中華民國第四十五屆中小學科學展覽會
作品說明書

高中組 生活與應用科學科

最佳團隊合作獎

040811

你可以再靠近一點－數位影像縮放之研究

國立桃園高級中學

作者姓名：
高二 陳浩法 高二 劉宗旻 高二 蔡旭宸
高一 郭善群

指導老師：
吳佳憲 蕭序良

第　45　屆中小學科學展覽會
作品說明書

科　　別：生活與應用科學

組　　別：高中組

作品名稱：你可以再靠近一點–數位影像縮放之研究

關 鍵 詞：影像處理、數位影像縮放、插值演算法

編　　號：

壹、摘要

本專題嘗試以程式處理數位影像的縮放，藉由縮放演算法及相關影像處理的研究與設計，期望能將縮放後的影像，維持一定的品質。若此技術能有所突破，相信未來數位影像的使用，應能更加的普及低廉。

貳、研究動機

「靠近，靠近一點，你可以再靠近一點……！」爲什麼明星能如此勇敢的展現自我呢？大概是「品質」吧！爲什麼世界上有「化妝品」的誕生呢？爲了遮掩外表上的不完美！每人都有缺點，數位影像也不例外！請看以下的例子：

一、目前流行的手機相機，雖然隨地隨拍很方便，但是沖洗出來的影像，其品質卻不能保證令人滿意。
二、在故宮的網站提供藝術品的圖片，雖然方便瀏覽，，然而許多名畫的呈現效果卻不如原圖清晰，放大時，也可能出現惱人的鋸齒狀。
三、工業界不可或缺的工程圖，由於零件越做越小，在傳送影像資訊時，若擷取影像設備的解析度不足，亦會出現影像圖片不清晰的問題。
四、最近許多醫院的 X 光片、斷層掃描等資料，也開始運用數位影像資訊的方式儲存及傳遞，但是細微部分的不清晰仍會造成醫師診斷的困擾。
五、重大刑案的監視器所拍攝到的車牌號碼或嫌犯因爲影像太小或不清楚，而導致案情膠著無法突破。

上述問題的解決方式或許可以從擷取設備（ 例如：掃描器、數位相機 ）的方向加強，但是一分錢一分貨、羊毛出在羊身上，高級擷取硬體（如高解析度及高倍率的光學鏡頭）價格昂貴，也不是那麼普及。而所謂「窮人也有窮人的玩法」，從普通的硬體擷取畫面，若能以程式將圖片縮放且維持一定的品質，相信應能符合大多數人的需求。

基於對這個問題的興趣，本小組決定以電腦課所學到的影像處理相關知識與演算法，配合自行研究的方法來挑戰以上「不可能的任務」！

參、研究目的

本專題主要的目的在探討如何將數位影像以程式進行縮放，並經由適當的處理來達到低失真且具一定品質的效果。雖然在市面上已有多數的影像處理軟體能處理影像縮放的工作，但放大後的圖片仍不盡然理想，因此本專題希望藉由對縮放演算法的相關研究並結合相關的數位影像處理(如濾鏡)，尋找出能夠使一般人能夠接受的影像縮放解決方案。

本專題的最終目的，則是希望數位縮放的技術能夠在特定領域取代光學縮放

的技術，使一般大眾能夠以合理(低廉)的代價享受高品質的影像。屆時以價格低廉的入門數位相機，或許就能達到目前仍嫌昂貴的專業數位相機的拍攝品質了。

肆、研究設備

一、硬體：個人電腦、筆記型電腦、數位相機

二、軟體：Borland C++ Builder、Shortcut PhotoZoom Pro、 PhotoImpact 7

伍、研究過程或方法

一、點陣影像縮放演算法

影像的縮放是影像處理裡面重要的一環，使用合適的演算法來進行影像縮放，對於影像的品質的提升有很大的幫助。目前常見的演算法有**最鄰近點演算法**(Nearest Neighbor)、**雙線性插值演算法**（Bilinear Interpolation）、**雙三次插值演算法**（Bicubic Interpolation）……等，以下爲其相關介紹。

