影像處理 LAB2

學號 x1052078
姓名 陸周濤

3_1 Image Enhancement Using Intensity Transformations

做法說明

首先，確定公式中的常量 c：找到 intensity 矩陣中的最大值（由於 transformation function 是 monotonically increasing），取其 transform 之後的值的 inverse 即能夠保證轉換之後的 intensity 取值範圍 cover [0,1]的常量 c。
利用 matlab 中矩陣計算的方式，可以省去兩個 nested `for` loop，直接將矩陣代入 log 和 power law 公式即可。
另外，對於 power law 的變換公式 controller 腳本中測試了輸入的 γ=0.6,0.5,0.4,0.3,0.28,0.25,0.2 共 7 個 cases，來尋找能夠得到最好（依據主觀判斷）的變換結果的方式。

結果圖片


(a)


(b)


(c)


(d)


(e)


(f)


(g)

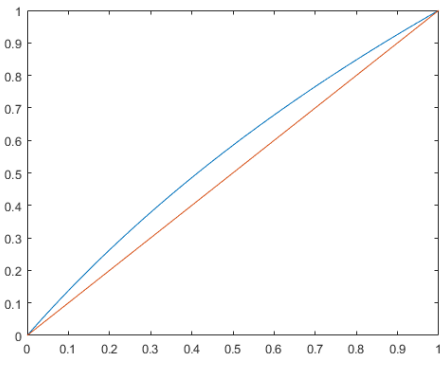

(h)

 (i)

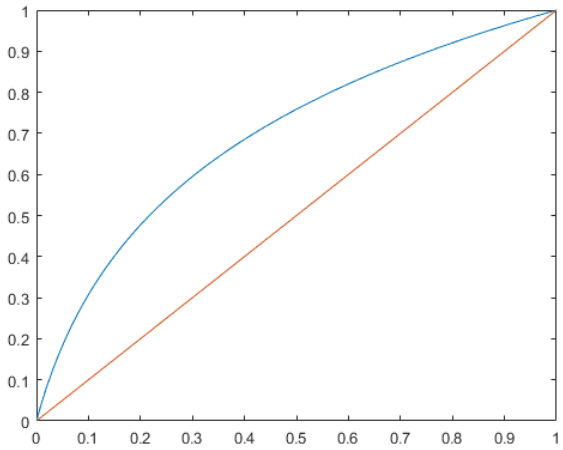
Figure: (a) The origin image, (b) The image transformed by log transformation function, (c) ~ (i) The image transformed by power law with γ=0.6, 0.5, 0.4, 0.3, 0.28, 0.25, 0.2

分析以及討論

1）對數變換公式 $s = c*log(1+r)$ 進行 intensity transformation: 由於 log 函數具有 monotonically increasing 的特點，同時其 fist-derivation 呈下降的趨勢，可以讓原圖中較為 dark 的 component 能夠將 intensity 擴展到一個更大的範圍，從而能夠揭示這些區域的細節。通過將原圖和結果圖像的對比，可以發現整體的亮度值有所提升，也能夠看到一些暗區域細節，但不夠明顯。從整個過程看，原因應該出在變換的公式上，因此我做出了 log 函數的圖像，為了方便比較，同時附上了 identity 的變換曲線，如下。

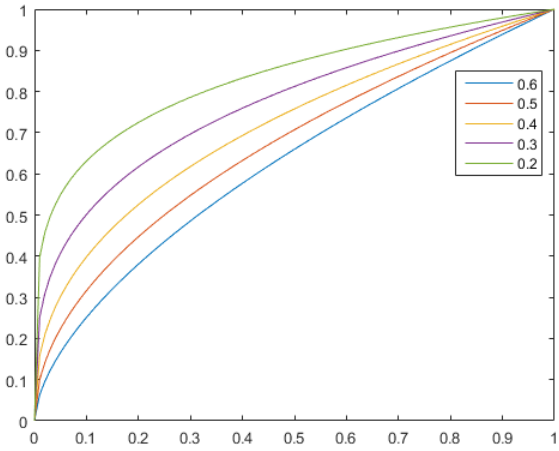


可以發現，log 函數的圖像和 identity 的圖像非常接近，這樣得到改進不大的結果也比較合理了。從變換公式 $s = c*log(1+r)$看，參數 c 是一個 passive constant（scaling the range of intensity），同時常數 1 是為了使得曲線能夠落在(0, 0)點上，因此能夠修改的只有 r 這個 term。嘗試使用 $s = log(1+12*r)/log13$,



從中可以看出，圖像 dark 區域的細節揭示得更加清晰了。

2）用 power law 公式進行變換：根據圖像本身的特點和增強的目標（將暗域的細節顯示出來），在 Gamma 公式中應該選擇的 γ 參數必須是小於 1，所以首先嘗試了 γ=0.6, 0.5, 0.4, 0.3, 0.2 這 5 中情況（對應的結果圖像分別為(c)(d)(e)(f)(i)）。可以看到，隨著冪次的下降，許多暗域的細節得到了增強，但是在 γ 取值小於 0.3 之後，圖像中的脊髓圖樣和背景之間的色差減小，導致整體的效果逐漸下降。為了說明這些圖像之間的差別的原因，再次做出對



應的變換曲線。從圖中可以看出，隨著 γ 取值的減小，能夠將圖像中低亮度範圍擴展到更大的亮度範圍，從而能夠更好地顯示細節；從另一個角度看，在原圖中的暗域和亮域中的像素的色差也會隨著 γ 取值的減小而變得更小，從而會使圖像的對比度減小，因此如果 γ 取值太小，會導致整個圖像被“沖淡”。

3）這兩種方法利用的函數的趨勢特性，能夠將原圖中暗色區域映射到更大的範圍，從而進行 enhancement。但是兩者之間存在一定的差別，應用 power law 的 transformation 能夠更明顯地得到想要的效果，而 log 函數由於函數的特性需要通過調整變元前的參數來調整曲線的整體 slope。兩者都存在的缺點是，

應用的 transformation 公式被其約定的形式所束縛，只能夠是 log 或者 power law，無法根據圖片亮度的不同分佈，進行各自的轉化，需要用下面的 histogram 進行處理。

