

**Advanced Digital Image Process**

HOMEWORK 2

Lab : VPILab

Advisor : Cheng-Ming Huang(黃正民)

Student : Yu Cho(卓諭)

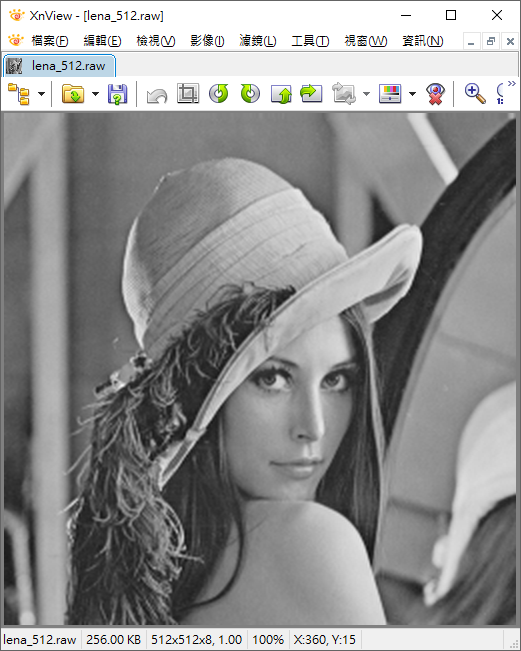
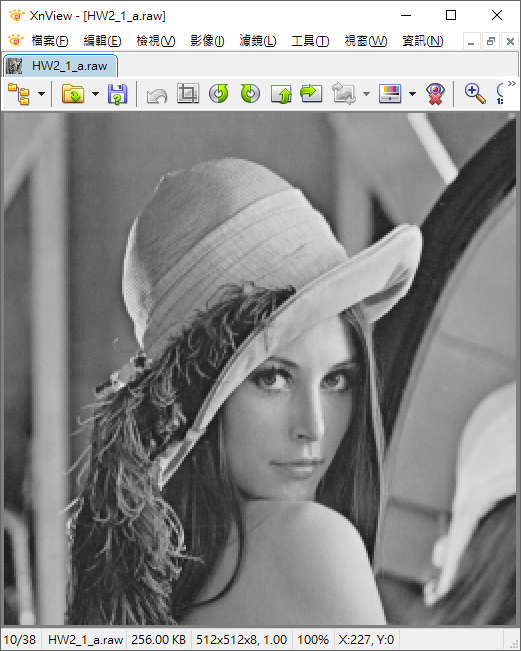
Student ID : 106318025

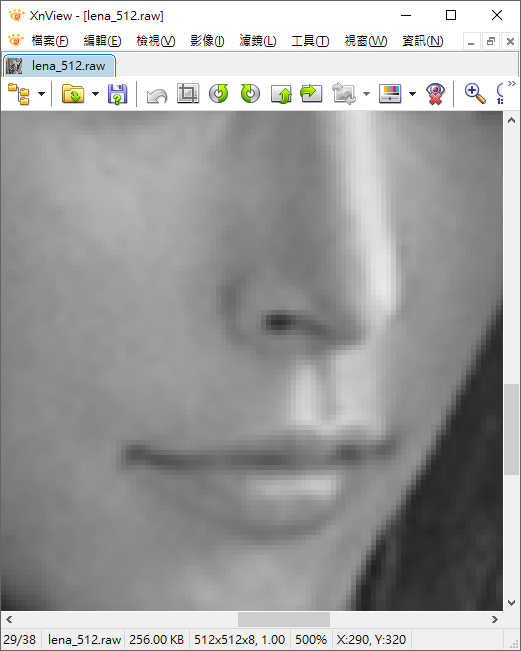
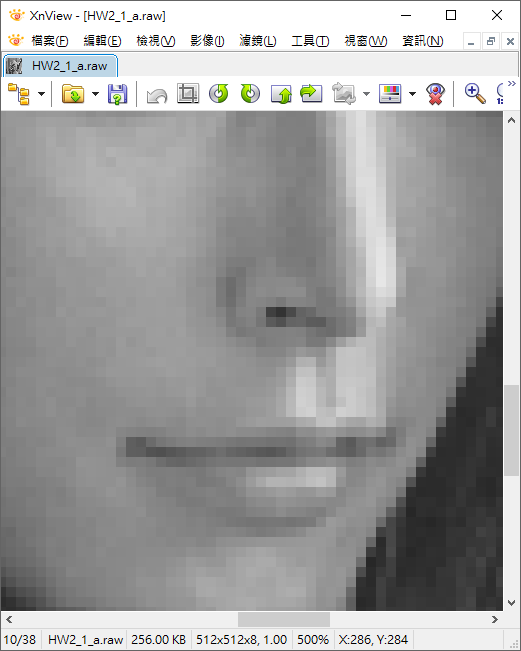
Data : 2017.10.16

