國立臺北科技大學
National Taipei University of Technology

Advanced Digital Image Process

HOMEWORK 2

Lab : VPILab

Advisor : Cheng-Ming Huang(黃正民)

Student : Yu Cho(卓諭)

Student ID : 106318025

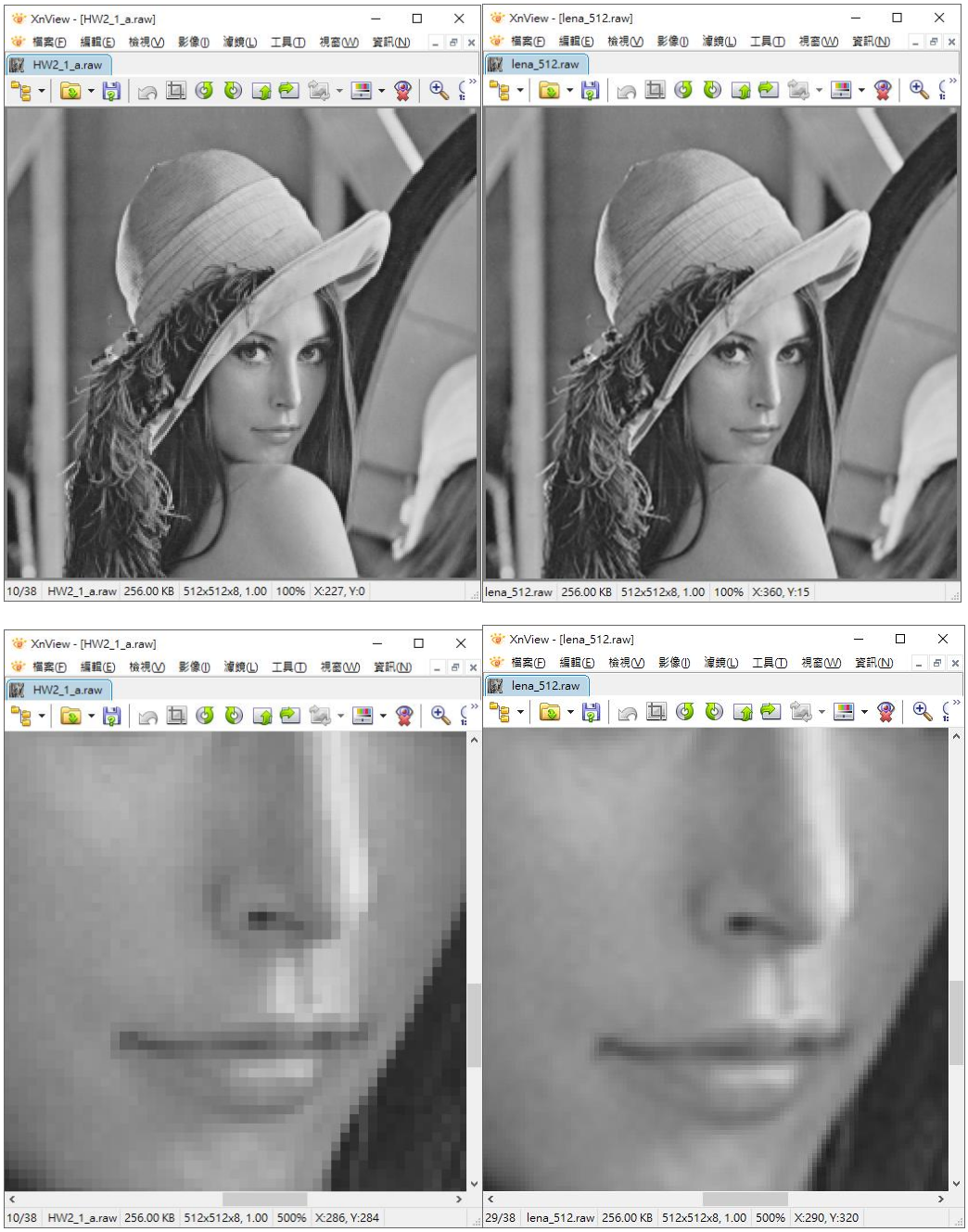
Data : 2017.10.16

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

Using C/C++ to perform the following tests on lena_256.raw

(a) Zooming the image with ratio 2:1 using row-column replication. Compare the output with lena_512.raw.

i. Figure



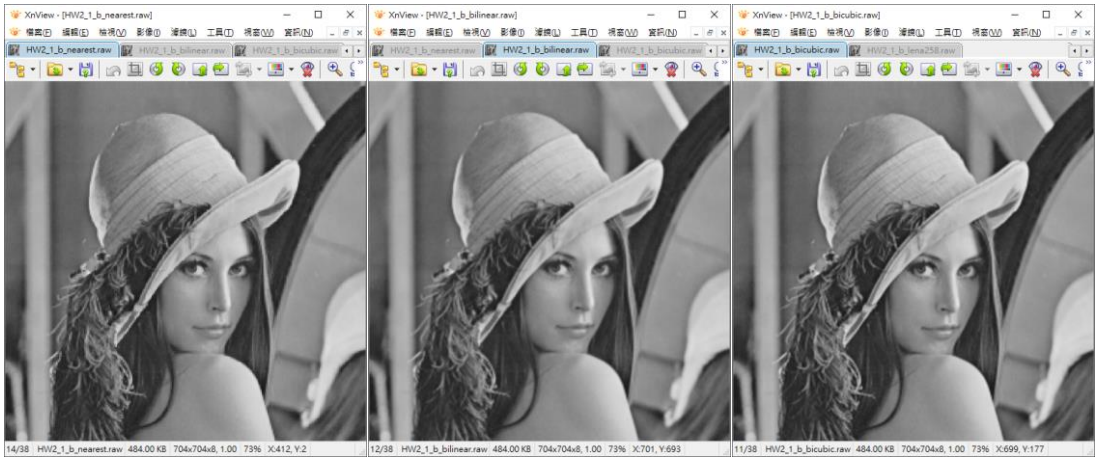
ii. Discussion

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。

(b) 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?