(一)最鄰近點演算法(Nearest Neighbor)

在**最鄰近點演算法**中，經縮放後的像素顏色，乃是參考其在原影像中最靠近的像素的顏色。如圖，$C_{x,y}$、$C_{x+1,y}$、$C_{x,y+1}$、$C_{x+1,y+1}$爲原影像中暨有之像素，$C_{x',y'}$爲影像縮放過後新增的像素。因爲$C_{x,y}$ 的位置最靠近$C_{x',y'}$，所以我們可以把$C_{x,y}$當作是$C_{x',y'}$的合理數值。將所有像素依此類推，即可得到縮放後的影像。

最鄰近點演算法：
令 zoom_ratio 爲縮放倍率
欲求$C_{x',y'}$
x=(int)(x′/zoom_ratio)
y=(int)(y′/zoom_ratio)
$C_{x',y'}= C_{x,y}$

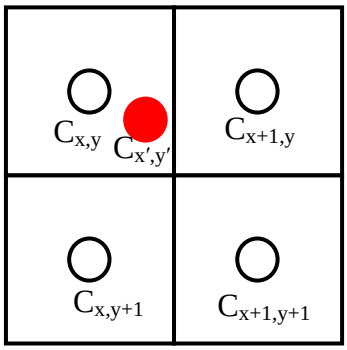


圖 7

最鄰近點插值演算法是最簡單也是最快的一種縮放演算法，因爲缺少的像素透過直接使用最靠近的原有像素來生成，但是這樣做的缺點則爲容易產生明顯的鋸齒狀（見下圖）。

圖 8、經最鄰近點演算法放大 4 倍後的影像

(二)雙線性插值演算法（Bilinear Interpolation）

雙線性插值演算法的作法如下：每個新影像中的像素爲原影像中與其最臨近的四個像素（2×2），依其距離施以不同比重運算而得的結果，這種演算法可有效地消除鋸齒現象。

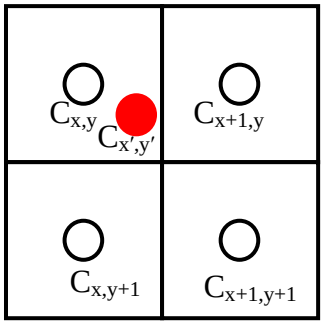


圖 9

雙線性插值演算法：

令 $\triangle x= x'-x$ ，$\triangle y= y'-y$

$C_{x',y'}= C_{x,y}+\triangle x(C_{x+1,y}-C_{x,y})+\triangle y(C_{x,y+1}+\triangle x(C_{x+1,y+1}-C_{x,y+1})-(C_{x,y}+\triangle x(C_{x+1,y}-C_{x,y})))$

$= C_{x,y}+\triangle xC_{x+1,y}-\triangle xC_{x,y}+\triangle yC_{x,y+1}+\triangle y\triangle xC_{x+1,y+1}-\triangle y\triangle xC_{x,y+1}-\triangle yC_{x,y}-\triangle y\triangle xC_{x+1,y}+\triangle y\triangle xC_{x,y}$

$= C_{x,y}+\triangle x(C_{x+1,y}-C_{x,y})+\triangle y(C_{x,y+1}-C_{x,y})+\triangle x\triangle y(C_{x,y}+C_{x+1,y+1}-C_{x,y+1}-C_{x+1,y})$

雙線性插值演算法的優點爲經放大後的影像較不容易產生鋸齒狀，但卻有放大後影像較模糊，且運算速度較慢的缺點。

圖 10、經雙線性插值演算法放大 4 倍後的影像

(三)雙三次插值演算法（Bicubic Interpolation）

雙三次插值演算法是雙線性插值演算法的改進演算法，此演算法中的每個像素都是原圖 16 個像素（4×4）運算的結果。這種演算法是一種很常見的演算法，普遍用在影像編輯軟體、印表機列印和數位相機上。

