影像處理 LAB2

103062129 姓名 李國緯

3_1 Image Enhancement Using Intensity Transformations

做法說明

運用 log 或 power 對照片進行 Image enhancement 。
方法為先取定適當的 C 值而後對每個 pixel 進行 log/power 的運算。

結果圖片


Fig0308(a)(fractured_spine)


log_transform_result


pow_transform_result_1


pow_transform_result_2

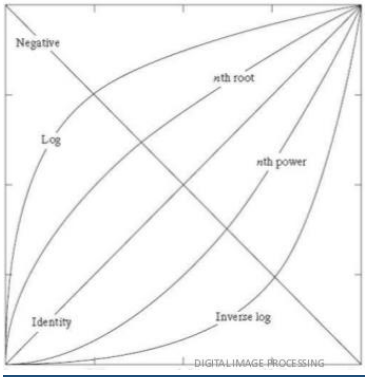

pow_transform_result_3


pow_transform_result_4


pow_transform_result_5

由左至右，由上至下為，(1)原圖 (2) Log Transform (3) Power, r=0.6 (4) Power, r=0.4 (5) Power, r=0.3 (6) Power, r=0.1 (7) Power, r=2.0

分析以及討論



- Log Transform 較傾向於將較低 intensity 的部分提升較多，如上圖所示，對於本身強度就較強的地方來說提昇的相對少，因此可以將原本比較不明顯的 part 顯現出來。相較於原圖而言，多了較多細節，對於整張照片的理解更多，但是整體而言會偏亮，失去原本顏色的深淺的對比。
- Power-Law Transformation 有一個可變化的 Gamma，因此有很多種不同的結果。大致上分為 Gamma（以下簡稱 r）> 1 與 Gamma < 1。當 r > 1 時，會將整體的 level 向下帶，如圖(7)，造成整體顏色偏暗。隨著 r 越大，越多比例的地方 intensity level 趨近 0。當 r < 1 時，越接近 0，越多比例的地方 intensity level 趨近 (L-1)，造成整體顏色偏亮。因此取適當的 r 可以增加對整體圖像的理解，又不至於失真。當 r < 1 時，某些結果會類似於 Log。

3_2 Histogram Equalization

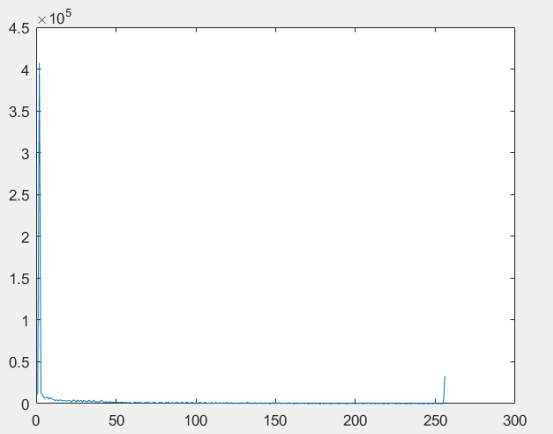
做法說明

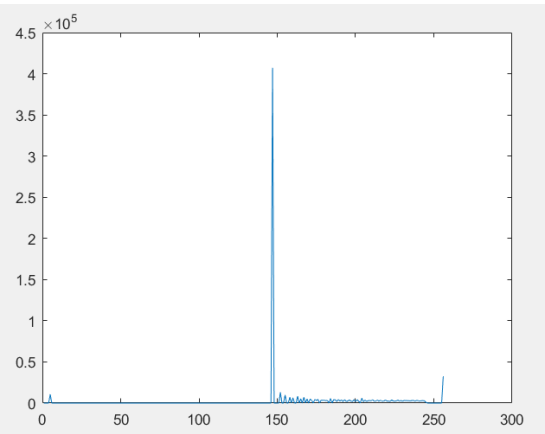
- imageHist(): 首先得知 image 的 Size 後，用兩層 for 迴圈計算。然後建立一個 Vector，在出現過的值的地方做 increment。
    - `Vector(input(I,J)) = Vector(input(I,J)) +1;`
- histEqualization() : 先取得已經計算好的原圖的 Histogram，然後除以(Width * Height)，可得每個顏色在總 pixel 數中的比例，然後依據此比例做累加，可得 Cumulative Distribution Function。若圖是為連續，則用積分，若為 Discrete(本題)，則使用 Sigma。有了分布較平均的 Histogram 後，

  `output(i, j) = T(input(i, j)+1);`

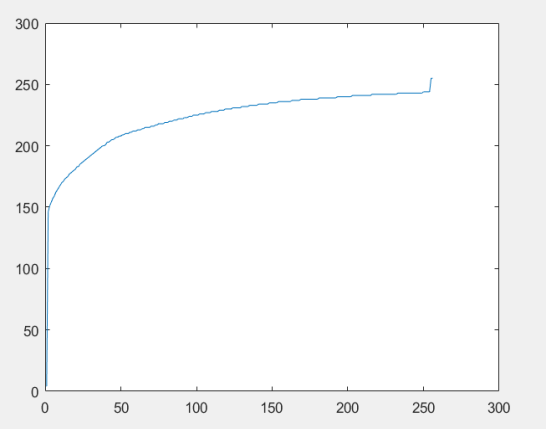
  再依據 CDF 來得知原本的顏色現在應該如何分布。

結果圖片





右圖為 CDF



下左為 enhanced
下右為原圖

分析以及討論

由原圖的 Histogram 我們可以得知，整體的分布較為集中，因此我們運用 Histogram equalization 來使影像的細節或特徵更為凸顯。原來在 Histogram 分佈中比較窄化的影像，低對比度的影像, 透過 Histogram Equalization 的過程, 其亮度灰階的分佈變成均勻分散, 而成為高對比度的影像. 對於影像偏暗的部分提高亮度, 偏亮的部分則降低亮度, 使得細節呈現更為清晰。因為是比例分配，所以不會造成處理完後反而無法分辨物體(顏色相近的混合再一起)。