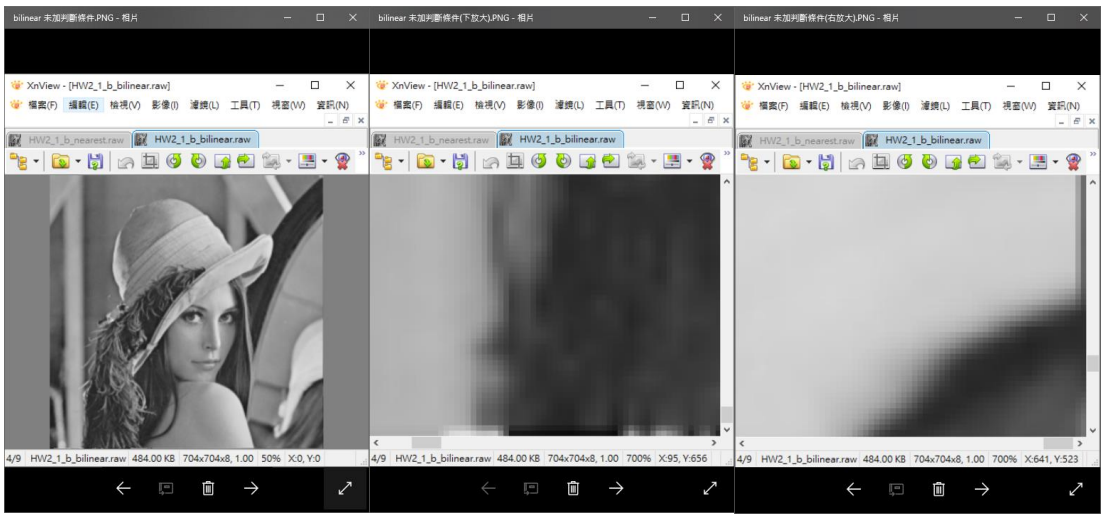
i. Figure

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



ii. 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。

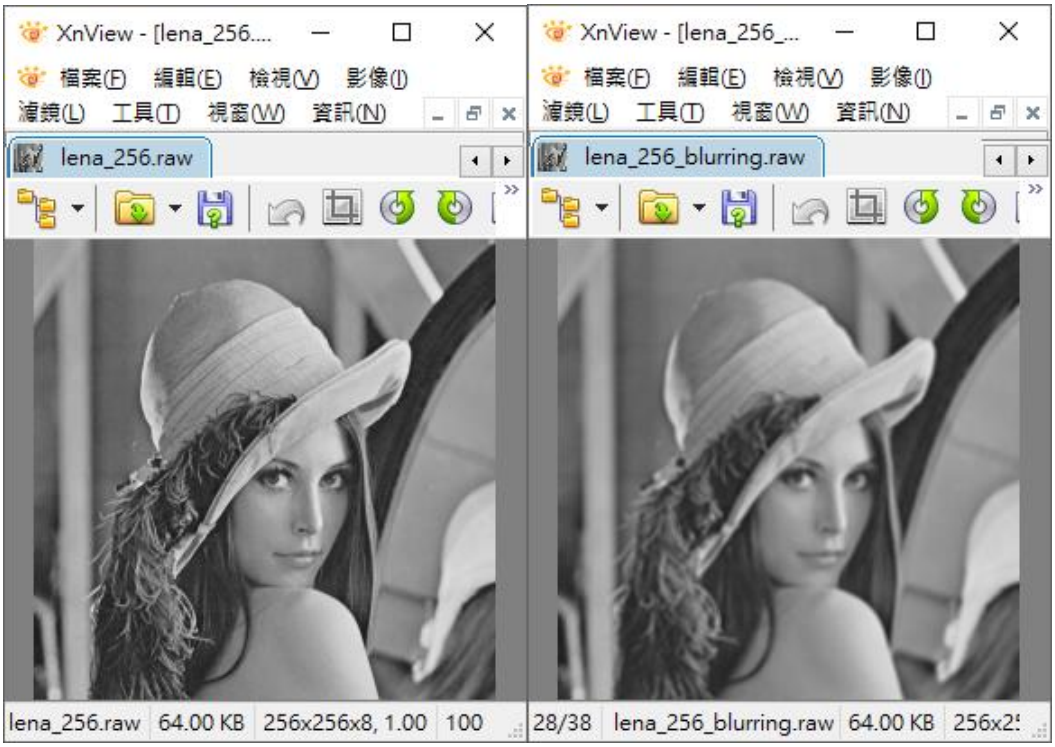
(bicubic interpolation)

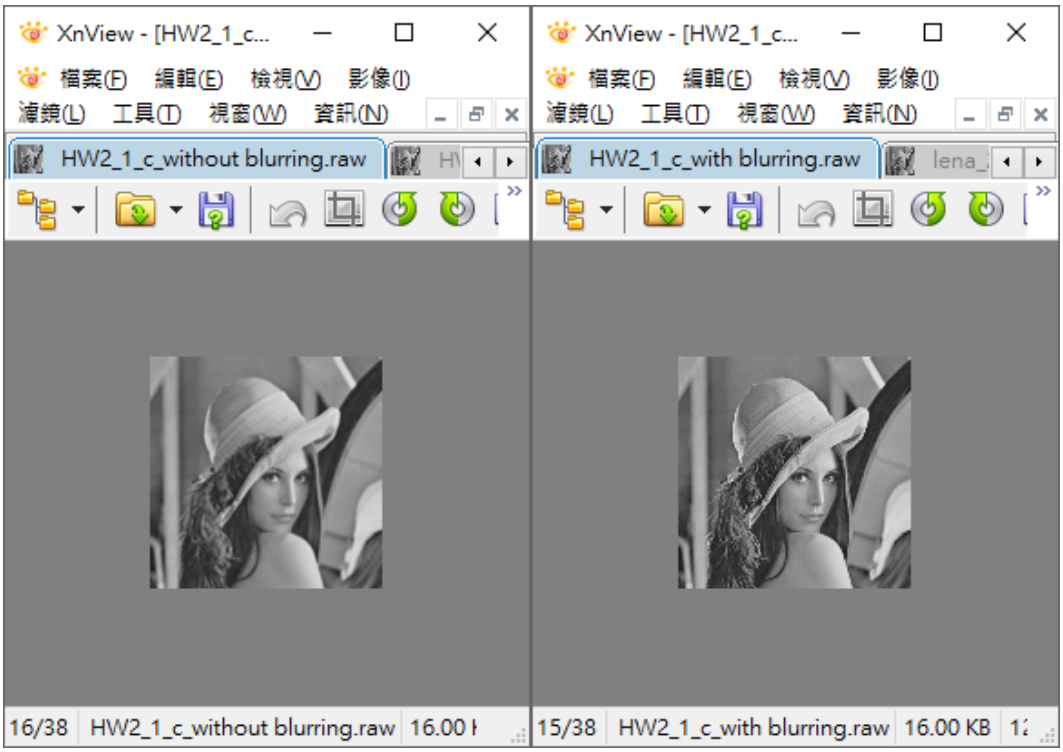
使用 bicubic 會遇到四邊邊界問題，原因是因為相較於 bilineear 取 4 個像素點做內插，bicubic 是取了周圍兩層也就是 16 個像素值做雙立方內插，我解決此邊界問題的方法為直接把原圖外擴出一個像素，256*256 變成 258*258，外框像素值與邊界像素值相等，再去使用 16 個像素值的雙立方內插。