1. Zooming and Shrinking(C/C++)

Using C/C++ to perform the following tests on lena\_256.raw

1. Zooming the image with ratio 2:1 using row-column replication. Compare the output with lena\_512.raw.
2. Figure



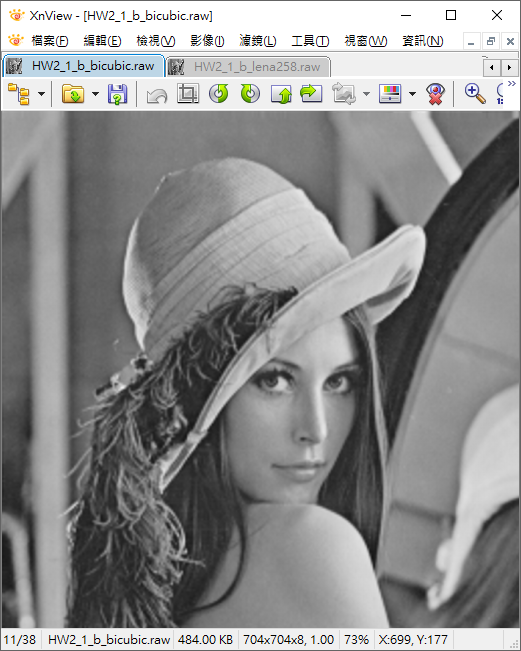
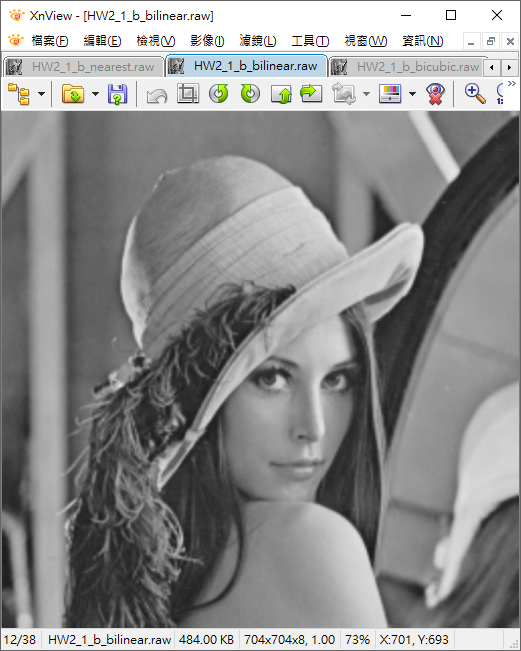


1. Discussion

左圖為使用row-column replication 所放大的影像，會與512\*512原圖有所差異是因為此方法是將原圖的1個pixel複製4個到新圖上，因此細緻程度不夠，而產生鋸齒狀。

