Digital Image Processing HW1

M093781 趙宇涵

1. Image Input / Output (30%)

BMP，全名為 bitmap，是 Windows 系統採用的圖像檔案儲存格式，檔案由四個部分組成，分別是：

a. Bitmap file header

b. Bitmap info header

c. Color table (Palette)

d. Bitmap array

四個部分的儲存資料如下表所示：

	Shift	Name	Size (bytes)	Content
Bitmap File Header	0000h	Identifier (ID)	2	'BM'【註 1】
	0002h	File Size	4	整個點陣圖檔案的大小（單位：byte）
	0006h	Reserved	4	保留欄位
	000Ah	Bitmap Data Offset	4	點陣圖資料開始之前的偏移量（單位：byte）
Bitmap Info Header	000Eh	Bitmap Header Size	4	Bitmap Info Header 的長度【註 2】
	0012h	Width	4	點陣圖的寬度，以像素（pixel）為單位
	0016h	Height	4	點陣圖的高度，以像素（pixel）為單位【註 3】
	001Ah	Planes	2	點陣圖的位元圖層數【註 4】
	001Ch	Bits Per Pixel	2	每個像素的位元數 1：單色點陣圖（使用 2 色調色盤） 4：4 位元點陣圖（使用 16 色調色盤） 8：8 位元點陣圖（使用 256 色調色盤） 16：16 位元高彩點陣圖（不一定使用調色盤） 24：24 位元全彩點陣圖（不使用調色盤） 32：32 位元全彩點陣圖（不一定使用調色盤） 【註 5】
	001Eh	Compression	4	壓縮方式【註 6】： 0：未壓縮 1：RLE 8-bit/pixel 2：RLE 4-bit/pixel 3：Bitfields
	0022h	Bitmap Data Size	4	點陣圖資料的大小（單位：byte）【註 7】。
	0026h	H-Resolution	4	水平解析度（單位：像素/公尺）【註 8】
	002Ah	V-Resolution	4	垂直解析度（單位：像素/公尺）
	002Eh	Used Colors	4	點陣圖使用的調色盤顏色數【註 9】
	0032h	Important Colors	4	重要的顏色數【註10】
Palette	0036h	Palette	N*4	調色盤資料。 每個索引值指定一種顏色：0x00RRGGBB 其中最高位元組保留為零
Bitmap Array	-	Bitmap Data	-	點陣圖資料【註11】

參考資料：https://crazycat1130.pixnet.net/blog/post/1345538#mark-6-BI_BITFIELDS

其中，這次的作業會使用且需要運算的參數有 width, height, bits per pixels，分別為在 header 的第 18、22、28 個 bit，因此定義變數_height, _width, _bitDepth 來儲存這些值，接下來分為幾個步驟進行讀寫：

Read:

a. 建立一個 char 把原圖 header 的資訊讀取出來。

b. 讀取寬高深度等資料，指定到變數中。

c. Padding，由於 bmp 資料格式中，寬度的儲存大小必須要是 4 的倍數，因此設計一個算式來計算原圖中一行正確的儲存格數目，才能定義一個正確的變數大小。舉以下例子，若有一張圖寬度為 2 pixels，則在 bmp 內儲存的大小寬度要以 3*2+2=8 來儲存。

1	2	3	4	5	6	7	8

圖一、若 w = 2，原尺寸為 2*3 = 6 bit，需 padding 2 bit 使寬度為 4 的倍數，圖中黃色為畫素*3 通道應有尺寸，綠色為 padding 尺寸。

d. 最後一個步驟就是把檔案照順序寫入_image 變數中。

Write:

由於是大小不變的讀寫，因此_image 變數大小與讀取的檔案大小相同，因此只要使用 read 步驟中的_height, _width, _padding 即可完成複寫。

2. Resolution (30%)

這個部分的目的是減少色彩數量，因此這裡採用位元左移運算(<<)與位元右移運算(>>)來處理，每一像素由 8 個位元組成，可以透過位移來決定要省略後面多少位元，算式為：

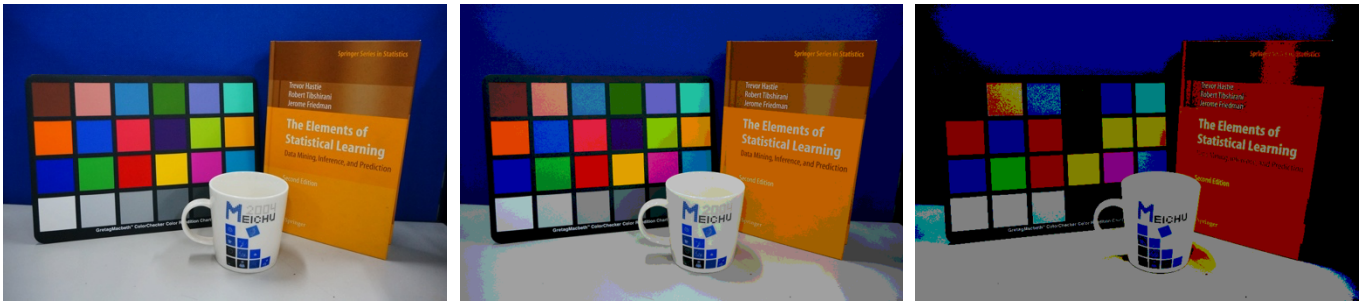
image_out = (image_in >> quanti_factor) << quanti_factor