雙三次插值演算法：

令 $\triangle x = x' - x$　$\triangle y = y' - y$

$t1 = -\triangle x(1-\triangle x)(1-\triangle x)C_{x-1,y-1} + (1-2\triangle x^2+\triangle x^3)C_{x,y-1} + \triangle x(1+\triangle x-\triangle x^2)C_{x+1,y-1} - \triangle x^2(1-\triangle x)C_{x+2,y-1}$

$t2 = -\triangle x(1-\triangle x)(1-\triangle x)C_{x-1,y} + (1-2\triangle x^2+\triangle x^3)C_{x,y} + \triangle x(1+\triangle x-\triangle x^2)C_{x+1,y} - \triangle x^2(1-\triangle x)C_{x+2,y}$

$t3 = -\triangle x(1-\triangle x)(1-\triangle x)C_{x-1,y+1} + (1-2\triangle x^2+\triangle x^3)C_{x,y+1} + \triangle x(1+\triangle x-\triangle x^2)C_{x+1,y+1} - \triangle x^2(1-\triangle x)C_{x+2,y+1}$

$t4 = -\triangle x(1-\triangle x)(1-\triangle x)C_{x-1,y+2} + (1-2\triangle x^2+\triangle x^3)C_{x,y+2} + \triangle x(1+\triangle x-\triangle x^2)C_{x+1,y+2} - \triangle x^2(1-\triangle x)C_{x+2,y+2}$

$C_{x',y'} = -\triangle y(1-\triangle y)(1-\triangle y)t1 + (1-2\triangle y^2+\triangle y^3)t2 + \triangle y(1+\triangle y-\triangle y^2)t3 + \triangle y^2(\triangle y-1)t4$

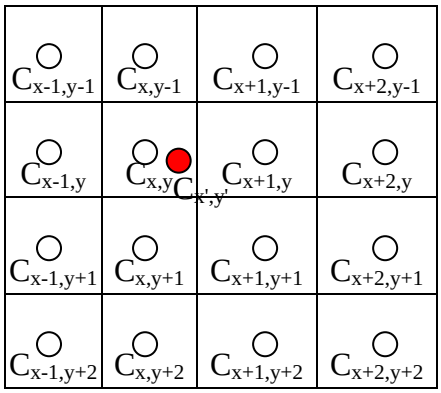


圖 11

雙三次插值演算法的優點爲：經放大後的影像暨不容易產生最鄰近點插值法的鋸齒狀，模糊的程度也比雙線性插值法來的輕微，但運算速度則比前兩種方法來的慢。

圖 12、經雙三次插值演算法放大 4 倍後的影像

二、濾鏡(Filter)

當我們用前述的方法放大影像後，會發現經放大後的影像邊緣部份出現鋸齒狀(使用 Nearest Neighbor 演算法)或呈現模糊的情形(使用 Bilinear 及 Bicubic Interpolation 演算法)。此時可利用濾鏡運算來改善，而經過處理後的圖形看起來會較自然。濾鏡的演算方式與像素本身與其周圍像素的對比有關，請看以下的介紹。