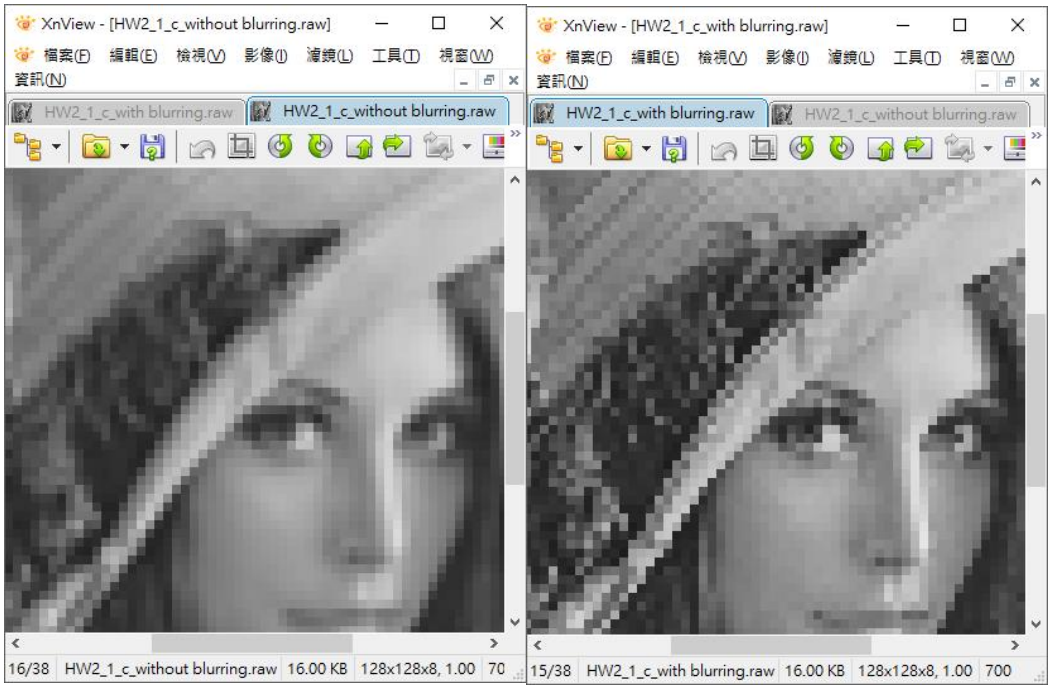
(c) Shrinking the image with ratio 1:2 row-column deletion. Check your result with or without blurring(using XnView) your input image before shrinking.