3_2 Histogram Equalization

做法說明

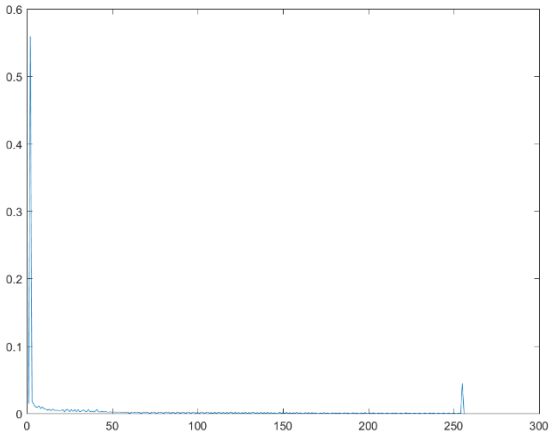
請在這塊說明 imageHist()以及 histEqualization()做法。

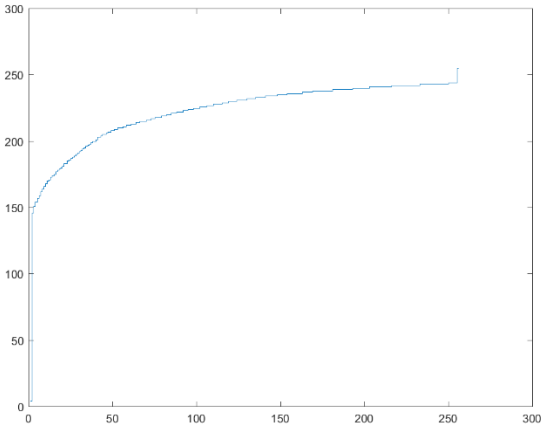
1) `function [ histVector ] = imageHist( input )`: 通過 walk through 整個 image，并記錄不同 intensity 出現的像素點的個數。首先初始化 histVector 為擁有 256 個元素的零向量，然後循環整張圖像，用對應的 intensity+1 作為下標累計該亮度的像素數，最後將這個圖像除以 MXN，即所需的 histogram 向量。所需的 time-complexity = O(M*N+L)
2) `function [ output,T ] = histEqualization( input )`: 首先計算 transformation function，使用的公式是$s_k = \mathrm{T}(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$。在計算時，為了避免重複計算，使用遞推公式$\sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k-1} p_r(r_j) + p_r(r_k)$来计算 CDF。

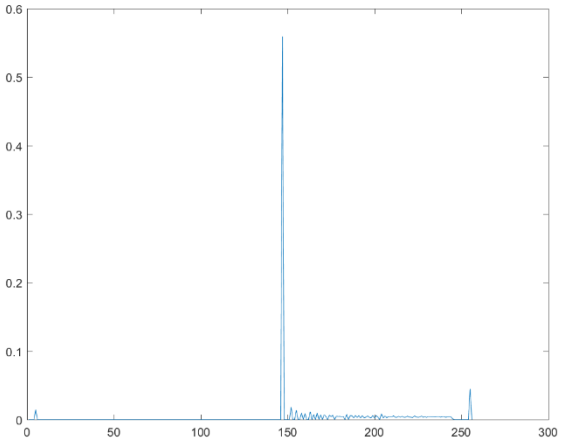
```
sum = 0;
for i = 1:256
    sum = sum + histVector(i);
    I(i) = round(sum*255);
end
```

这样，计算 transformation function 的 time-complexity = O(L). 然后即可利用 matlab 的矩阵计算特点，直接利用 histVector 将每一个像素点的 intensity 进行转换（当然，由于下标和 Intensity 取值起点的不同，需要先对原 image 加 1）。这里的时间复杂度是 O(M*N). 因此进行转换的总 time-complexity = O(M*N+L)

結果圖片







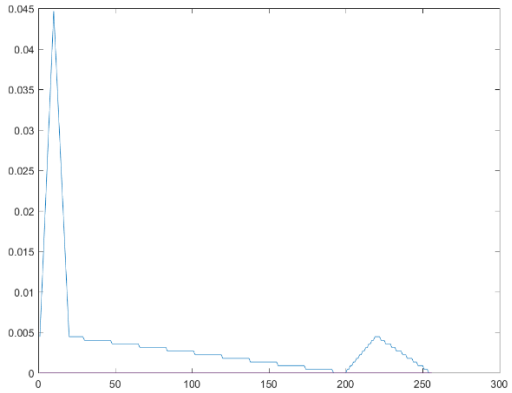
分析以及討論

可以很清楚得看到，原始圖片的最大特點是暗色區域的亮度值和亮色區域的亮度值相差甚大，而且暗色區域佔據了超過圖片的 50%。這一點從原始圖片的 histogram 中也可以看出，圖片有超過 55%的區域都是在暗色區域，而且暗色區域的亮度值幾乎為 0，亮色區域的亮度值都在 255 附近。

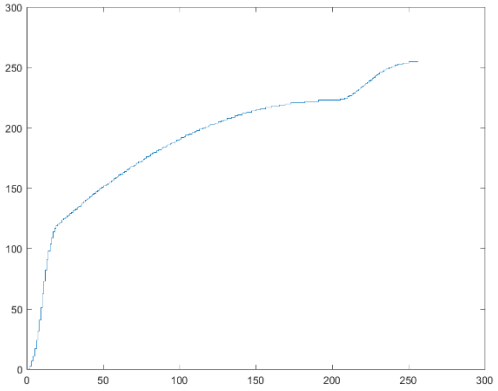
這樣，在計算 transformation function 時，可以預測原始亮度接近 0 的區段，亮度將會被 transformed 到亮度值非常高的區域，從 transformation function 圖上起始驟然上升的變化可以得到印證（原始亮度接近 0，但是對應的亮度值接近為 150）。

因此，最後得到的圖片的起始亮度也會非常高（接近 150），從而導致整個圖片的亮度範圍都集中在較亮的區域。雖然圖片大部分細節被顯示出來了，但是整個的主觀效果並不是特別好，有種亮度被“沖淡”的效果。

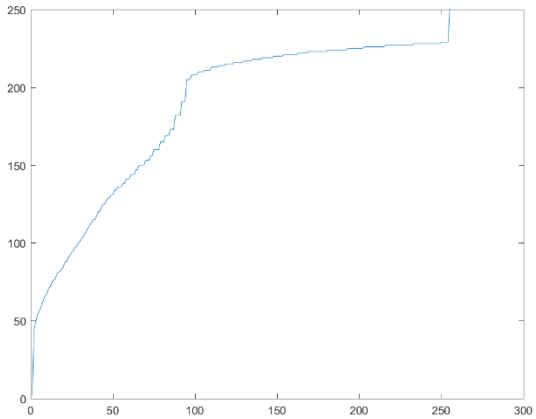
為了解決這個問題，嘗試使用 Histogram Matching 技術。新建一個函數，首先用一連串的 `if-else` 生成指定的 histogram 圖。如下所示：



