Proj03-01

孫元駿
107061123

Original

(Log) C = 1

(pow) r = 0.1

(pow) r = 0.5

(pow) r = 0.8

(pow) r = 1.5

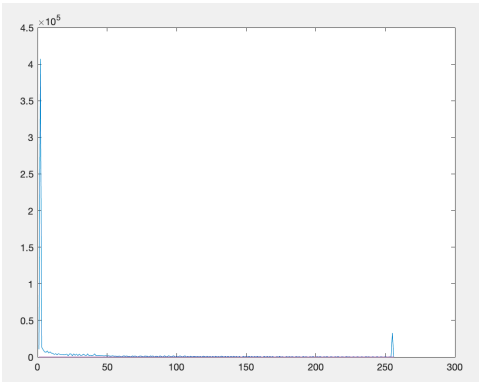
(pow) r = 2

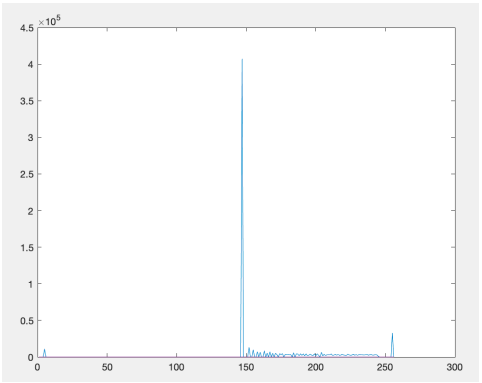
Explanation :
Log transformation: “output = uint8(255 * mat2gray(c * log(1+double(input))));”
Pow transformation: “output= c * double(input).^r;”

Comparison:
Log transformation會提升low-level的值，並且壓縮high-level的值。因此，在圖片中可以看出來，原本一片黑的區塊會增加很多細節，但是原本白色的部分細節會消失。

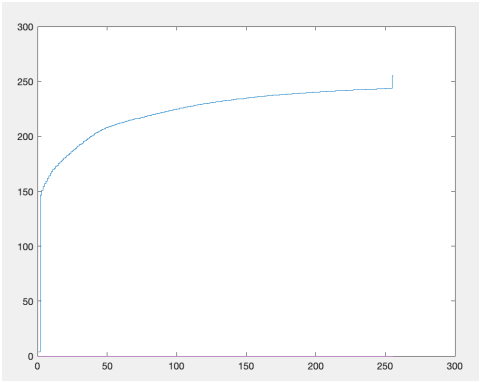
Pow transformation中，如果power > 1，會提升high-level的範圍，並且壓縮low-level，也就是說白色的部分細節會提升，黑色的部分細節會減少。如果power < 1，會提升low-level的範圍，並且壓縮high-level，也就是說黑色的部分細節會提升，白色的部分細節會減少。因此，我將power調整成0.1, 0.5, 0.8, 1.5, 2可以發現符合預期。其中，在這張範例圖中，0.5的表現最好。

Proj03-02





Original **Enhance**



Histogram equalization transformation function

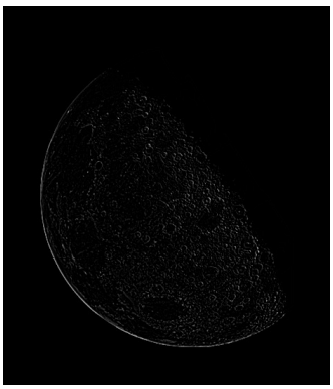
Explanation :
利用for迴圈得到每個值出現的次數，這樣就可以得到出現的機率。再利用 T(i) = prev + 255*(histVector(i))/(m*n); 來計算出transformation function，這樣就可以把原本的圖片對應到新的值。

Comparison:
原始的圖片，大多數的值集中在low-level，黑色的佔比非常大，經過image equalization，可以發現圖片變亮，也比原本的圖片保留更多細節。從histogram可以發現，經過調整後的圖片最多的分佈在150左右，這個也可以從transformation function看出趨勢。

Proj03-03

Original

a



b

c

d

(a)利用Spatial Filtering產生模糊的效果。
(b)計算Scaled Laplacian
(c)用 [0,1,0;1,-4,1;0,1,0]當作mask
(d) 用[1,1,1;1,-8,1;1,1,1]當作mask

Explanation :
在這邊的題目我都使用ignore the boundary，因為這些圖的邊界基本上全黑。我利用if/else區分哪些需要進行filter的計算。在laplacianFiltering中，我將圖片和mask傳入Spatial Filtering，這樣就可以得到要提升的值，再加上原圖本身就是增強後的圖像。

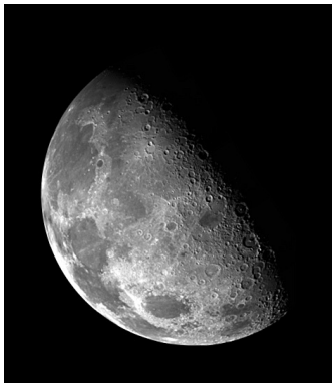
Comparison:
可以發現(a)的效果很符合預期。(b)的部分顯示出經過Laplacian可以強化邊緣，因此在邊緣部分可以發現白色的邊。(c)和(d)分別用了兩種mask並且scale都設定在-1，可以明顯發現(d)在邊緣和細節的強化做得比較好，但是(c)比較自然一些。

Proj03-03

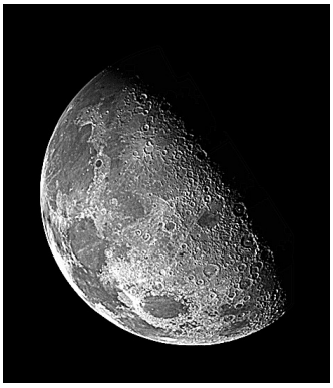
Original

Scale = -0.1

Scale = -0.5



Scale = -1



Scale = -5

可以發現當我把scale調整越小的時候，圖片的邊緣和細節更加明顯。雖然在scale = -5時，圖片有很明顯的細節，但是也讓圖片不太自然。