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

做法說明

- spatialFiltering() : 首先取得 mask 和原圖的大小。然後將 MASK 旋轉 180 度，將其記錄下來。再來將此矩陣 MASK 到一個 PIXEL 上，並以其為中心做 element 的相乘(weight)，並累加所有乘積，得到一個新的濾波影像像素值。

- laplacianFiltering() :

  先做一次 spatial filtering(無 scale)然後得到邊緣偵測，再經過 scale 之後，疊加回原圖就可以得到 sharpen 效果的圖像。

結果圖片

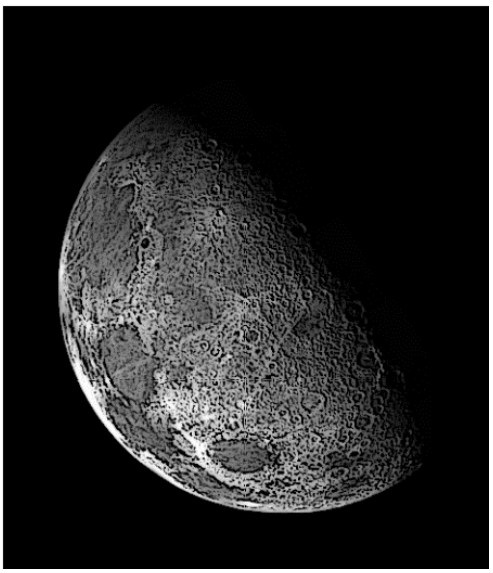
由左至右為 (1.) b (laplaician but no scaled), (2)原圖 , (3) d (mask3.37a) , (4) e (mask3.37b)

分析以及討論

此例中，月球的紋路增強了，及表面的凹凸不平更為明顯。Scale 則決定加強的程度。因為越大的 scale 乘上 spatial 之後會讓 sharpen 的效果增加，然後再疊上原本的圖片，會更明顯。

Laplacian Filter 對於影像中快速變化的區域(包含 edge)具有很大的強化作用，因此常結合 Zero Crossing Detection 作為邊緣偵測的工具。

圖(1)為運用{0,1,0; 1,-4,1; 0,10,}的結果且無 scale。圖(4)為運用{1,1,1; 1,-8,1; 1,1,1} 做的 sharpen， scale 為 -1，若 scale 更大(-1*n)，則輪廓更



明顯。

上圖中 scale 為 -5。

若 scale 為正數，如: 5，則結果如下圖，整個輪廓會近白色。

Scale 越大，sharpen 的感覺會越明顯。

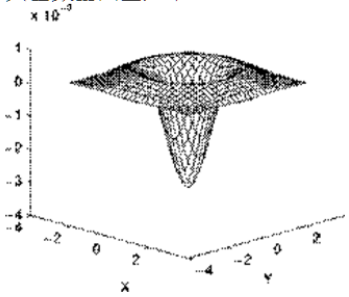
左圖為 5x5 的 mask，scale = -1，所做出來的結果，比起 3x3 mask 有更好的 sharpen。

但正因為也會同時強化雜訊，因此通常會先降低雜訊干擾後，再進行 Laplacian，如 LoG (Laplacian of Gaussian)。

因此所謂的適當，需要看需求，若需要 edge detection，有其適當的 mask 和 scaled，若為 sharpen，則 scale 不宜太大，否則會影響原本的細節。

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

其函數曲面圖如下：



- BONUS: Dealing with boundary

  通常有三種處理邊界的方法

  (1) 直接忽略邊界

  (2) Pad with Zeros

  (3) Mirroring

  當沒有處理邊界時，Kernel window 滑動到最右邊時，超出了邊界，他事實上會取到最左邊的 column，這代表最左邊的 edge 會影響到最右邊的 edge，這會破壞原本的圖片，而不是我們所要的。因此補上 0，比重是 0，因此不會影響到原本的圖片，但是會使 filter 處理不完全(因為某些 mask 上的值乘以 0)。Mirroring 就是複製 edge 的 pixel，在外面多包一層 edge，他一樣可以讓 Kernel 的每一個 element 都對應到一個 pixel，然後最加權平均，然後又不會太突兀，因為是 replicate edge 的 pixel，但實作上較為麻煩。

3_5 Unsharp Masking

做法說明

boxMask 是一個 average mask，此處用[1 1 1; 1 1 1; 1 1 1]/9，和 spatial filtering 來得到 blurred。再用 input- blurred 得到 gmask，最後 output 出 input – gmask*scale。

Mask = [1,1,1;

1,1,1;

1,1,1]/9

結果圖片

原圖

Blurred

Unsharp Mask

Result of using unsharp mask

分析以及討論

Mask size = 5

mask size = 5 時，效果比 Mask size = 3 來的好。在此範例中，unsharp 銳化了照片，強調了字母的邊緣。然而 unsharp filter 雖然可以銳化照片，但是在某些照片的某些地方的呈現上會比較不精確，若 mask 的 size 過大，在此例中，會讓背景更淡，因此，選取適當的 mask size 和 scale 才能有最棒的結果。