(一)濾鏡的原理

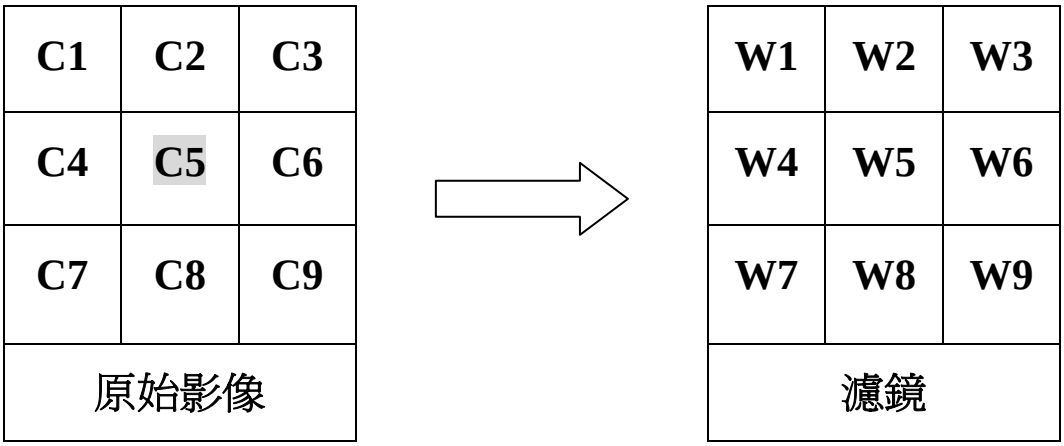


圖 13

上圖中左方九宮格中的代表原始影像中某一像素及其四周像素的顏色值，右方九宮格則代表濾鏡的參數。則 C5 像素經過濾鏡處理後的顏色可以下列公式表示：

C5(處理後) = C1*W1+C2*W2+C3*W3+ C4*W4+C5*W5+C6*W6+
C7*W7+C8*W8+C9*W9

(二)柔化(低通)濾鏡(Smooth/Low Pass Filter)

柔化(低通)濾鏡的設計原理，基本上乃是參考某一像素及其周圍像素再加以均化(Average)而達成模糊的效果，可以用來修飾鋸齒狀過於明顯的影像。本專題使用的柔化濾鏡請見下圖：