然後再次跟之前的 histEqualization 中一樣，計算對應的均一化的 transformation function，即 $G(s)$。如下圖：

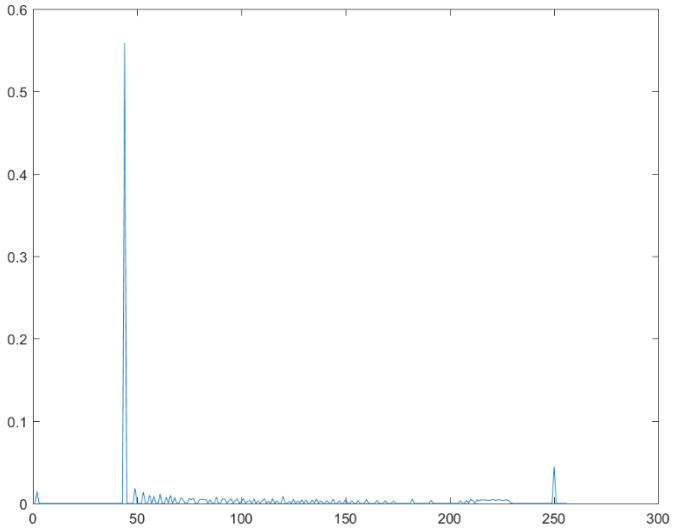


然後計算 $G^{-1}(T(s))$，由於是離散的版本，可能存在找不到逆函數對應點，因此通過一個循環來找到 closest 對應的亮度。得到下面的圖：



，即真正進行轉換的 transformation function。（上述對應的代碼見 `problem2/specMatching.m` 文件）然後作用到原圖之後，得到的結果如下：

 ；對應的 histogra 如下：



可以看到，通過這種方法能夠使得圖像的 histogram 分佈在更大的範圍，最終得到如同 power law 中指定最優係數 γ=0.3 所得到的類似的結果。:)

3_3 Spatial Filtering & 3_4 & Enhancement Using the Laplacian

做法說明

這部分請講解 spatialFiltering()做法，並說明 laplacianFiltering()如何利用 spatialFiltering()實作。

`spatialFiltering()`：該函數輸入的參數是原始的圖片以及要作用在圖片上的 mask，另外函數假設輸入的 mask 的 size 均為 odd（具體 size 沒有指定）。整個 function 用了四個 nested loop，其中外圍的兩個 loop 是用來 traverse 圖像中的每一個點，計算結果圖片中對應的亮度值，內層的兩個 loop 是用來 traverse mask 中的每一個值，以計算坐標點(x, y)對應的亮度值。

由於 loop 的變數（即 index）均為矩陣的下標，因此需要解決的第一個問題是 mask 中

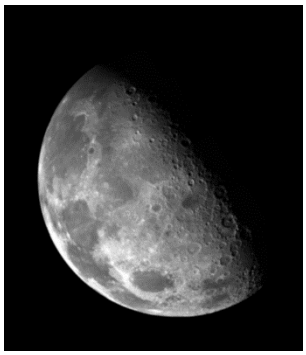
的某一個點應該和原圖像中的哪一個點相乘。假設我現在正在計算坐標(x, y)的亮度值，mask 的 size 是 mxn。現在計算 mask 矩陣中坐標為(i, j)的點對應圖片中的點。由於在公式中用到的 mask 的原點現在落在了((m+1)/2, (n+1)/2)，因此在計算時需要減去(m+1)/2 和(n+1)/2。即現在坐標(i, j)對應的點為(i-(m+1)/2, j-(n+1)/2)。這樣同這個點的 mask 值相乘的是原圖中坐標為(x-i+(m+1)/2, y-j+(n+1)/2)點。

另一個需要解決的問題是邊界值的確定。由於將 mask 的原點選在中心，因此在計算邊界上的像素點的亮度時會缺少計算的因子，我選擇了 mirroring 的方法。具體的做法並不是直接擴增，而是當計算了對應的圖片坐標之後（如上段所述），如果發現那個坐標點落在邊界外，即通過對稱從圖片內取出相應的亮度值。對兩個維度進行兩個邊界的判斷：

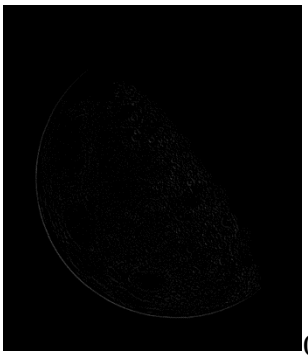
```
if(target_x <= 0)
    target_x = 1-target_x;
else if(target_x > M)
        target_x = M-(target_x-M);
    end
end
if(target_y <= 0)
    target_y = 1-target_y;
else if(target_y > N)
        target_y = N-(target_y-N);
    end
end
```

`laplacianFiltering()`：直接利用 spatialFiltering()函數，傳入指定的 laplacian mask 就可以得到圖片亮度發生改變的二階導數邊界，乘上合適的 scale 值（如果使用 Fig. 3.37(a) 的公式則必須為負數），加到原來的圖片就可以擴大圖像邊界的亮度變化效果，從而突出圖像的邊界以 sharpen 圖像。

結果圖片



(a)



(b)

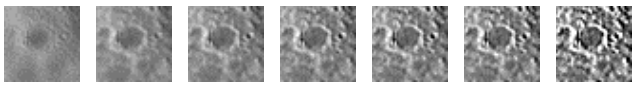

(c)

(d)

【重現課本 FIGURE 3.38】(a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. c) Image sharpened using the mask in Fig. 3.37(a). (d) Image sharpened using the diagonal mask.