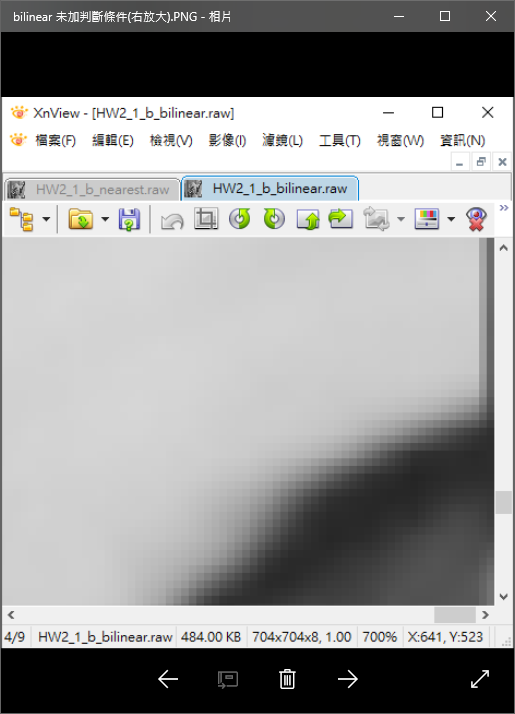
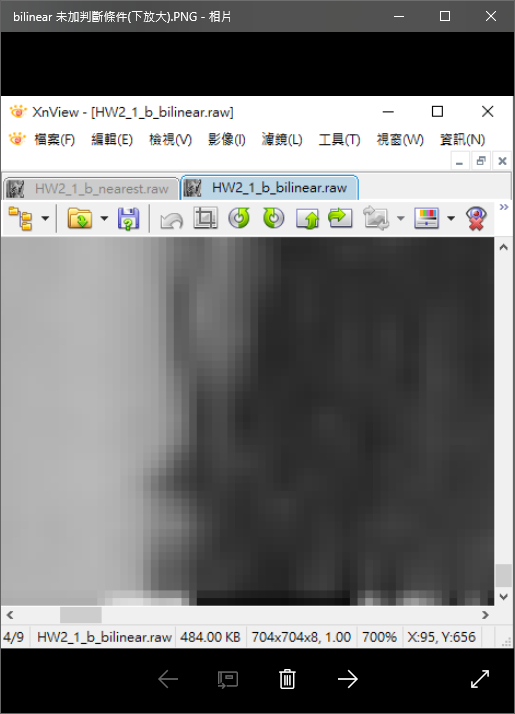
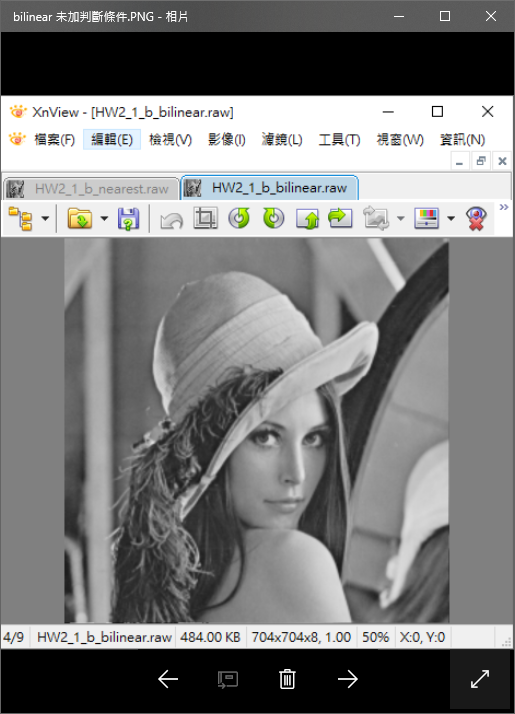
1. Zooming the image to size of 704\*704(i.e. ↑2.75).Compare the results of nearest neighbor, bilinear, bicubic interpolation approaches. How do you process your boundary pixels in your program?
2. Figure

(nearest neighbor interpolation) (bilinear interpolation) (bicubic interpolation)



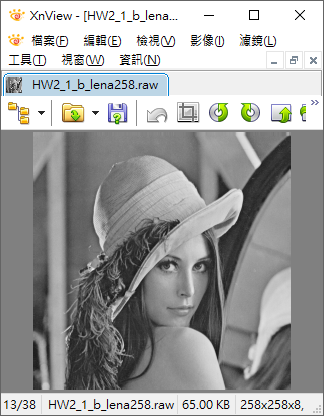
1. Discussion

(bilinear interpolation)



這是在用bilinear 處理影像時會發生的邊界問題，可發現整張圖片下排及右排都會發生錯誤，判斷是因為bilinear方法是將原圖單一像素對應至要處理影像的四個像素點，而基準點在左上方的情況下往右跟下個擴一個像素值去做線性內插，這樣的情況當基準點跑到右255或下255時，會出現問題，因為抓不到更右或更下的像素，導致出錯，我的解決方法是當基準點到邊界時，讓往右及往下抓的像素值等於基準點像素值，即可解決問題。

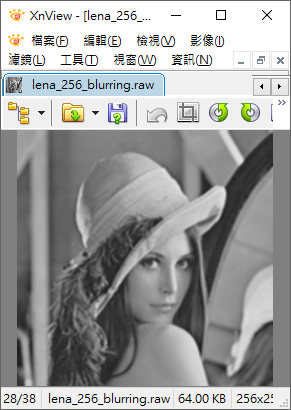
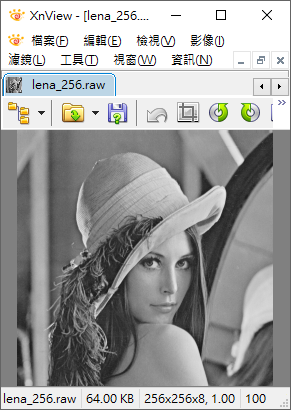
(bicubic interpolation)

