**高等影像處理**

**作業#(1)**

姓名：梁仔弘

|  |
| --- |
| 1.1.b |
| 此題為螢幕截圖 |
| *1\_1\_b.raw* |
| Discussion |
| 為了讓 ”lena256.raw” 能夠正常在XnView中顯示，必須從Options -> General -> Read/Write -> Read標籤 -> RAW 裡面調整參數，高度跟寬度必須調整到256x256，Channel Order 為 Interleaved（不過似乎對於灰階影像來說，此項並不影響。）， Channel Type 調整為Greyscale，之後才可以正常顯示這張圖片。 |

|  |
| --- |
| 1.2.b |
| **第一小題** |
| Intensity at 123rd row and 234th column is **157** |
| **第二小題** |
| The 5487th element is **119** |

|  |
| --- |
| 1.2.c |
| 此題為螢幕截圖 |
| *lena256\_out.raw* |

|  |
| --- |
| 1.2.d |
| Figure |
| *1\_2\_d.raw* |
| Discussion |
| 這一題我認為是最麻煩的一題，我把它分成三個步驟：   1. 分割四塊 2. 旋轉左上角與右下角的區塊 3. 拼接(concatenate)   而我在寫的時候，只寫了旋轉+90度的程式，因為旋轉-90度其實就是旋轉270度，因此我將左上角圖塊旋轉了三次以達到同樣的效果。  我覺得我的方法有點太過複雜了，或許存在一種更快速便捷的方式，比如說掃描一整張圖的同時就把左上角和右下角旋轉完成。 |

|  |
| --- |
| 1.2.e |
| Figure |
| *1\_2\_e.raw* |

|  |
| --- |
| 1.2.f |
| Figure |
| *1\_2\_f.raw* |
| Discussion |
| 這個題目其實並不難，留心觀察就能發現他其實是由**四次鏡射**完成的圖。也就是將原本的向右（或向下）鏡射兩次，再向下（或向右）鏡射兩次即可。而其中在向右鏡射的時候，必須特別注意新圖座標與舊圖座標間的轉換。  舉例來說，第一次向右鏡射的時候，新圖片大小為256×512，而新圖在座標(1, 256)時，對應的值應該是原圖的(1, 255)。可是，當轉換成一維的時候，新圖的座標在512×1+256=768，而原圖對應的座標為256×1+255=511。但我一開始卻也將新圖前面的256乘上512，所以怎麼做都不對，後來才發現並解決了問題。至於我採取的方式則是向右鏡射兩次，再向下鏡射兩次完成。 |

|  |  |
| --- | --- |
| 1.3 | |
| Figures | |
| *1\_3\_a.raw* | *1\_3\_b.raw* |
|  |  |
| **Discussion (1.3.c)** | |
| 由以上兩張圖可以很明顯發現出現overflow及underflow的情況出現。我們的圖片是以unsigned char的資料型態處理，而這個資料型態的值只會介於[0, 255]。  那麼，當我們再從255往上加時(overflow)，他就會跳回0，從0開始繼續計算，這就是為什麼圖1\_3\_a有些許部分出現黑色面積。  至於underflow，則是與overflow相反，從0往下再扣就會跳回255，而這就造成了圖1\_3\_b出現白色噪點以及由overflow造成的黑色噪點，因為亮度值增加的數字是完全隨機的，所以每個像素點都有機會overflow或者是underflow。 | |

|  |
| --- |
| 2 |
| Figure |
| *2.png* |
| Discussion |
| 其實有OpenCV就很方便，只要利用OpenCV創造一些點，再將點套進去他的函式裡面，OpenCV就會自動幫你處理好。這題目最難的部分應該就是找到像素點的座標以便我們框出威利（但其實也不難），不過使用XnView就能輕鬆取得座標值了。而印學號上去也是一樣的道理，有專門的函式可以讓我們使用，非常的無腦。 |