分析以及討論

1. 不同的 scale 值的不同的效果比較。結果圖片中展示的是 scale 為 1 的情況，為了比較通過 scale 值對於圖片效果的影響，下面展示了從 scale 值依次為 1, -0.5, -1, -1.5, -2, -2.5, -5 所產生的結果圖片中截取的同一內容：








從上面的圖片中可以看出，當 scale 值為正數的時候，整個圖片會變得“模糊”，這是因為從 laplacian mask 計算得到的是二階導數在圖像亮度上升區域的邊界，其中對應亮度剛上升的邊的值是大於零的，對應亮度上升停止的邊的值是小於零的，因此直接加上去，就會使得邊界連讀上升變得平緩，停止上升的趨勢也變得平緩，這樣邊界就被模糊了。另一方面，隨著 scale 值從-1 變化到-5 圖像的邊界變得越來越清晰，可以從截取的相同內容的圖像中看到，在 scale 值變化過程中，邊界左右的亮度差在逐漸擴大。所展示的坑洞的高亮度邊緣越來越亮，中間暗色區域逐漸變得更暗。這正是當 scale 值變小的時候（絕對值變大），亮度變化區段的邊緣的色差被拉大所造成的。

2．兩種 mask 的效果比較。結果圖片的(c)和(d)分別展示了利用$\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}$和

$\begin{bmatrix}1 & 1 & 1\\1 & -8 & 1\\1 & 1 & 1\end{bmatrix}$兩個 mask 得到的圖片。可以非常明顯地看到，使用 diagonal mask 在相同 scale 的情況下，sharpen 的效果更好。從兩個 mask 的特點看，diagonal 版本考慮了八個方向的亮度變化情況，具有很好的 isotropic 性。這樣八個方向的亮度變化都能得到增強，從而能夠更好的包含圖片的細節，當然如果所用的 scale 過大，就會將一些不需要的 noise 也給 sharpen（非 diagonal 版本也一樣），這樣效果可能會受到影響。

3. 使用不同 size 的 Laplacian mask 的效果比較。為了考察在不同 neighborhood 範圍的條件下，進行 Laplacian filtering 的效果的差別，進行了對於不同 size 的 Laplacian of Gaussian filtering 的測試。LoG 是用於 edge detection 的非常理想的模型，然而圖片的 enhancement 其實也是基於 edge detection 的思想，因此下面的測試採用不同 size 的 LoG mask。

   關於 LoG 這個模型，簡單來說是對圖像進行 Gaissian smoothing 和 Laplacian filtering 的結合，由於在邊界偵測中會將一些不必要的 noise 也加入偵測的範圍，因此首先通過高斯平滑可以消除一些 noise 的影響，然後再通過 laplacian 過濾就能夠得到更加理想的邊界（實作邊界偵測時，還需要進行 zero crossng 的 local variantion 處理，但這裡不做這個處理，原因見後文）。

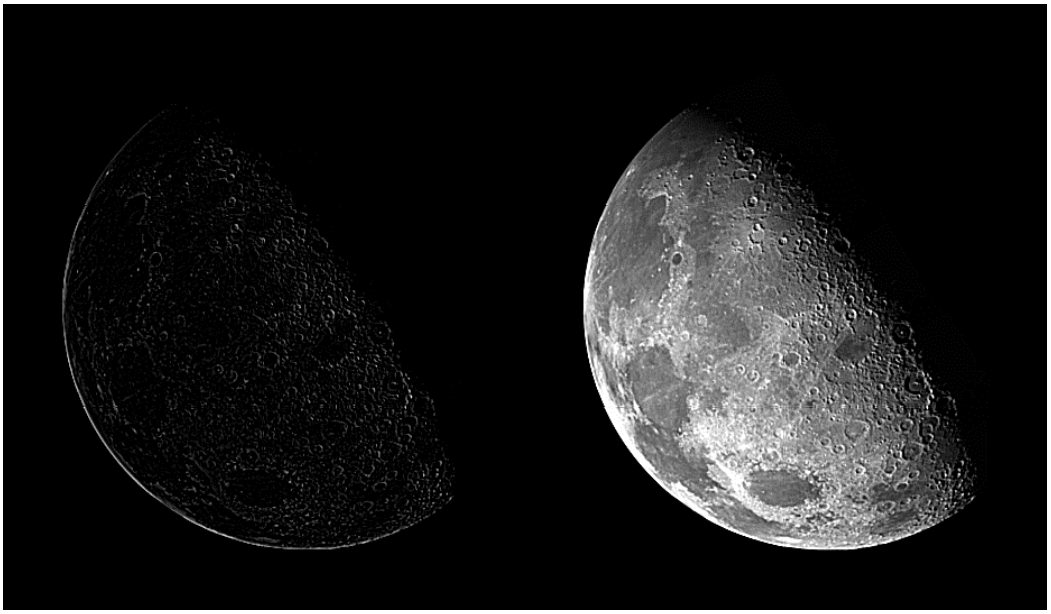
   這樣，藉助 LoG kernel 的公式

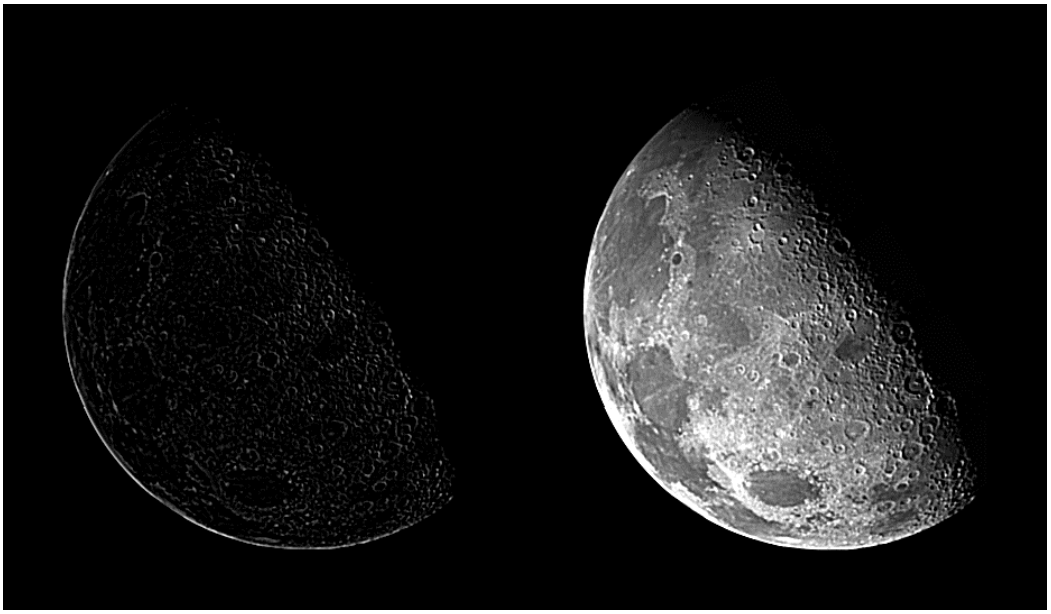
$$\nabla^2 G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