i. Figure

使用 XnView 模糊化 lena_256.raw，比較縮小結果。





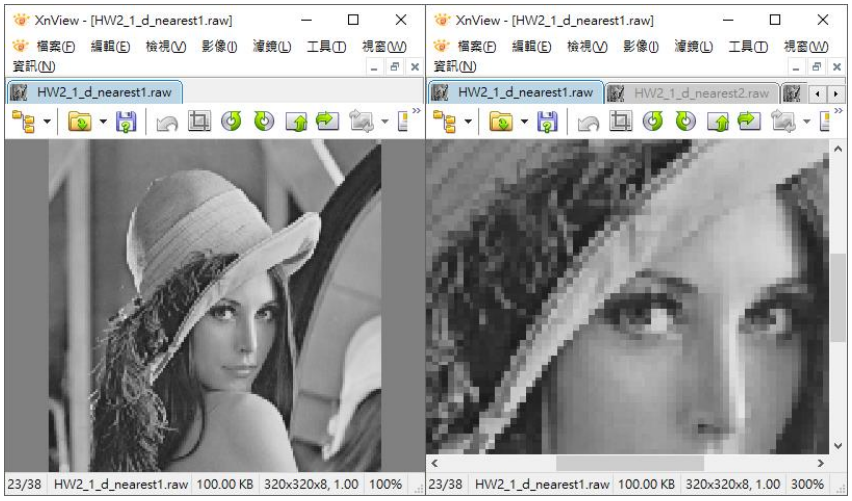


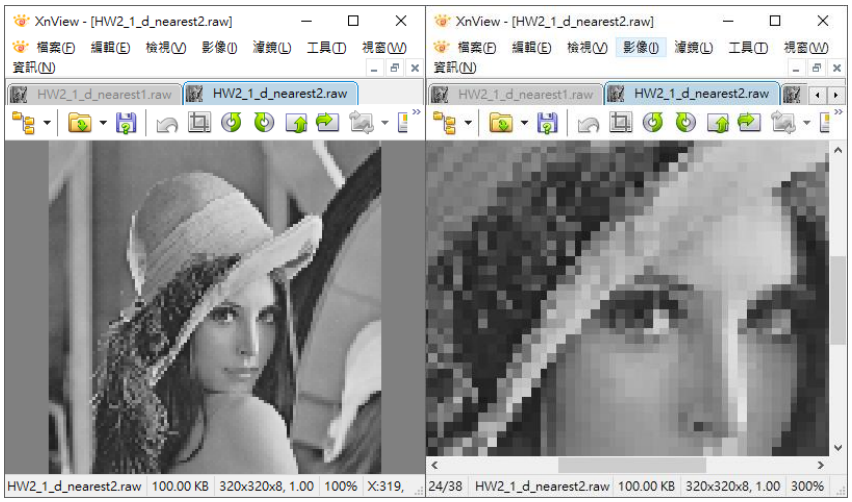
ii. Discussion

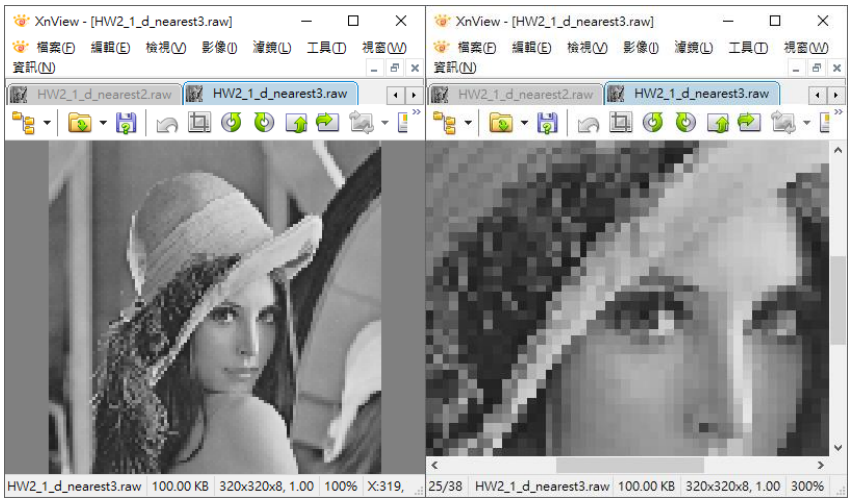
由結果圖可以得知，先做模糊化後再縮小會使處理後的影像對比度更為強烈，可以表現出更多細節。

(d) 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.

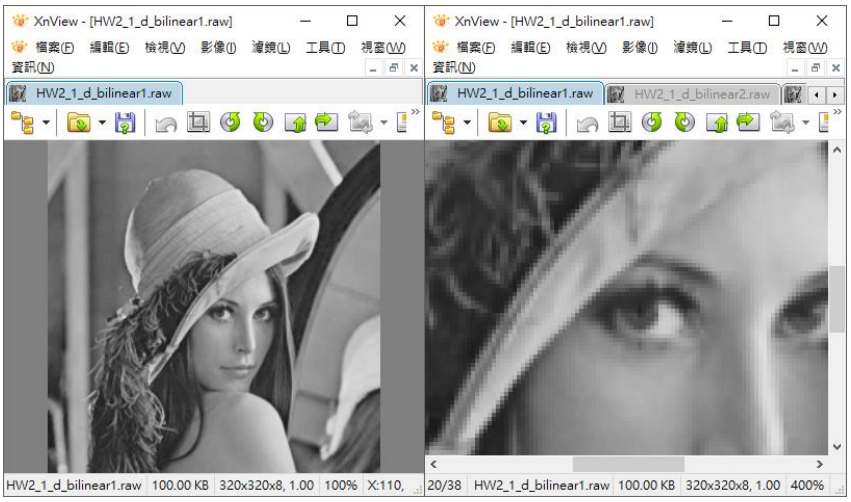
i. Figure(依序為↑2.75 ↓2.2、↓2.2 ↑2.75、↑1.25)
(nearest neighbor)