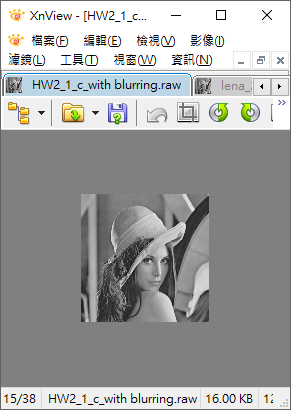
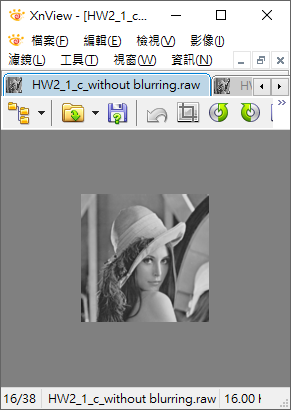


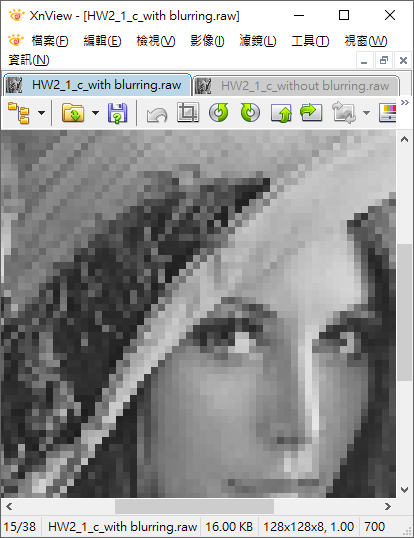
使用bicubic會遇到四邊邊界問題，原因是因為相較於bilineear取4個像素點做內插，bicubic是取了周圍兩層也就是16個像素值做雙立方內插，我解決此邊界問題的方法為直接把原圖外擴出一個像素，256\*256變成258\*258，外框像素值與邊界像素值相等，再去使用16個像素值的雙立方內插。

1. Shrinking the image with ratio 1:2 row-column deletion. Check your result with or without blurring(using XnView) your input image before shrinking.
2. Figure

使用XnView模糊化lena\_256.raw，比較縮小結果。





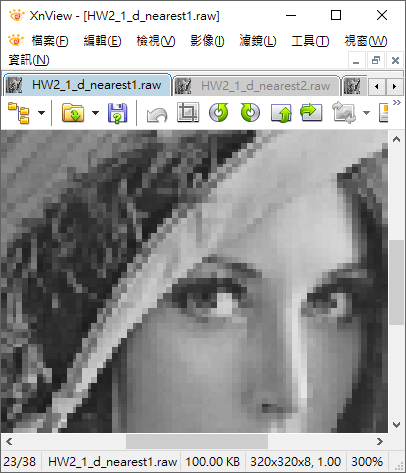


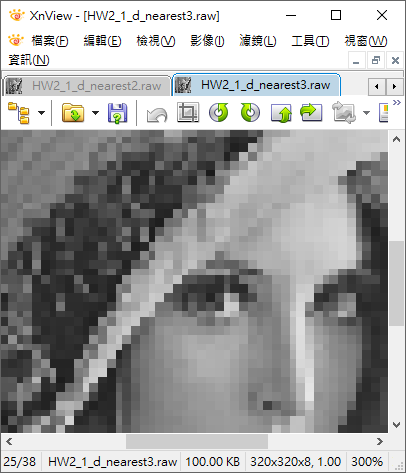
1. Discussion

由結果圖可以得知，先做模糊化後再縮小會使處理後的影像對比度更為強烈，可以表現出更多細節。

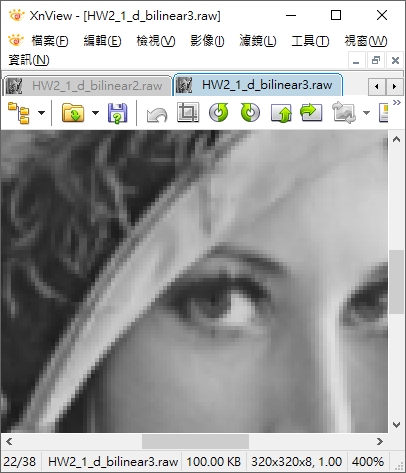
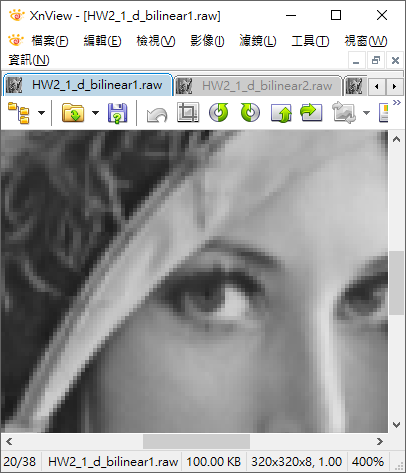
1. Compare ↑2.75 ↓2.2 and ↓2.2 ↑2.75 and ↑1.25 with nearest neighbor, bilinear, bicubic interpolation approaches. Please discuss the difference in execution time, image quality and any other issues. Explain the difference with reason.
2. Figure(依序為↑2.75 ↓2.2、↓2.2 ↑2.75、↑1.25)