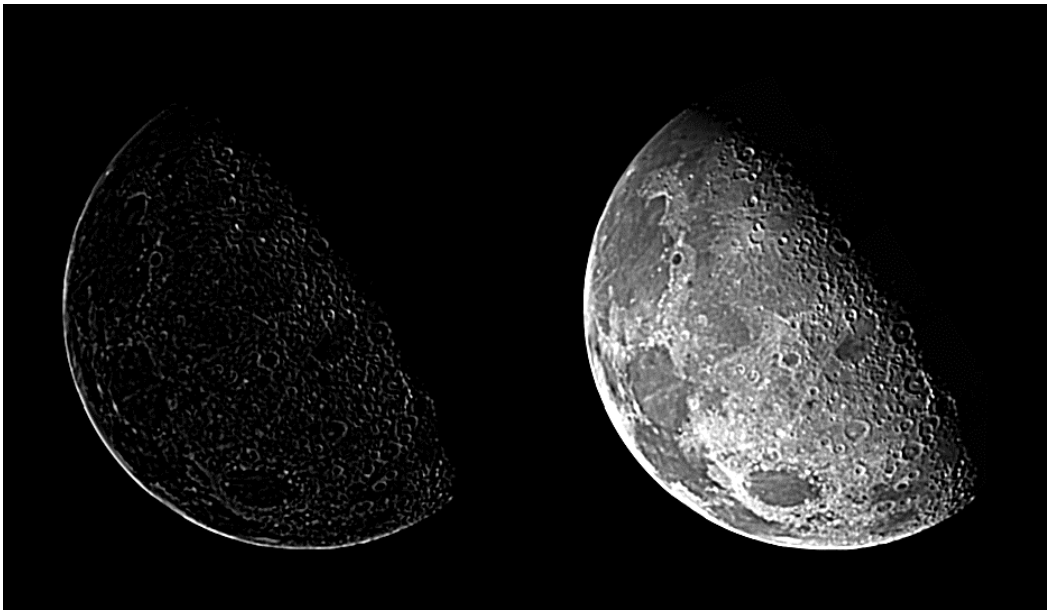
   就可得到任意 size 的 filtering kernel。（下面是 matlab 代碼）

```
function [ mask ] = getLoG( size, sigma)
% get the Laplacian of Gaussian mask of size

center_x = (size+1)/2;
center_y = (size+1)/2;

sum = 0;
for i = 1:size
    for j = 1:size
        mask(i,j) = ((i-center_x).^2+(j-center_y).^2-
                     2*sigma.^2)/(sigma.^4)*exp((-(i-center_x).^2-(j-
                     center_y).^2)/2/sigma.^2);
        sum = sum + mask(i,j);
    end
end
mask = mask-sum/(size*size);
end
```

   保持其他的因素相同，下面是利用上述代碼生成 3*3，5*5，7*7 的 LoG kernel（其中使用的 sigma）之後進行 filtering 的結果圖片：







首先從上面展示的第一個圖(3*3)可以發現同直接用 Laplacian 3*3 的 diagonal 版本相似，能夠較好地 sharpen 圖片。隨著 size 的增大，sharpen 效果確實變明顯了，但是出現的問題是，清晰度有所下降。從【教材 p.738】內容推測，當 size 變大，而高斯部分對於圖像的模糊效果更加明顯，從而通過 laplacian filtering 能夠偵測的就是那些比較大的邊界，而一些細微的邊界將會被平滑化。這個過程對於那些需要去除 noise 的圖像是比較適用的;但是對於這個圖片，為了更加清晰地展示月球表面的地形輪廓，我們預期的希望是捕捉（強化）那些在原圖中沒有展示出來的細微邊界，因此增大 size 的效果不會更佳。也正是因此,我們在處理的時候更加不需要進行 zero crossing 的 variation 辨別。

3_5 Unsharp Masking

做法說明

這部分請講解 unsharpFiltering ()如何利用 spatialFiltering()實作。

`unsharpFiltering ()`：這裡對於函數 `spatialFiltering()` 的呼叫主要是在對圖像進行 blur 的時候，傳入指定的 box mask 就可以得到 blurred image。這便是圖像的 lowpass filtered component，直接從原圖中減去這個部分，就得到了 highpass filtered component。這樣，想要 sharpen 圖像，只要將 scale>0 乘上 highpass filtered component 加回原圖即可使得 highpass 部分被增強，lowpass 部分從而減弱。

結果圖片

(a)

(b)

(c)

(d)

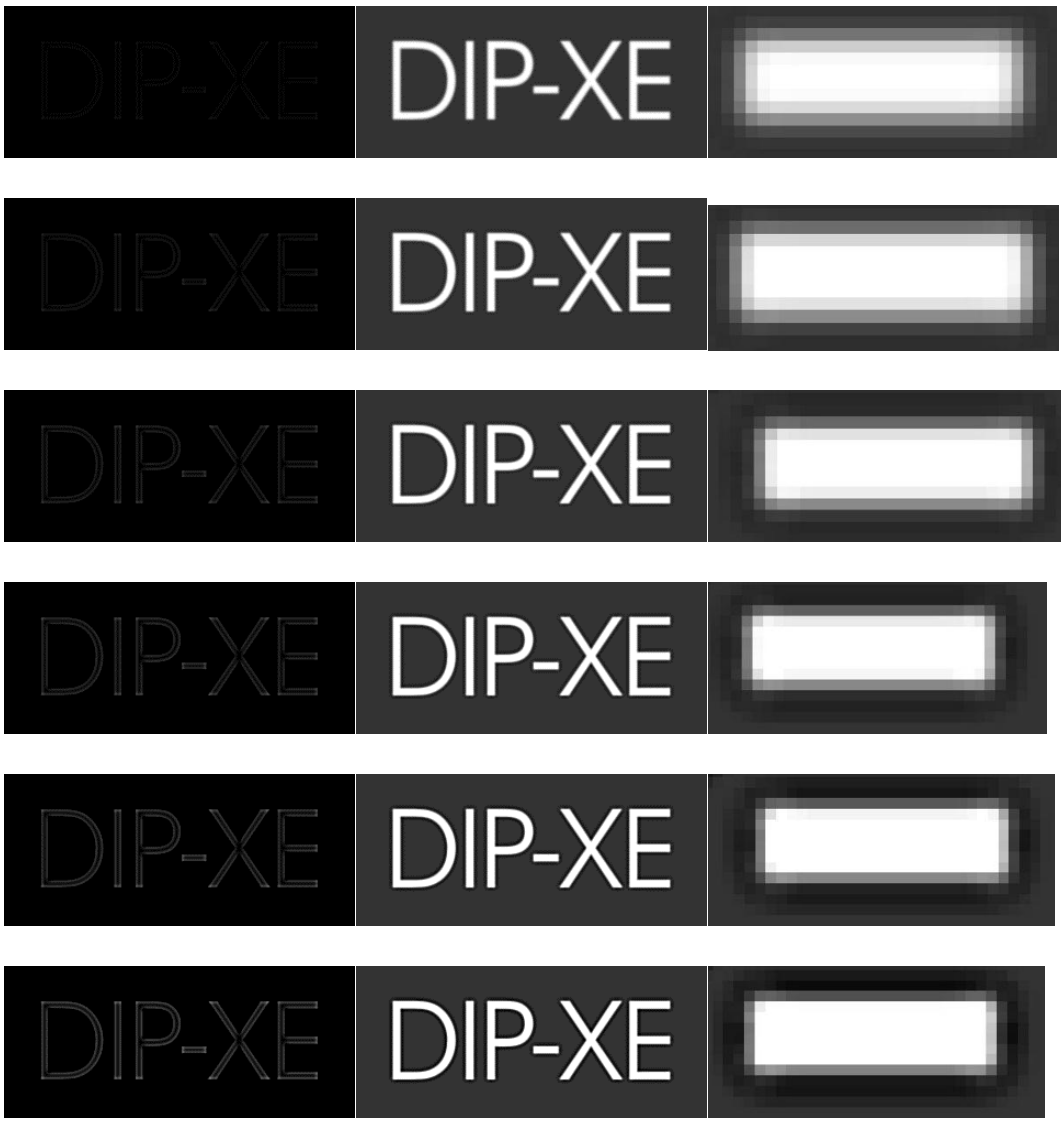
(e)