舉例說明，若某一像素的色彩以二進位表示為 10110110，先使用右進位位移 2 個單位，會使數值變成 00101101，最後兩位的數值會被去除，接著再往左位移兩單位回去，後面的數值會補 0，最後變成 10110100，對每一個畫素都進行此操作，則會去除每個像素最後兩位元，亦即最後兩位元的數值對影像色彩無影響。quanti_factor 區間為 0~7 設計越大，代表省略的色彩越多，最極端的數值 7 指的是圖片剩下 0 和 128 兩個數值。範例如下：

原圖：	1	0	1	1	0	1	1	0		
>> 2：	0	0	1	0	1	1	0	1	~~1~~	~~0~~
<< 2 ：	1	0	1	1	0	1	0	0		

圖二、像素位移範例

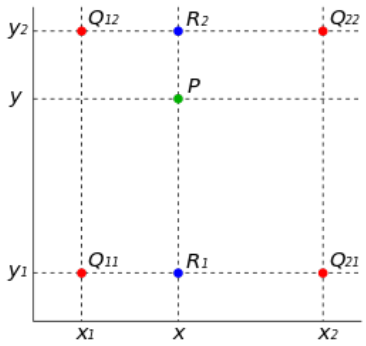
如圖所示，觀察最後兩個位元，會因爲位移而歸零。觀察輸出圖片，也可以發現當 quanti_factor 設定為 7 時，圖片顯示的色彩只有三原色：紅綠藍，以及三原色組合出的色彩：洋紅、黃、青色以及黑白。



圖三、由左至右 quanti_factor 設定為 2、5、7，可以觀察到最右邊的色彩僅剩三元色以及他們的組合色彩。

3. Scaling (40%)

此處題目要我們做的是用 Bilinear Interpolation 方法放大縮小 1.5 倍，此方法的重點是放大後，每個像素的值為原圖放大後所對應的的相鄰 4 個像素，各自乘上權重而取出的數值，以下用示意圖說明：



圖四、雙線性插值法示意圖

以圖四為例，欲求 P 點的數值，設原圖放大後的像素位置為 $Q_{11}(x_1, y_1)$、$Q_{12}(x_1, y_2)$、$Q_{21}(x_2, y_1)$、$Q_{22}(x_2, y_2)$，$a' = x_2 - x_1$ 、$b' = y_2 - y_1$、$a = x - x_1$ 、$b = y - y_1$ 則 P 點的線性加權數值算法為：

$$P(x,y) = ((a'-a)(b'-b)Q_{11} + a(b'-b)Q_{21} + b(a'-a)Q_{12} + abQ_{22}) / (a' * b')$$

另外，縮小則沒有這種限制，縮小僅需要選擇最靠近該像素的色彩填入即可。

實驗結果與檢討：

寫完這次作業，我發現有幾處問題無法解決：

1. Padding 數值：

   Padding 數值理論上是讓 w*channel 補齊為 4 的倍數，因此算式理應為(4-w*channel%4)%4 才對，但我使用這個算式跑 input2.bmp 卻無法正常顯示，直到更改成(channel-w*channel%4)%4 才會正常顯示。

2. Quantization resolution：

   在第二部分，我用 mac 系統運行，輸入 input1.bmp 可以正常顯示，輸入 input2.bmp 卻只能顯示位移 7 位元的結果，其他結果均無法顯示，但我借用同學的 windows 系統以及上傳到雲端，都可以正常顯示，這可能是因為 bmp 檔案為針對 windows 檔案開發，在 mac 系統中較為不相容的關係。

3. Scaling：

   與上述很類似的問題，輸入 input1.bmp 都可以正常執行，但是 input2.bmp 卻只有放大可執行，縮小則無法顯示，上傳到雲端的結果則是下圖：

   推測可能是在除以 1.5 的時候取整數錯誤所導致有些區塊區了一些顏色。

參考資料：

1. https://crazycat1130.pixnet.net/blog/post/1345538#mark-6-BI_BITFIELDS
2. https://www.86duino.com/?p=1413&lang=TW
3. https://jason-chen-1992.weebly.com/home/nearest-neighbor-and-bilinear-interpolation
4. https://github.com/yyeh26/NCTU_DIP_HW