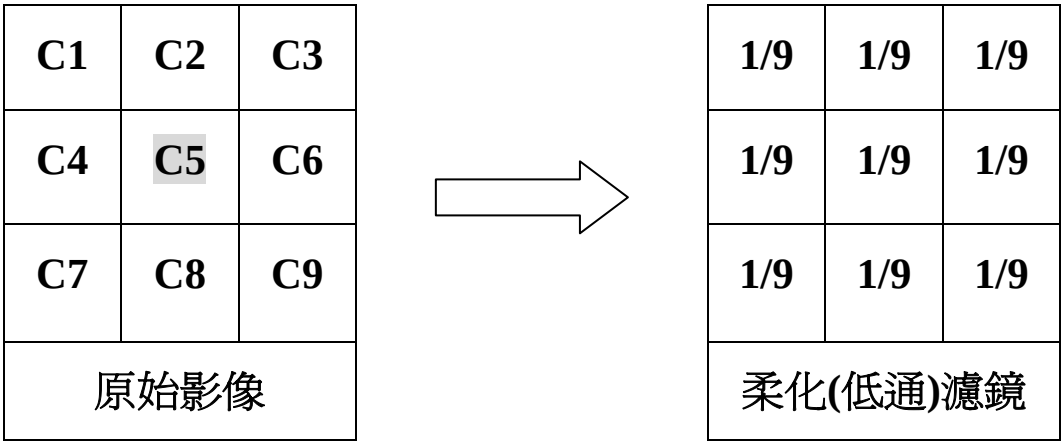


圖 14

C5 像素經過柔化濾鏡處理後的顏色可以下列公式表示：

C5(處理後) = C1*1/9+C2*1/9+C3*1/9+ C4*1/9+C5*1/9+C6*1/9+ C7*1/9+C8*1/9+C9*1/9

例：

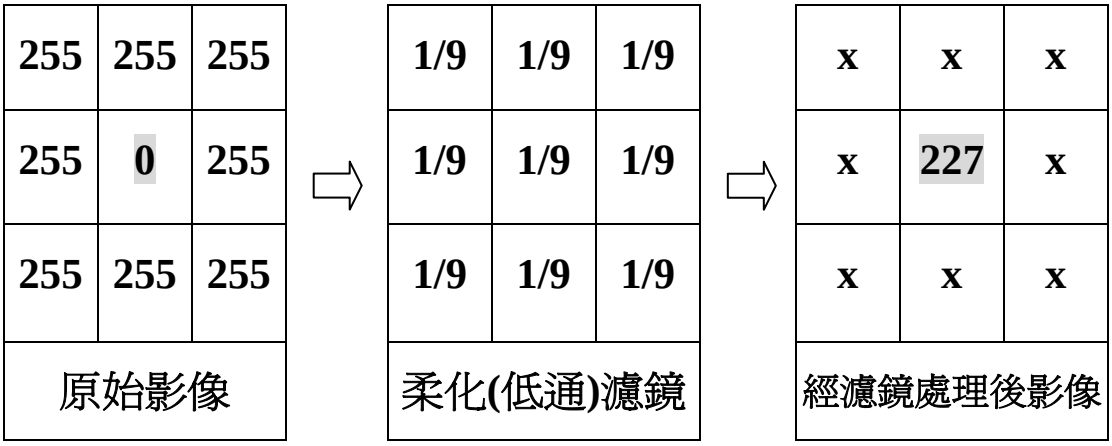


圖 15

經過柔化濾鏡處裡前後的影像請參考下圖：

圖 16、原始影像

圖 17、經柔化濾鏡處理後之影像

由以上兩對照圖片可清楚的看出，經柔化濾鏡處理過後的圖片，鋸齒狀的現象已明顯改善，邊緣的部分較爲圓滑且自然了許多。

(三)銳利(高通)濾鏡(Sharp/High Pass Filter)

銳利(高通)濾鏡的設計原理，基本上乃是參考周圍像素並加以提升其反差(Contrast)而達成銳利的效果。本專題使用的銳利濾鏡請見下圖：

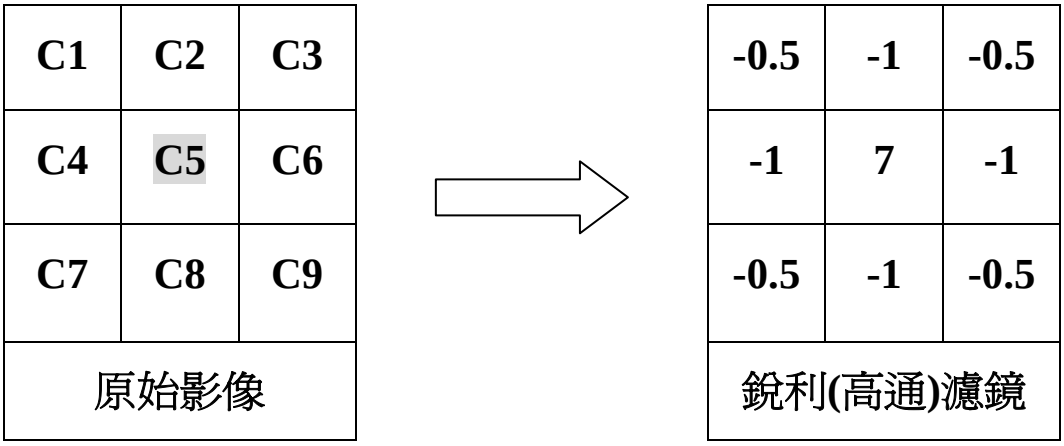


圖 18

C5 像素經過銳利濾鏡處理後的顏色可以下列公式表示：

C5(處理後) = C1*(-0.5)+C2*(-1)+C3*(-0.5)+ C4*(-1)+C5*7+C6*(-1)+ C7*(-0.5)+C8*(-1)+C9*(-0.5)

一般來說，提升對比度亦即提升銳利度，而對比度可分爲以下三種情形來處理：
1.當中間像素的顏色值較周圍低時，經加強對比處理後其顏色值會變更低
2.當中間像素的顏色值較周圍高時，經加強對比處理後其顏色值會變更高
3.當中間像素的顏色值與周圍相同時，經加強對比處理後其顏色值不變