(nearest neighbor)

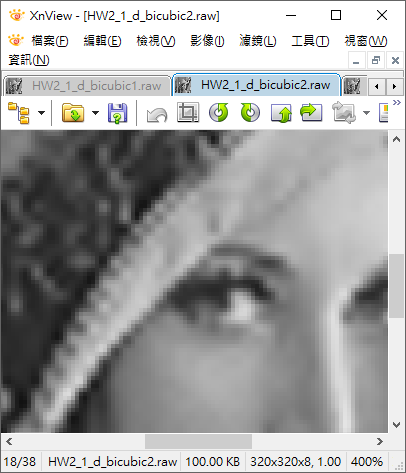
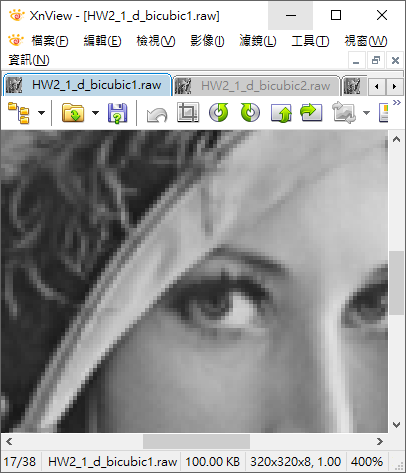




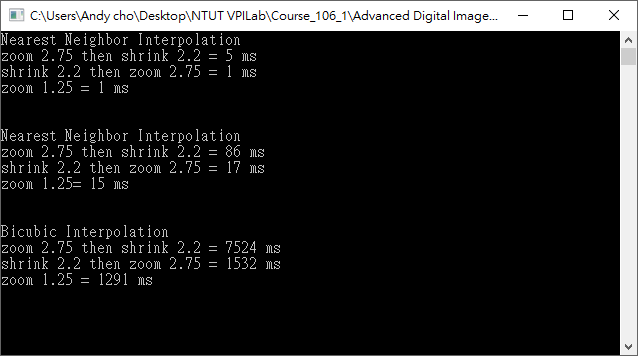
**(**bilinear**)**



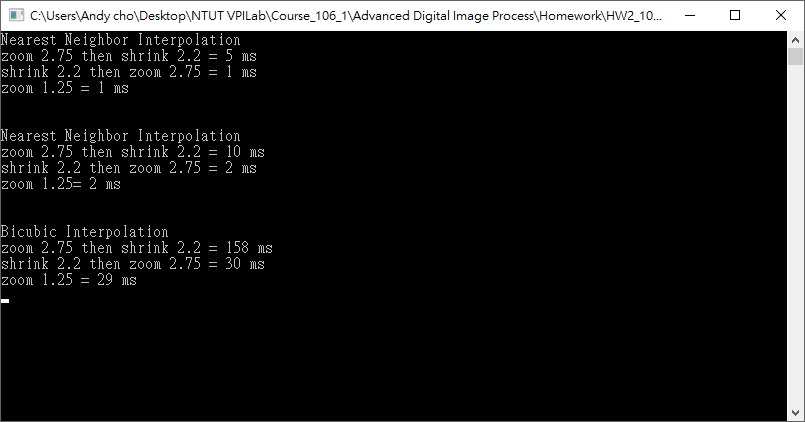
**(**bicubic**)**



(方案組態：Debug Win32)



(方案組態：Release Win32)



1. Discussion

在處理先放大再縮小會比先縮小再放大細節表現更好，原因是因為當先縮小後，因為縮喜影像的細節度不佳，而能夠放大的細節就相對變少，就會造成較為模糊的結果。而要注意的是先縮小再放大會出現溢位，故必須在內插的迴圈外加上一個判斷式，將超出8bits像素值強制等於255或0。