【重現課本 FIGURE 3.40】 (a) Original image. (b) Result of blurring with a box mask. (c) Unsharpened mask. (d) Result of using unsharpened masking. (e) Result of using highboosting(scale=4.5) filtering.

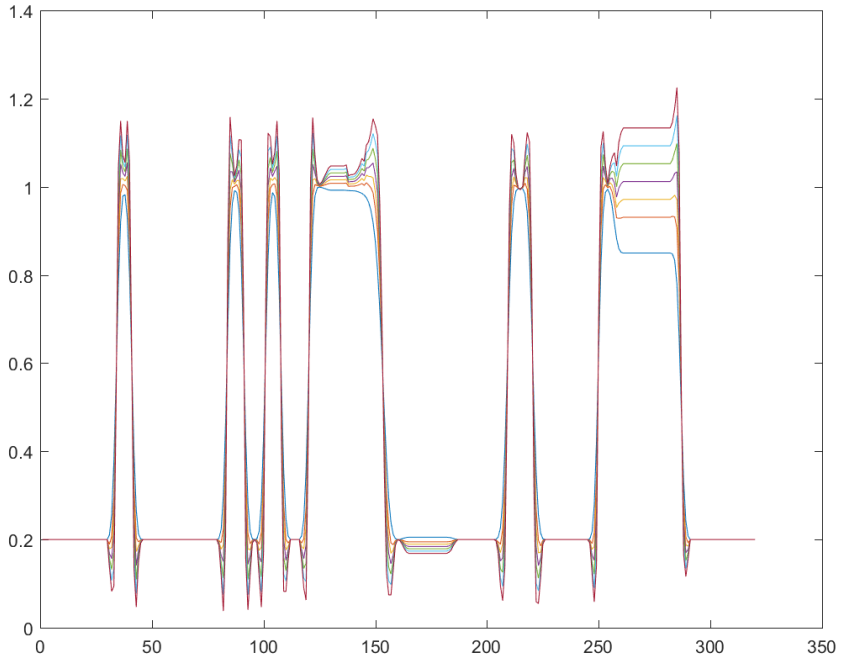
分析以及討論

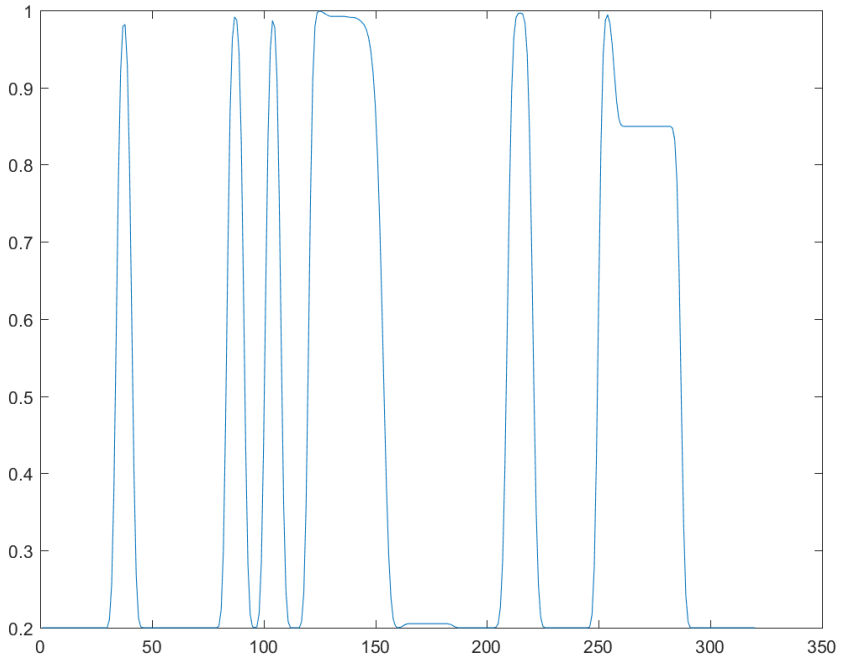
分析不同的 scale（即邊界亮度差放大倍數）對圖片的影響。分析使用不同的 mask 進行 blur 的效果差異，不同的 blur mask 最後會得到不同的 highpass filtered component，這樣加回原圖產生的效果就不一樣了。

1. 不同的 scale 的影響。從上面的 unsharpened 結果和 highboosting 的結果中可以看出，當提高 highpass 部分的比重再重新加回原圖時，能夠得到更加清晰的圖像。為了更加全面地比較不同 scale 值對於最終結果的影響，下面測試了 scale 值分別為-1, 1, 2, 3, 4, 5, 6 的結果圖像，得到如下結果（左列為 scaled highpass component，中間列為最終的結果圖像，右列是截取中間的短橫）：