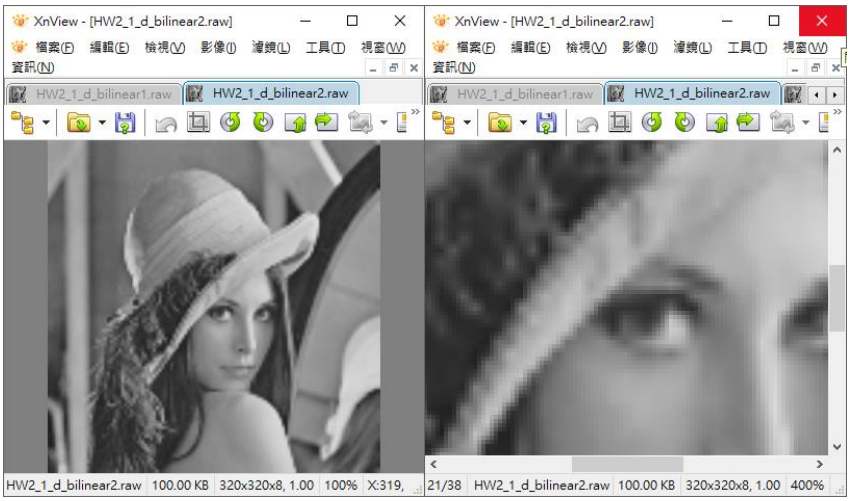


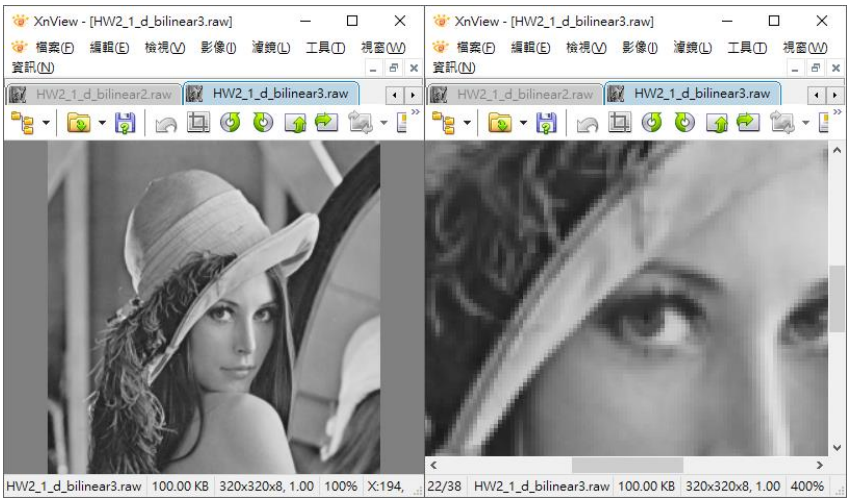




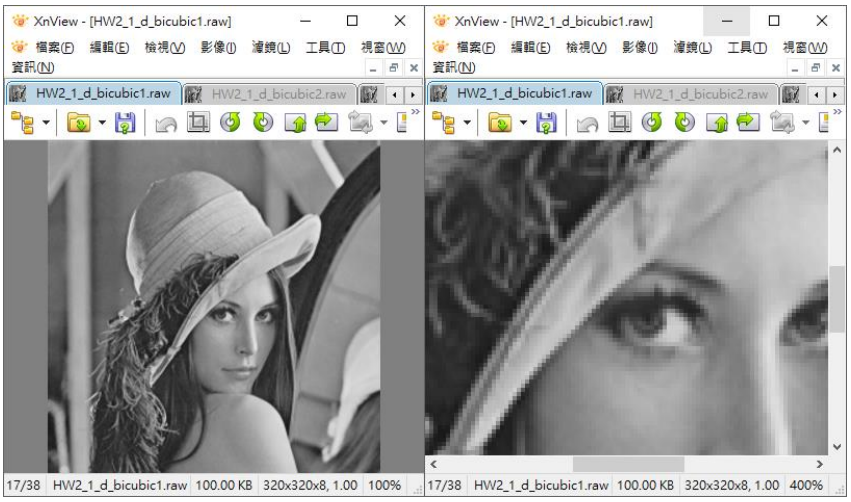
(bilinear)

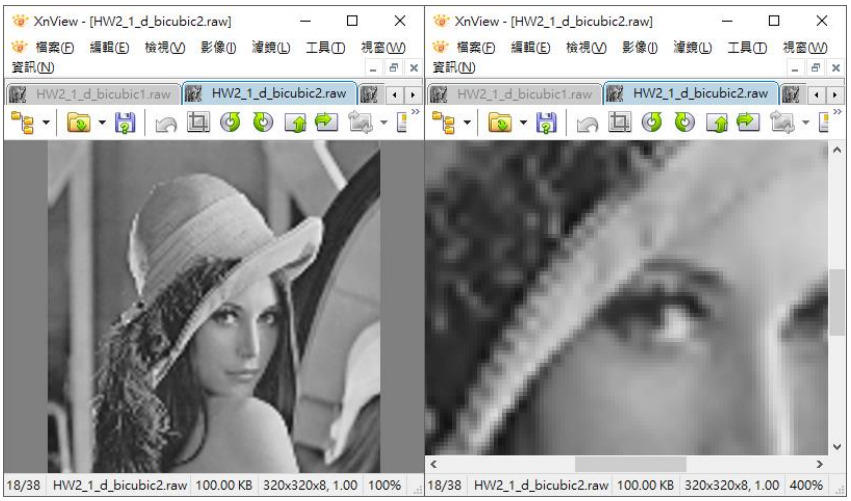

XnView - [HW2_1_d_bilinear1.raw]
HW2_1_d_bilinear1.raw
HW2_1_d_bilinear1.raw 100.00 KB 320x320x8, 1.00 100% X:110,
XnView - [HW2_1_d_bilinear1.raw]
HW2_1_d_bilinear1.raw
HW2_1_d_bilinear2.raw
20/38 HW2_1_d_bilinear1.raw 100.00 KB 320x320x8, 1.00 400%

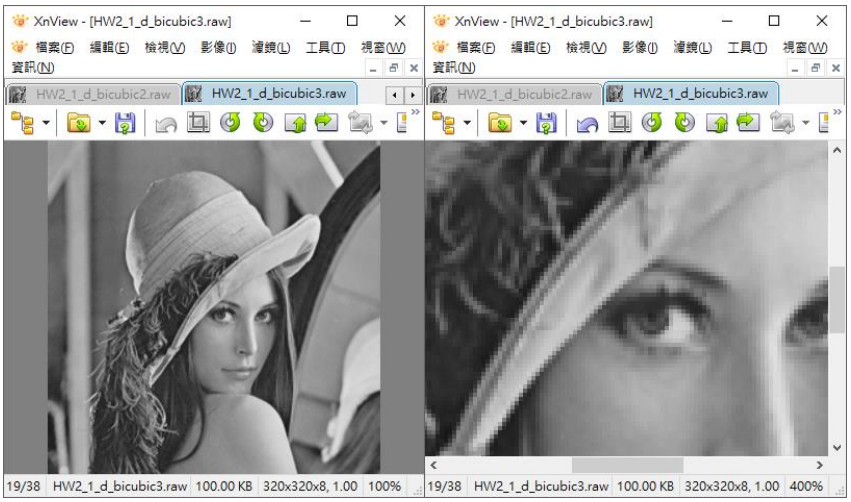

XnView - [HW2_1_d_bilinear2.raw]
HW2_1_d_bilinear1.raw
HW2_1_d_bilinear2.raw
HW2_1_d_bilinear2.raw 100.00 KB 320x320x8, 1.00 100% X:319,
XnView - [HW2_1_d_bilinear2.raw]
HW2_1_d_bilinear1.raw
HW2_1_d_bilinear2.raw
21/38 HW2_1_d_bilinear2.raw 100.00 KB 320x320x8, 1.00 400%


XnView - [HW2_1_d_bilinear3.raw]
HW2_1_d_bilinear2.raw
HW2_1_d_bilinear3.raw
HW2_1_d_bilinear3.raw 100.00 KB 320x320x8, 1.00 100% X:194,
XnView - [HW2_1_d_bilinear3.raw]
HW2_1_d_bilinear2.raw
HW2_1_d_bilinear3.raw
22/38 HW2_1_d_bilinear3.raw 100.00 KB 320x320x8, 1.00 400%