執行時間方面，

最近鄰法因處理像素點少，沒有過多的變數傳值及數學運算故能把時間壓在5ms內。

雙線性內插法，要求取出該像素點位置的周圍四個像素做雙線性內插，故資料傳輸及數學運算較最近鄰法大，執行時間相對長。

雙立方內插髮我使用了三個副程式及一個主程式，互相傳值做運算在記憶體的使用方面損失了成本，並且資料量及運算式最為龐大，才會出現最慢要7.5s才能計算完整張圖。

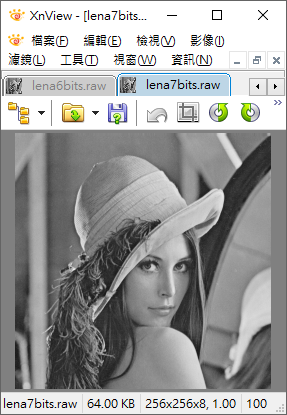
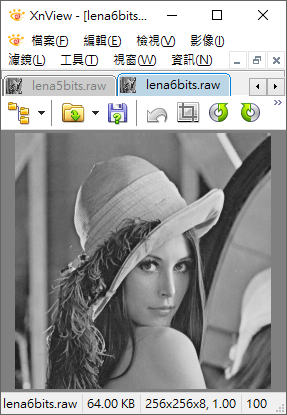
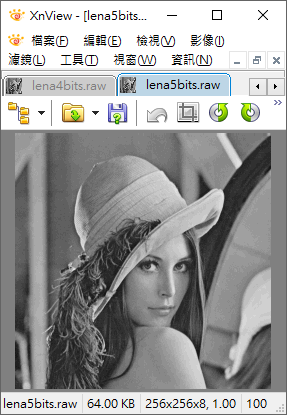
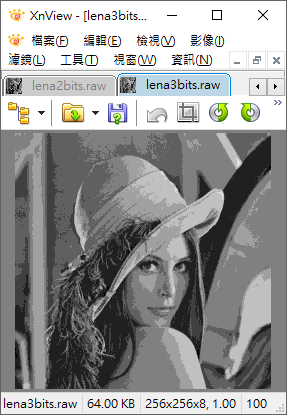
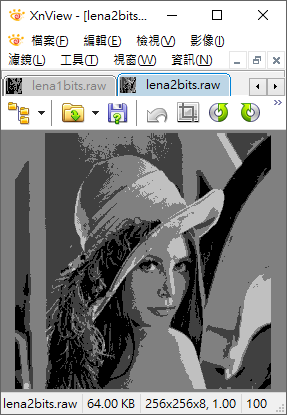
若將Visual Studio 方案組態Debug模式切換為Release模式，可以節省最多將近50倍的時間!

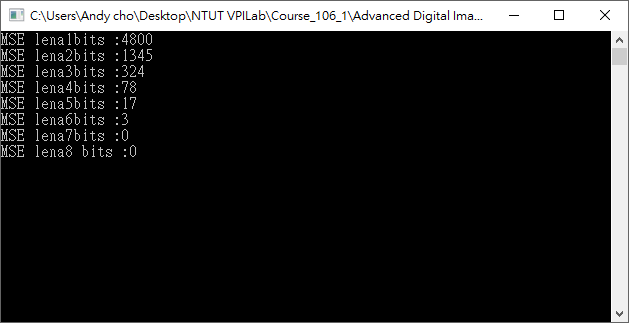
1. Gray-level resolution(C/C++)

Using C/C++ to quantize the gray-level resolution of lena\_256.raw and baboon\_256.raw from 8 bits to 1 bit. Show the results of these quantize images and the corresponding with MSE(Mean Square Error) value. Discuss the bit saving.

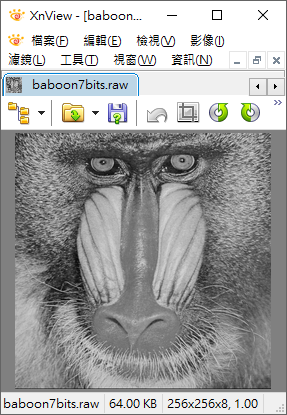
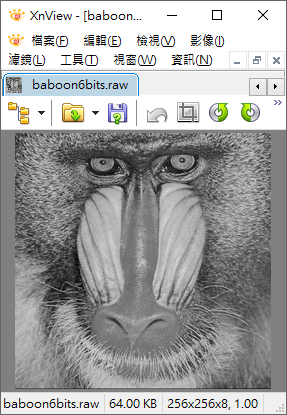
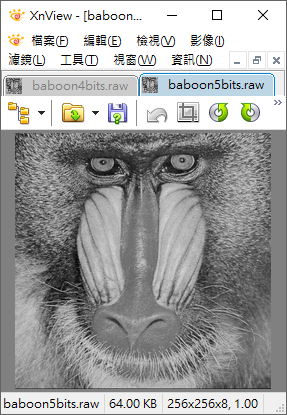
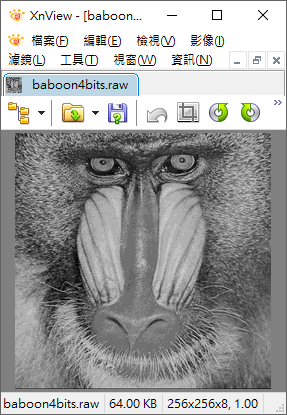
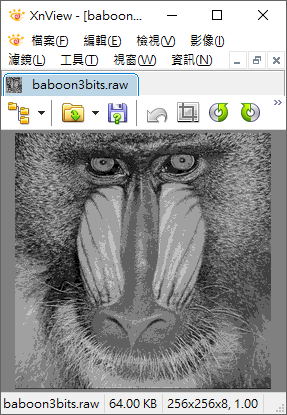
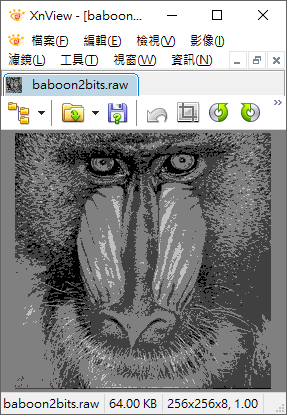
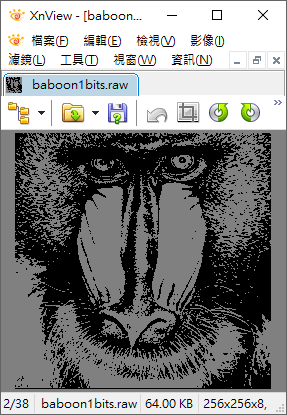
1. Figure