故上述銳利濾鏡公式可簡單證明如下

令 C′=average(C1+C2+C3+C4+C6+C7+C8+C9)
則 C5(處理後)=C5*7+C′*(-0.5-1-0.5-1-1-0.5-1-0.5)=C5*7-C′*6
C5(處理後)-C5=C5*7-C′*6-C5=C5*6-C′*6=(C5-C′)*6

若 C5<C′，則 C5(處理後)-C5<0 => C5(處理後)<C5，亦即符合 1 之處理情形
若 C5>C′，則 C5(處理後)-C5>0 => C5(處理後)>C5，亦即符合 2 之處理情形
若 C5=C′，則 C5(處理後)-C5=0 => C5(處理後)=C5，亦即符合 3 之處理情形

例一：當中間像素的顏色值較周圍低時，經濾鏡處理後顏色值會變更低(127=>67)。

137	**137**	**137**
137	**127**	**137**
137	**137**	**137**
原始影像		

-0.5	**-1**	**-0.5**
-1	**7**	**-1**
-0.5	**-1**	**-0.5**
銳利(高通)濾鏡		

X	**X**	**X**
X	**67**	**X**
X	**X**	**X**
經濾鏡處理後影像		

圖 19

例二:當中間像素的顏色值較周圍高時,經濾鏡處理後顏色值會變更高(127=>187)。

117	**117**	**117**
117	**127**	**117**
117	**117**	**117**
原始影像		

-0.5	**-1**	**-0.5**
-1	**7**	**-1**
-0.5	**-1**	**-0.5**
銳利(高通)濾鏡		

X	**X**	**X**
X	**187**	**X**
X	**X**	**X**
經濾鏡處理後影像		

圖 20

例三：當中間像素的顏色值與周圍相同時，經濾鏡處理後顏色值不變(127=>127)。

127	**127**	**127**
127	**127**	**127**
127	**127**	**127**
原始影像		

-0.5	**-1**	**-0.5**
-1	**7**	**-1**
-0.5	**-1**	**-0.5**
銳利(高通)濾鏡		

X	**X**	**X**
X	**127**	**X**
X	**X**	**X**
經濾鏡處理後影像		

圖 21

經過銳利濾鏡處裡前後的對照影像請參考下圖：

圖 22、原始影像

圖 23、經銳利濾鏡處理後之影像

由以上兩對照圖片可清楚的看出，經銳利濾鏡處理過後的圖片，邊緣及細節的部分較原影像來的明顯!

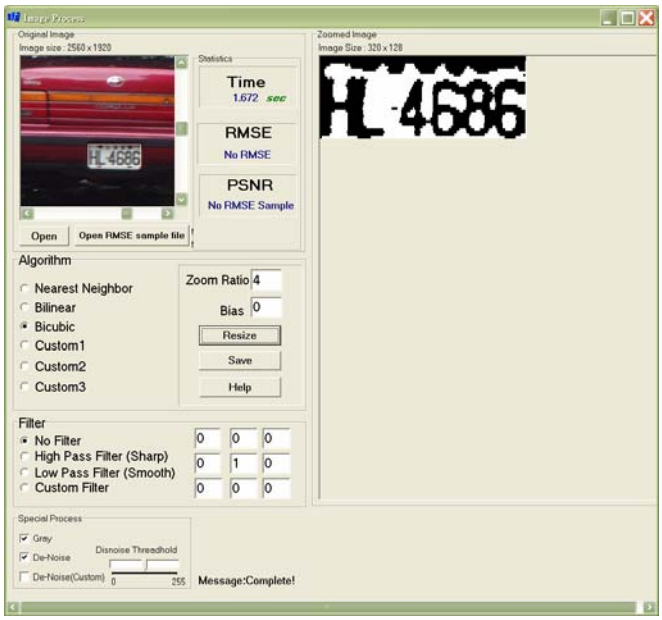
三、灰階化(Gray Scale)與去除雜點(De-Noise)