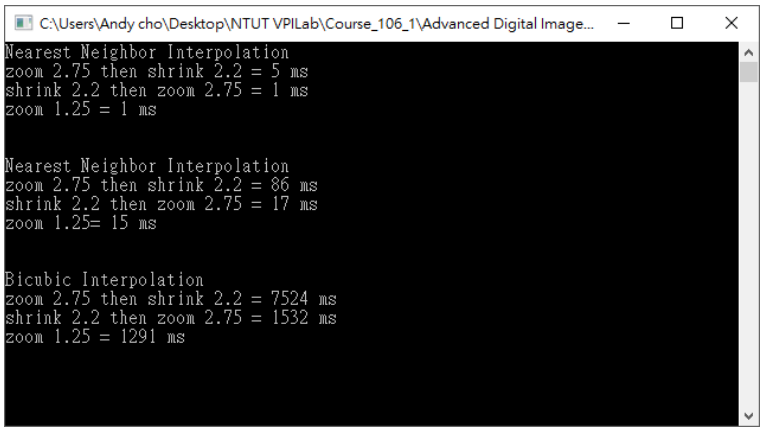
(bicubic)


XnView - [HW2_1_d_bicubic1.raw]
HW2_1_d_bicubic1.raw
HW2_1_d_bicubic2.raw
17/38 HW2_1_d_bicubic1.raw 100.00 KB 320x320x8, 1.00 100%
17/38 HW2_1_d_bicubic1.raw 100.00 KB 320x320x8, 1.00 400%

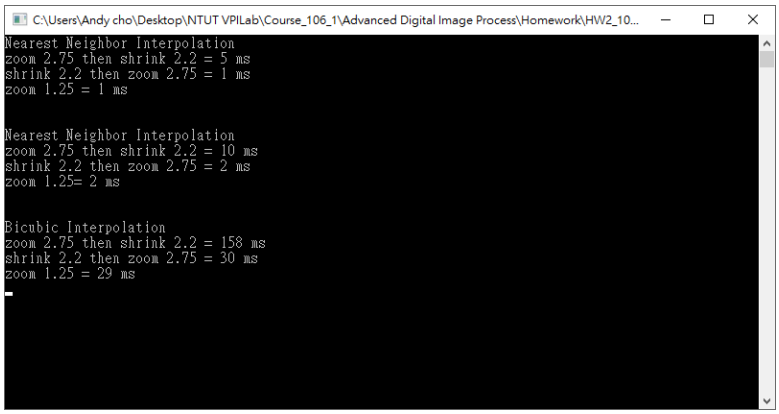

XnView - [HW2_1_d_bicubic2.raw]
HW2_1_d_bicubic1.raw
HW2_1_d_bicubic2.raw
18/38 HW2_1_d_bicubic2.raw 100.00 KB 320x320x8, 1.00 100%
18/38 HW2_1_d_bicubic2.raw 100.00 KB 320x320x8, 1.00 400%


XnView - [HW2_1_d_bicubic3.raw]
HW2_1_d_bicubic2.raw
HW2_1_d_bicubic3.raw
19/38 HW2_1_d_bicubic3.raw 100.00 KB 320x320x8, 1.00 100%
19/38 HW2_1_d_bicubic3.raw 100.00 KB 320x320x8, 1.00 400%

(方案組態：Debug Win32)



(方案組態：Release Win32)



ii. 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。

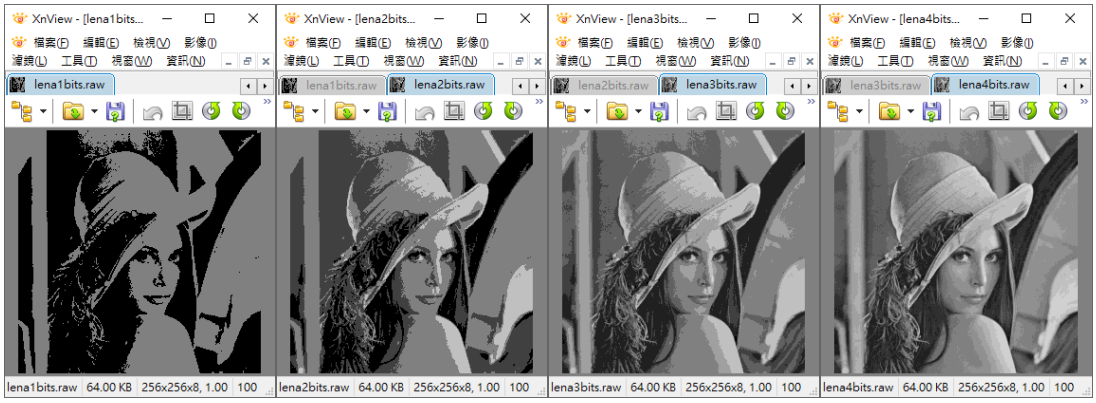
若將 Visual Studio 方案組態 Debug 模式切換為 Release 模式，可以節省最多將近 50 倍的時間!

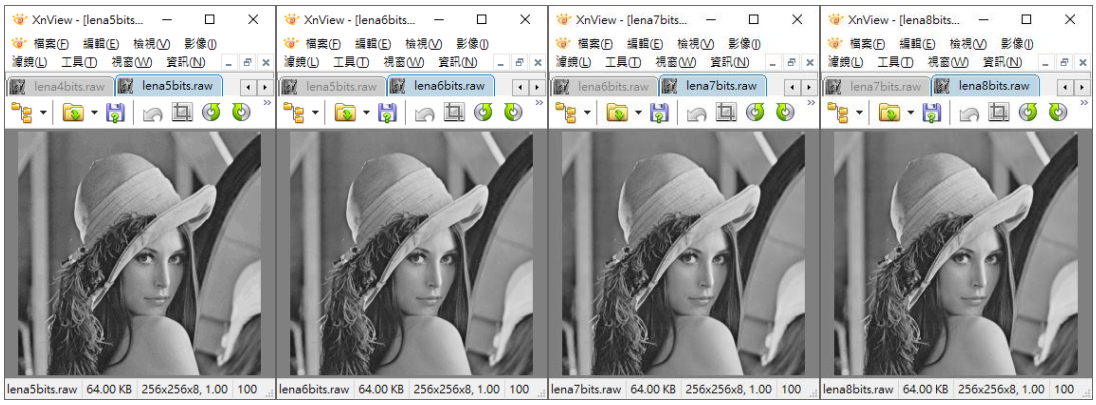
2. Gray-level resolution(C/C++)

