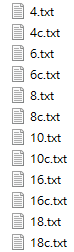
Numerical Analysis Homework 2

01157127 洪宗邑

․作業說明

1. 有灰階與彩色檔在K=4, 8, 16下所內插出的結果

•檔名為 ”數字.txt” 代表是灰階且數字等於K，若取兩邊pixel，則直接標示為 ”數字.txt” 代表是灰階且數字等於K+2

•檔名為 ’’數字c.txt” 則代表是彩色

/\* 為求辨識，將64\*64灰階檔名改為”gray1.txt”，512\*512改為”gray2.txt”

而64\*64彩色檔名改為”rgb1.txt”，512\*512改為”rgb2.txt” \*/

2. 有修改顯示程式使得display.exe得以顯示彩色圖，而須修改輸入規則

•使用時在exe後下個參數加上0表示輸入檔案為灰階

•使用時在exe後下個參數加上1表示輸入檔案為彩色

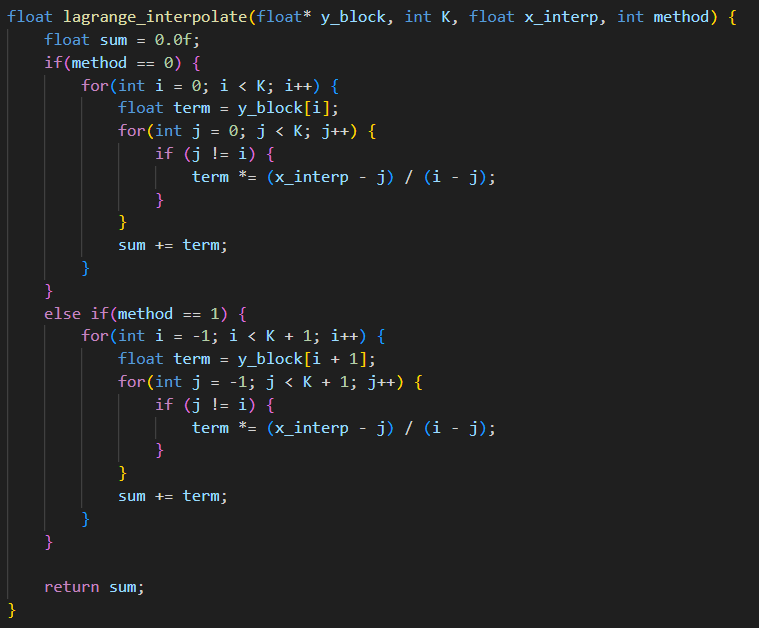
Ex: (灰階) display.exe 0 image1.txt

// 也有順便修改read.h, write.c以支援彩色檔

․計算思維與演算法–Lagrange interpolation

因為圖片位置是二維的，所以分成兩階段的內插，第一次對每一列做內插，解析度變化為NxN 🡪 NxM，第二次延續前面的輸入再對每一行做內插，解析度變化為NxM 🡪 MxM。

若直接對N個點進行N - 1的多項式內插會產生振盪，所以會將每幾個點分成一個區塊，來看區塊大小會對結果有甚麼影響。

因為若輸入跟函數一起位移，則其值不變，所以我主要的演算法是將原本區塊的K個點移到新的空間[0, K - 1]來做運算(不是看整體位置)，再找出新區塊中的每一個點對應到原本區塊的位置來做內插，程式碼如下：

而要將新影像位置映射到原本影像位置，根據原本區塊範圍[0, K - 1]和新區塊範圍[0, new\_block\_size-1]，對應程式碼如下：

*float step = (K-1)/(new\_block\_size -1.0);*

•在處理區塊範圍時有用一個變數method來做區別，等於零代表是普通情況(取K個)，等於一則包含兩邊各多取一個pixel，此時對應到其他函數也都是以此做判斷，並對兩種情況分別處理，需要特別說明的是，在演算法中原本的空間範圍我從[0, K - 1]修改為[-1, K]。

․遇到困難

1. 一開始想說把所有程式都整合到vscode的開發環境裡，結果在裝opengl 2.x跟freeglut時卻久久處裡不來，最後只好分開用，所以display的程式依然是在visual studio裡面，主要main函式則在vscode裡面處裡。

2. 因為要原本的檔都是處裡灰階的，而為了處裡彩色圖檔，自己設計了新的結構、新增對應的函式並修改了讀檔寫檔顯示檔等，內容並沒有很複雜但十分繁瑣且須小心細節，因此最後這反而是花費最多時間的部分。

3. 在整合程式時因為要處裡不同K值的情況、區塊取兩邊pixel的情況、灰階跟彩色圖檔，還要配合讀檔寫檔做調整，最後程式碼能優化的部分盡量做，但主要還是根據灰階跟彩色直接用暴力法分開取兩種情況，也因此程式碼的長度變得有點長。

․結果分析

|  |  |  |
| --- | --- | --- |
| Gray image | K | 1+K+1 |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |
| RGB image | K | 1+K+1 |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |

\*附上一個比較例子

// MSE越靠近0越好 / PSNR越高越好 / SSIM越靠近1越好

|  |  |  |  |
| --- | --- | --- | --- |
| K | 4 | 16 | 1+16+1 |
| 和正解比較結果  (Gray) |  |  |  |

1. 從以上圖片可以發現，不論是灰階還是彩色，對於內插的結果都是區塊範圍越小越好。

比如K=16時在每個區塊的靠近邊緣處有極為明顯的黑白方塊，因為Lagrange polynomial會同時穿過這16個點，而為了穿過所有點，他會在點與點之間產生扭曲，而在邊緣處的扭曲會被放大，因此會發生振盪，導致預測值一下超過正常範圍。

在K=4產生的影像是透過3次多項式內插出來的，會發現結果較為平順，且幾乎不會有振盪的情況發生。

2. 從以上圖片可以發現，不論是灰階還是彩色，每一個區塊多取兩邊的pixel產生的結果都會更好。

K=4的結果可能看不太出來，但K=16可以稍微看出pixel之間有變得比較滑順，振盪的效果也有平滑一點，因為一開始區塊跟區塊之間是完全獨立的，也就是兩區塊內插的結果基本上是不相關的，但如果我們多取兩邊的pixel，會導致兩相鄰區塊會有兩個pixel是共用的，這樣就能讓區塊邊緣之間更加平順。

․心得：

沒想到內插放大不如我一開始以為的多項式一次取多個點會比較準，反而是K=4所取的三次方程式最為平衡，而曾經的我使用套件只需簡單的寫上一行指令便能顯示漂亮的結果，沒想到其背後的原理跟整合都是一門學問，這次作業也讓我再次體會到影像處理的偉大。