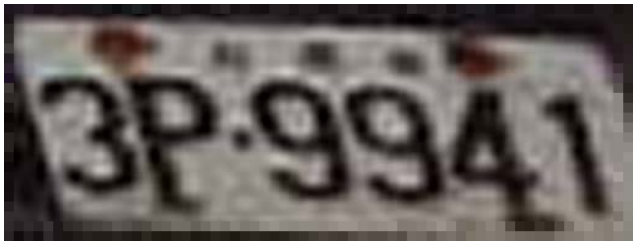
當我們處理某些特定影像時，可以對放大後的影像再加上特殊處理來強化其放大後的效果(如警方辨識車牌號碼)。爲了辨識方便，我們先將原始圖片格式由彩色(RGB)轉換成灰階。灰階值代表像素的明亮層次，其顏色範圍是從純黑(0)至純白(255)，轉換公式爲$G_i=(R_i+B_i+G_i)/3$。

爲了加強辨視的清析度，我們再將灰階影像轉換爲黑白影像，以去除雜點。轉換的方式是將灰階值大於臨界值的像素設爲白色，小於臨界值的像素則設爲黑色。而此臨界值是以整張影像的灰階平均值乘以 1/3 來當成臨界值。

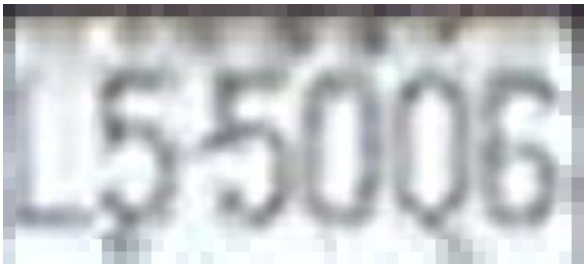
$$G(\text{threadhold})=\frac{\sum_{i=1}^{n}\frac{G_i}{n}}{3}$$

以下爲利用本專題程式將照片放大處理後的結果。





左上：原始影像(原圖 34%)
右上：局部放大
右下：去除雜訊



左上：原始影像(原圖 50%)
右上：局部放大
右下：去除雜訊

左上：原始影像(原圖 50%)
右上：局部放大
右下：去除雜訊

左上：原始影像(原圖 60%)
右上：局部放大
右下：去除雜訊

三、程式

(一)程式功能簡介 本程式目前已用 BCB(Borland C++ Builder)實作出下列功能。

1、影像縮放演算法（Resize Algorithm）：

(1) 最鄰近點演算法(Nearest Neighbor)

(2) 雙線性插值演算法(Bilinear Interpolaion)

(3) 雙三次插值演算法(Bicubic Interpolaion)

(4) 自訂方法 1(Custom 1)：

一張圖片經過二次的縮放處理，效果一定不同，因此我們突發奇想，將 Bilinear 與 Nearest Neighbor 兩種演算法合併使用。先使用 Bilinear 處理過一次，再用 Nearest Neighbor 將處理後的影像作再處理。

(5) 自訂方法 2(Custom 2)：

第二種自訂演算法，採用與第一種自訂方法類似的概念。先將影像用 Bilinear 處理過一遍；對處理後的影像中的每個像素再取其最鄰近四點（(x,y)及 (x+1,y)、(x,y+1)、(x+1,y+1)）的像素值作平均，本方法的目的爲柔化縮放後的影像。

(6) 自訂方法 3(Custom 3)：

第三種自訂演算法直接將原影像中最鄰近四點（(x,y)、(x+1,y)、(x,y+1)、(x+1,y+1)）的像素平均後當作縮放後的像素值。本方法期望以較簡單(較快)的作法，達到近似雙線性插值法的效果。

2、特殊濾鏡（Filter）：

(1) 銳利濾鏡(Sharp/High Pass Filter)

(2) 柔化濾鏡(Smooth/Low Pass Filter)

