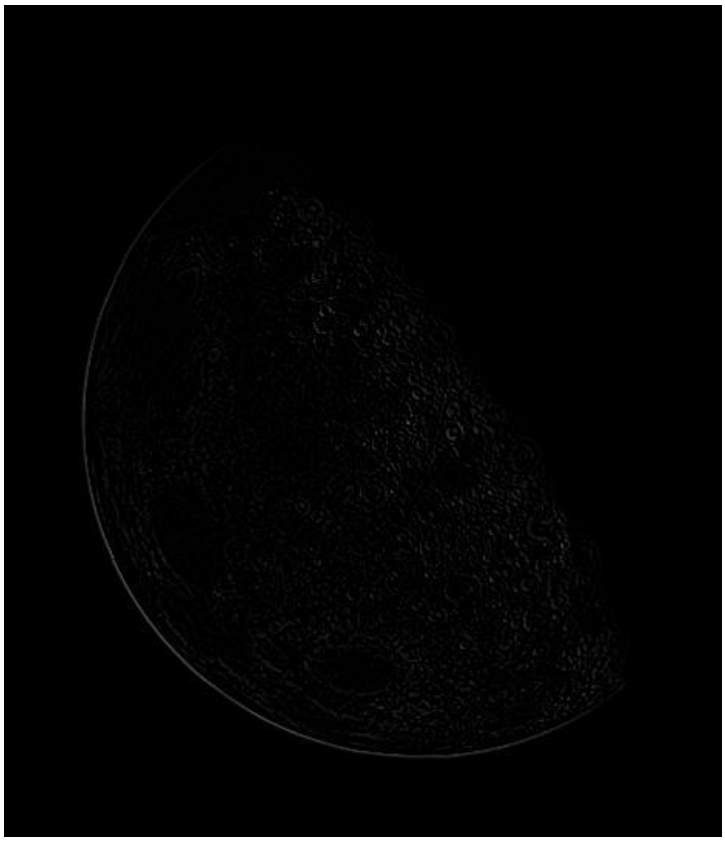
laplacianFiltering.m

```
function [output, scaledLaplacian] = laplacianFiltering(input, laplacianMask, scale)
    [temp] = spatialFiltering(input, laplacianMask);
    scaledLaplacian  = single(scale) * single(temp);
    output = single(input) + scaledLaplacian;
end
```

2. 問題與討論：

下圖是對原圖，使用 laplacianMask filtering 所得到的

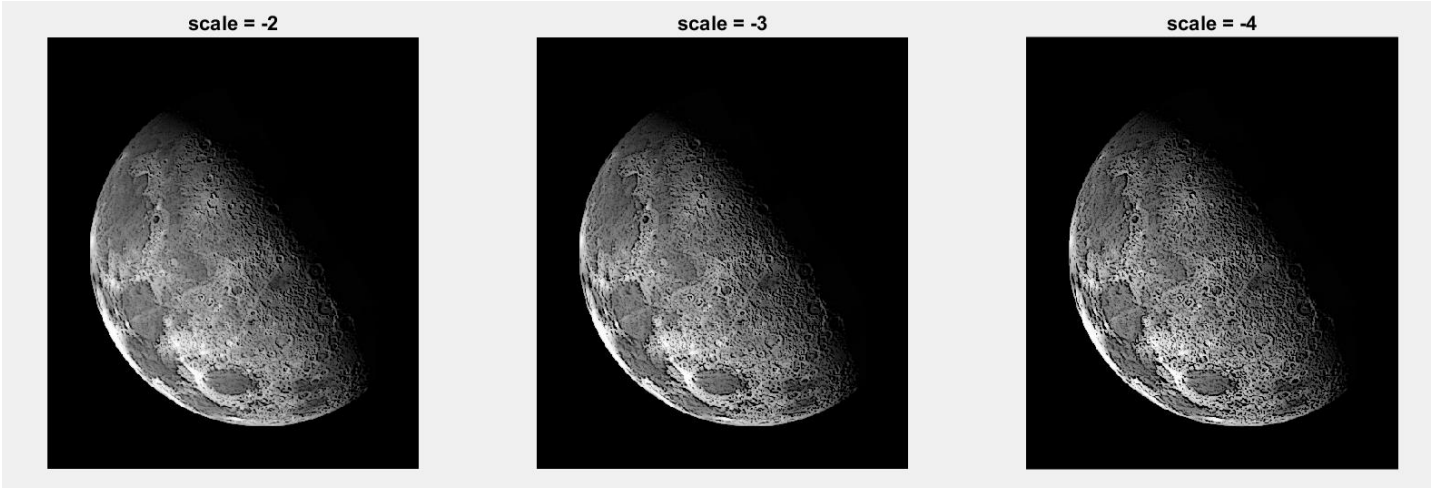
LaplacianImage，基本上，就是 sharp 的部分



然後接下來兩張圖對原圖做 lapacian filtering 的處理，但使用的是不

同的 filter，差別就在於有沒有做邊緣處理。

左圖使用[0,1,0;1,-4,1;0,1,0]，而右圖是使用[1,1,1;1,-8,1;1,1,1]



上圖在使用不同的 scale 時，所產生的不同結果。在這裡可以看到因為不同的 scale，原圖在和 Laplacian 疊圖，Laplacian 被強調的程度就不同。因為 Laplacian Image 中，high intensity 的部分就表示那個位置上，原圖的 intensity 有變化，所以隨著 scale value 的越小就代表 shar 的情形更為明顯。