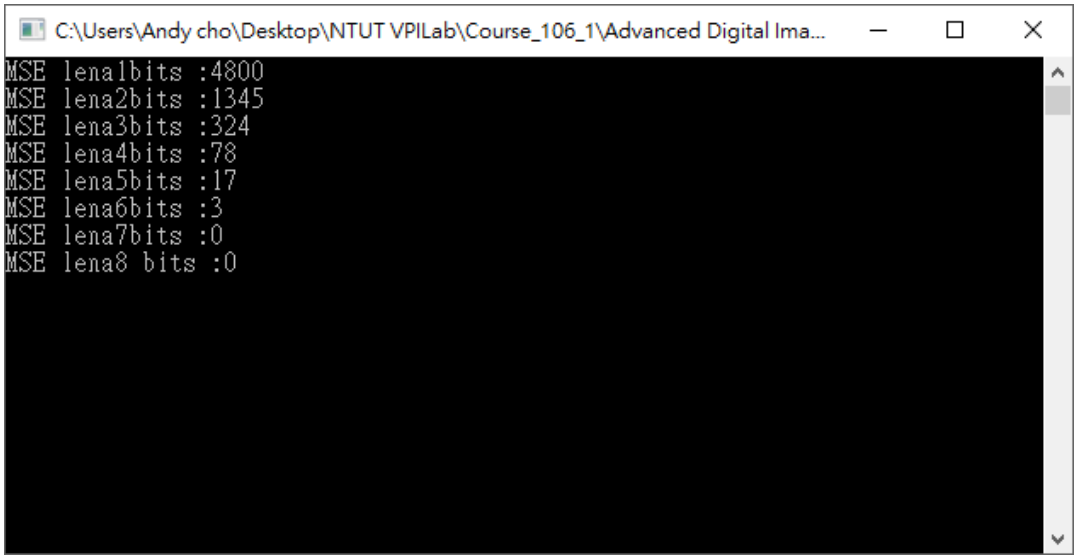
Using C/C++ to quantize the gray-level resolution of lena_256.raw and baboon_256.raw from 8 bits to 1 bit. Show the results of these quantize images and the corresponding with MSE(Mean Square Error) value. Discuss the bit saving.

i. Figure

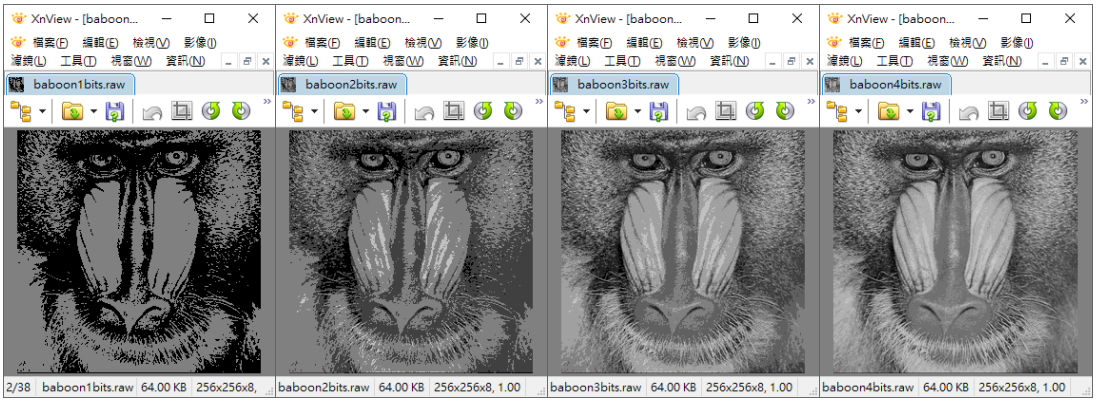
(lena_256.raw)