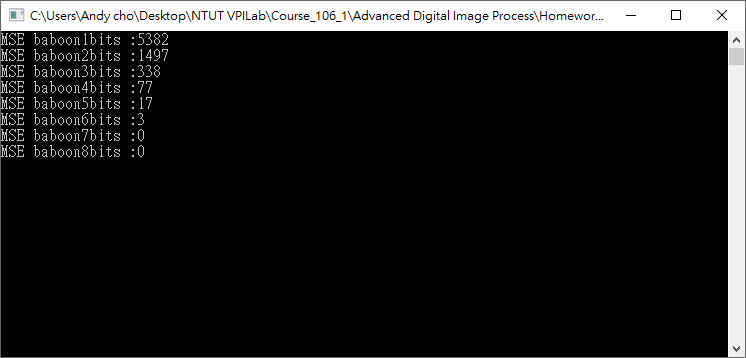
(lena\_256.raw)





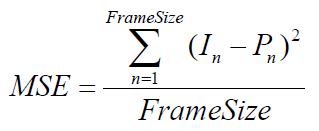
(baboon\_256.raw)





1. Discussion

MSE指 Mean Square Error，In指原始影像第n個pixel值，Pn指經處理後的影像第n個pixel值，FrameSize是影像長度x寬度x通道數（灰階為1，彩色為3）。



故當影像每個像素點省下越多的灰階值，則MSE就越高。

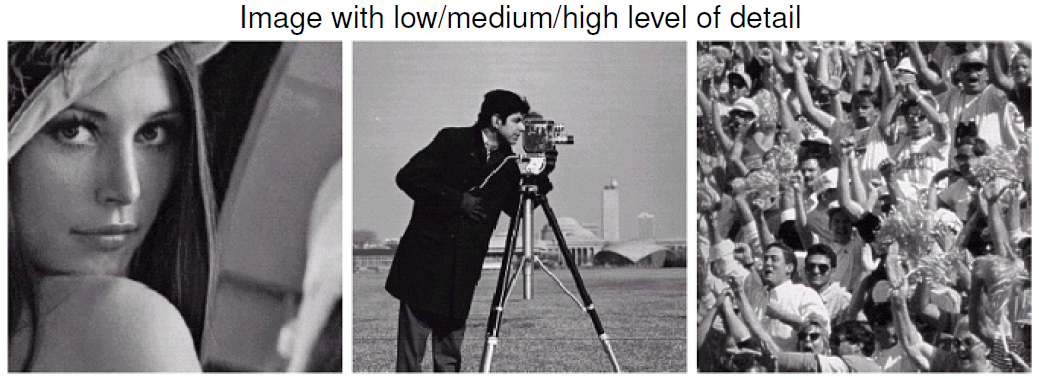
Bitrate 指的是單位時間內處理的位元數量，當色階逐次減少1bits，Bitrate 也越大，但是在最終的資料存取大小中並沒有任何差異，因為由Mat宣告的影像資訊一定為8bits只是我們能夠改變是否每增加1bit就要改變色階，以達到quantization。