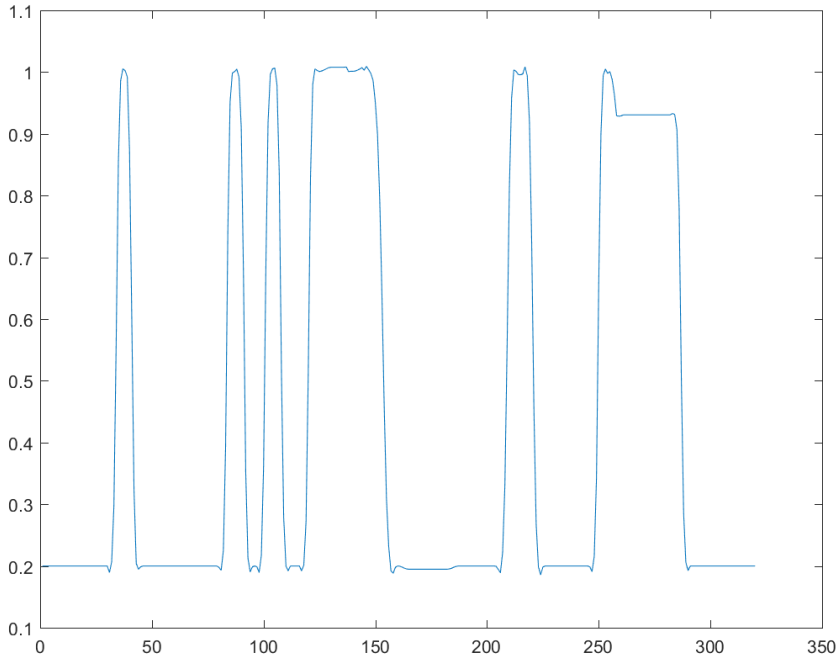


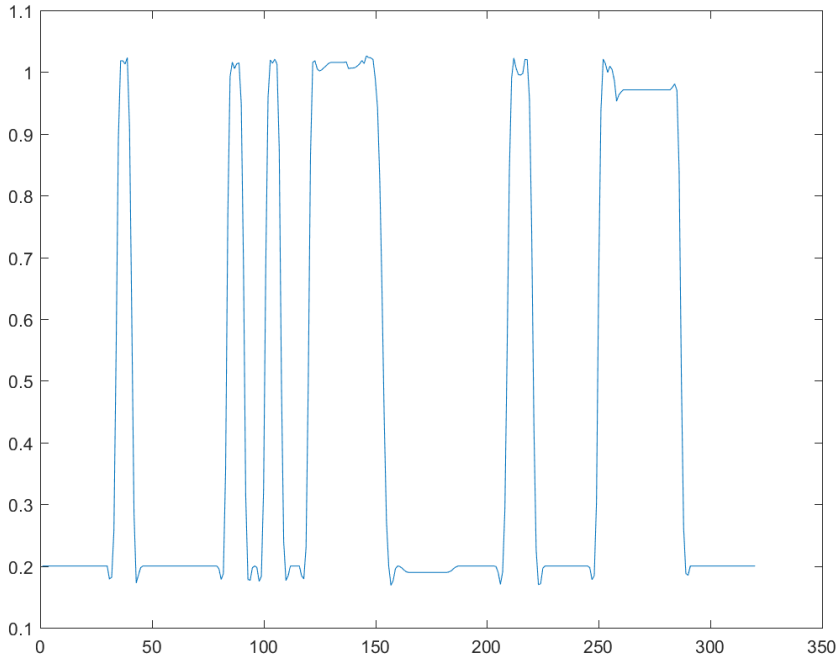
從上面的圖片中，可以看出，1）當 scale 為-1 時，即相當於把原圖的 highpass 部分減去，剩下的就是 lowpass 部分，圖片呈現模糊的狀態，從最右邊的截圖中也可以看出邊緣的亮度變化非常的緩慢，這就是將 lowpass 部分的特點。2）當 scale 值從 1 開始增大到 6 的時候，可以發現圖片的邊緣更加清晰可辨。從左側的 scaled highpass 部分可以看出，邊界處的亮度再被逐漸拉大，從而導致加回原圖時，能夠使結果圖片的邊緣更加清晰。從右側的圖片中可以再次得到印證，邊緣的色差變化不再那麼平緩，而變得更加的迅速；值得注意的是，邊緣的暗色側的亮度被降得非常低。為了能夠更加直觀得展示上述的變化，取圖像的一行像素，繪製對應的亮度變化趨勢。如下：