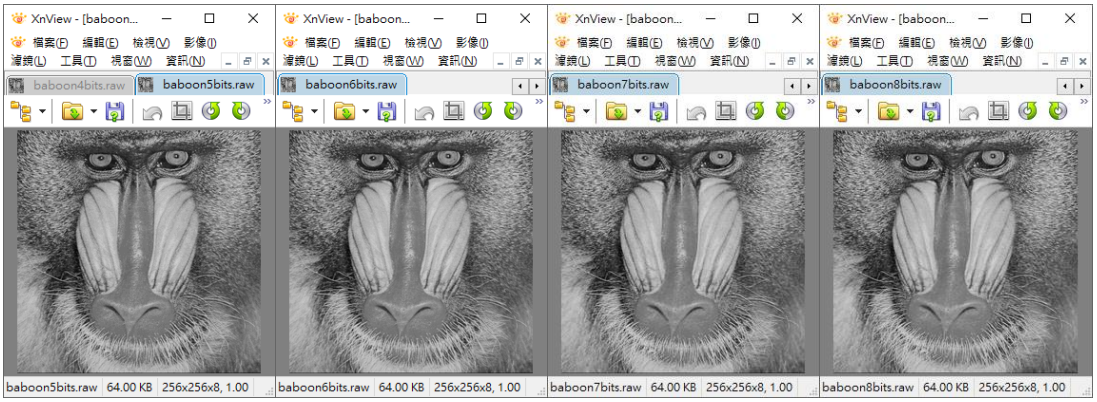


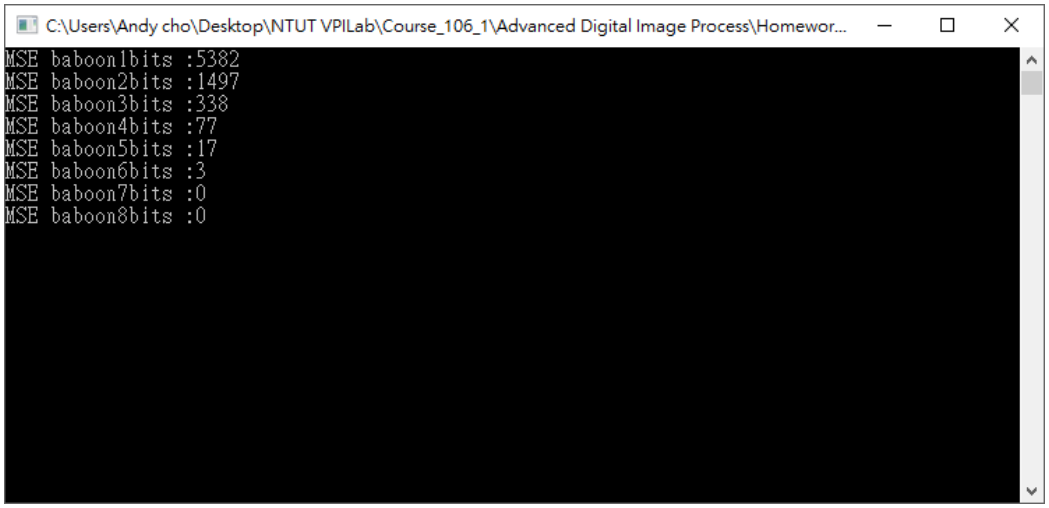




(baboon_256.raw)







ii. 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）。

$$MSE = \frac{\sum_{n=1}^{FrameSize} (I_n - P_n)^2}{FrameSize}$$

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。

3. Isopreference test(C/C++)

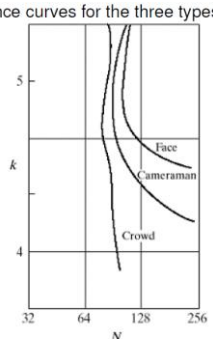
Test the isopreference on lena and baboon images using the programs written in Problems 1 and 2.

i. 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。

Image with low/medium/high level of detail

Isopreference curves for the three types of images



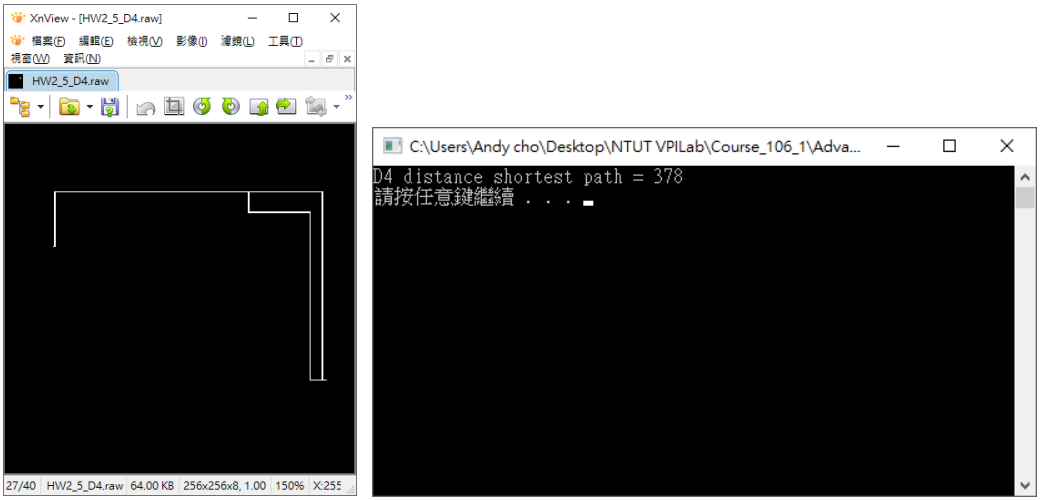
故在 lena_256.raw 的影像需要的色階要比 baboon_256.raw 高，因為相較於 baboon_256.raw，lena_256.raw 需要呈現平滑度。

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

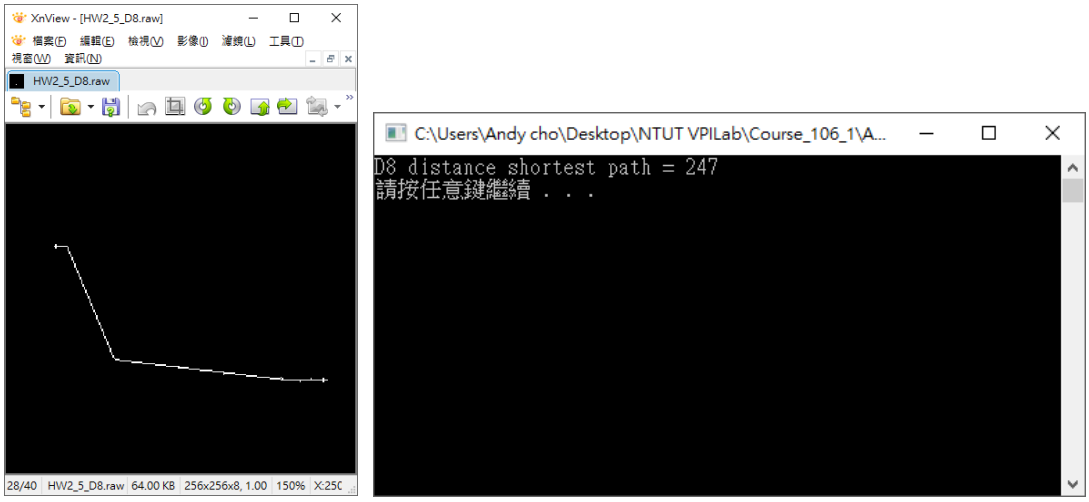
Find out the solution from A(35,89) to B(235,187) by using D_4 and D_8 distance on maze_256.raw and show the shortest path(Don't use ad hoc method to build the path)

i. Figure

(D_4 distance)



(D_8 distance)



ii. 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。