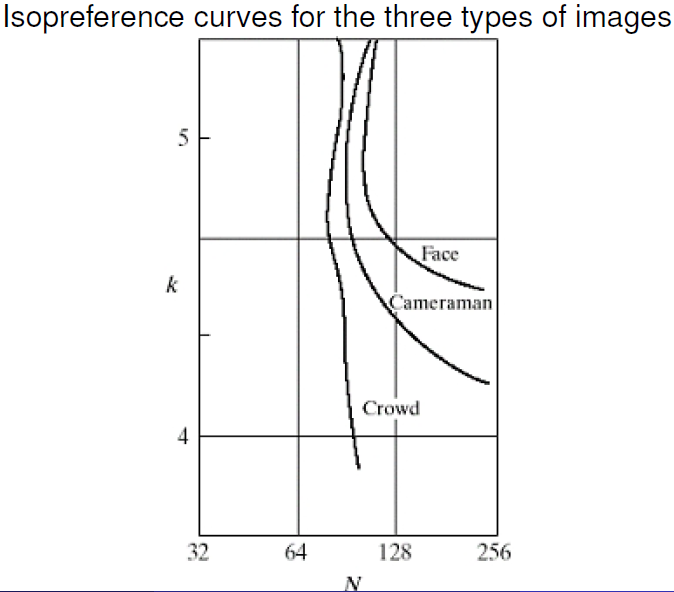
1. Isopreference test(C/C++)

Test the isopreference on lena and baboon images using the programs written in Problems 1 and 2.

1. Discussion

Isopreference 意思為當影像複雜度高的時候所需要相對少的色階，而相反的若影像複雜度低則要表現的平滑度可能就越明顯，則需要相對高的色階。





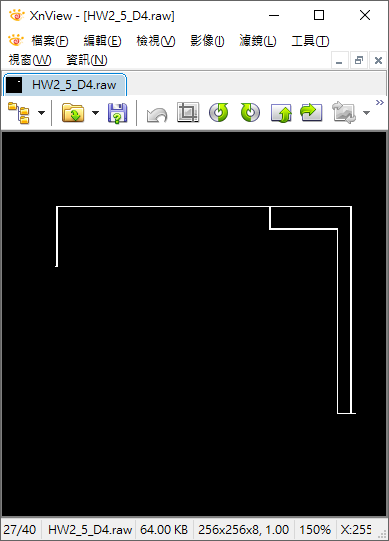
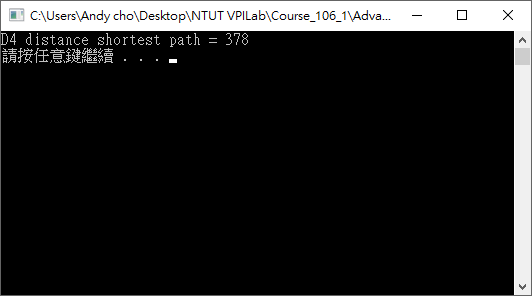
故在lena\_256.raw的影像需要的色階要比baboon\_256.raw高，因為相較於baboon\_256.raw，lena\_256.raw需要呈現平滑度。

1. Distance and Path(C/C++)

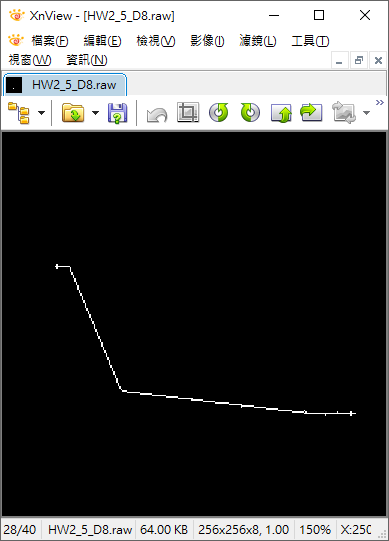
Find out the solution from A(35,89) to B(235,187) by using D4 and D8 distance on maze\_256.raw and show the shortest path(Don’t use ad hoc method to build the path)

1. Figure

(D4 distance)

(D8 distance)

1. Discussion

使用 D star algorithm 去解決最短路徑規劃的問題，我設了兩個矩陣，一個矩陣的內容存放距離數，另一個輸出成最短路徑，兩者大小都為256\*256。

第一步：我將初始化距離矩陣，將每一個內容都預設為256\*256，因為每點存放距離不可能超過整張影像的大小。

第二步：將起始點的位置對應至距離圖且將其設為0(起點距離為0)。

第三步：開始進入迴圈及判斷式，判斷當前位置在哪，對應至原圖是否有路可走，是否走過那條路。

第四步：設定判斷式在最內迴圈中，判斷若走到終點則程式停止，並輸出對短距離。

第五步：下一雙迴圈開始掃整張256\*256從終點依據1.距離遞減性2.連續性一個點一個點由終點往起點回推最終輸出路徑圖。