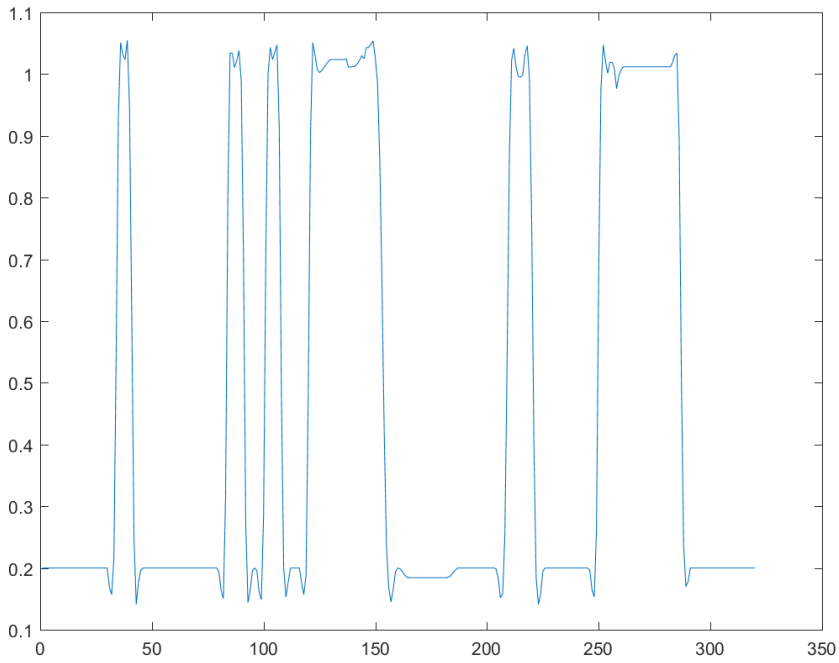

1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250
300
350

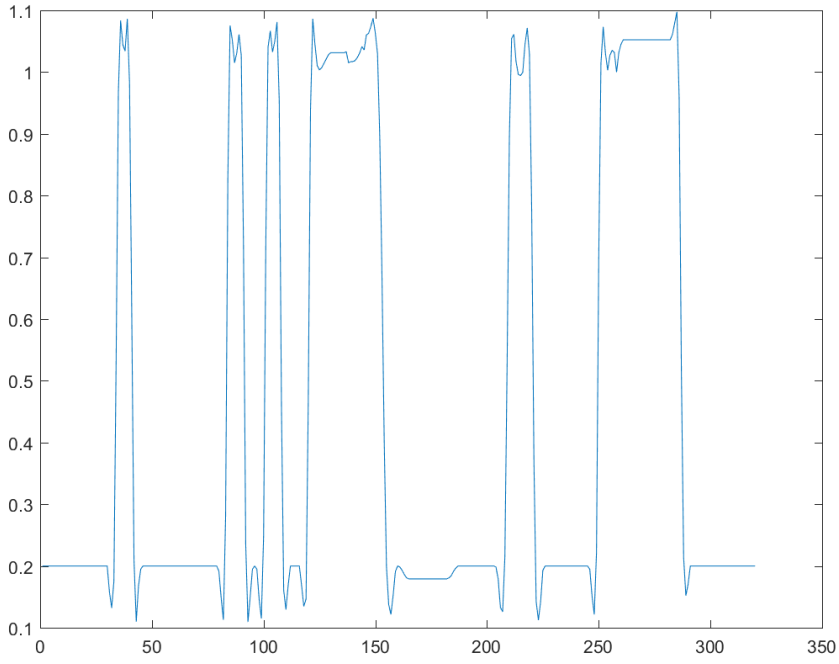

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
50
100
150
200
250
300
350

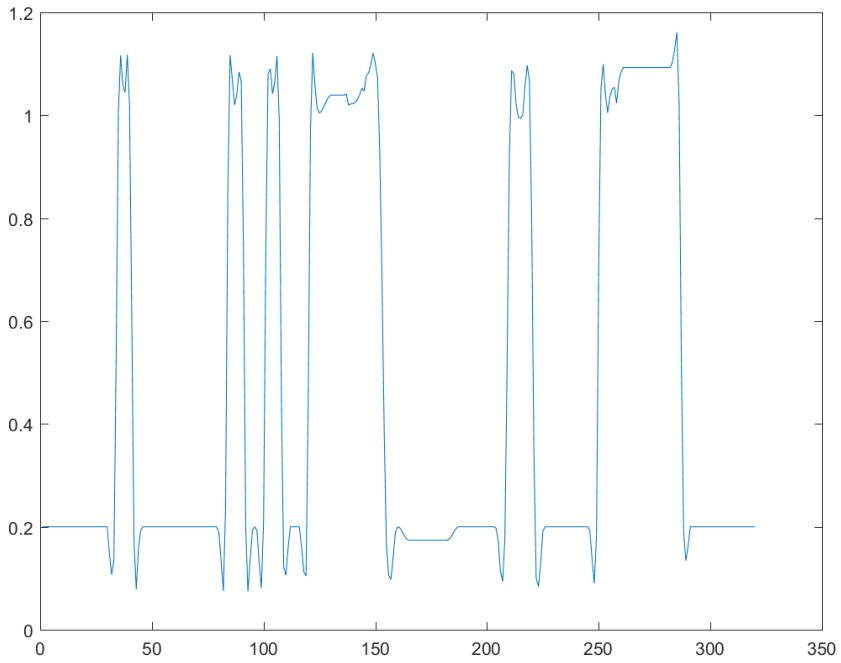
2.

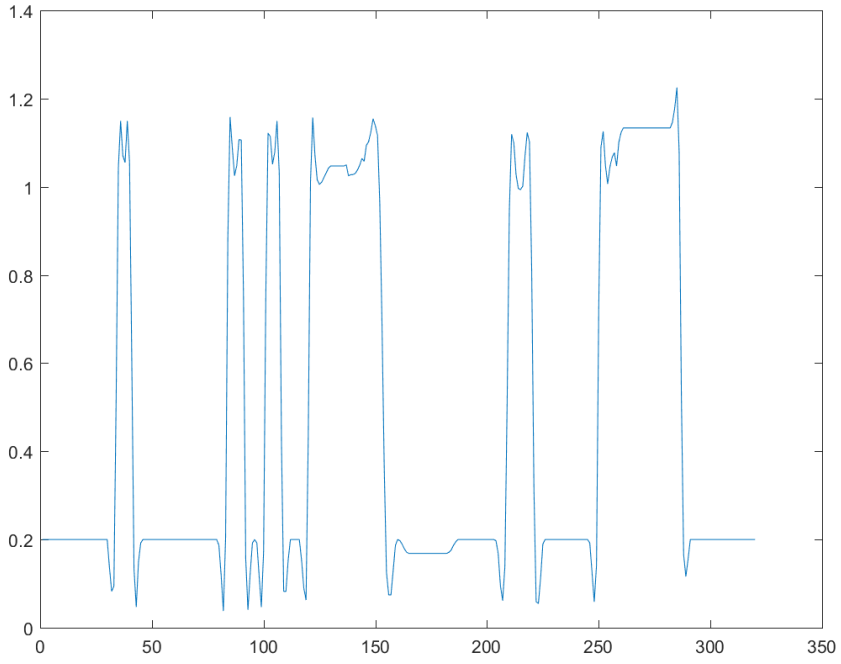

1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
350


1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
350


1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
350


1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
350





從這些亮度變化的趨勢圖中能夠更加清楚的看到隨著 scale 的增大，邊界處的亮度變化的幅度逐漸增大，變化逐漸陡峭化。

3. 不同 blur mask 的影响。由於不同的 blur mask 能夠產生不同的 highpass 和 lowpass 的劃分，為了測試不同的 blur mask 是否會對結果的圖片產生明顯的影響，測試了用 box filtering 和更加關注中心像素點亮度值的 Gaussian average mask。以下是得到的結果：

scale=2

 (gaussian)

 (box)

scale=4

 (Gaussian)

 (box)

從結果的圖片中可以看出在相同 scale 的條件下，Gaussian average mask 的效果比 box average mask 的效果稍差。由於 box average mask

$$\frac{1}{8}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$$

對於 neighborhood 的每一個像素點的權重都是 1，而 Gaussian average mask

$$\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$$

對於 neighborhood 中鄰近的點的權重更大。這樣導致的結果就是求得的 blur image 之間存在差異，Gaussian average 離中心點越近權重越大，這樣能夠更好的維持原來的像素，而受到中衛的像素的亮度值的影響更加小，因此 blurred 的圖片更加接近原圖。接著相減之後得到的結果也就不同了，box average 之後再相減得到的 unsharpened mask 所包含的變化範圍就更大，這樣，當將 scaled unsharpened mask 加回原圖的時候，結果就會更加明顯。