(3) 使用者自訂濾鏡(Custom Filter)：使用者可自行輸入欲使用的濾鏡參數。

3、計算 RMSE、PSNR 值：

本專題程式使用 RMSE 與 PSNR 值來評估經縮放處理後影像的品質。

(1) **RMSE(Root Mean-Square Error)**：此數值越**低**表示誤差越低，品質越好。

W：影像寬度，H：影像高度，

R'G'B'爲測試影像的 RGB 值，RGB 爲參考影像的 RGB 值。

$$\textbf{RMSE}=\sqrt{\sum_{i=1}^{W}\sum_{j=1}^{H}\frac{1}{3\times W\times H}\left(\left(R'_{i,j}-R_{i,j}\right)^2+\left(G'_{i,j}-G_{i,j}\right)^2+\left(B'_{i,j}-B_{i,j}\right)^2\right)}$$

(2) **PSNR(Peak Signal to Noise Ratio)**：此數值越**高**表示誤差越低，品質越好。

$$\textbf{PSNR}=\ 20\log_{10}\left(\frac{255}{RMSE}\right)$$

4、將影像灰階化(GrayScale)及去除雜點(De-Noise)

(二)程式執行流程

1、程式介面： 本程式分左右兩顯示區塊，左邊爲原始影像，右邊爲縮放後影像。

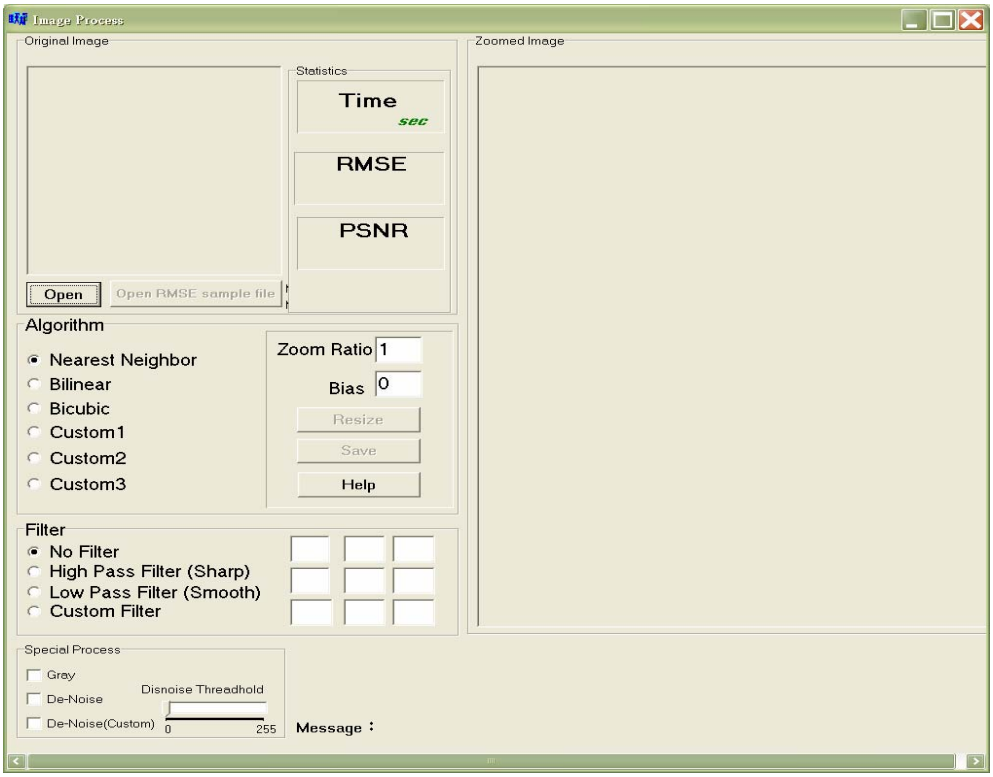


圖 24、程式介面

2、載入圖檔：按下「Open」按鈕即可選取欲處理的檔案(BMP)。

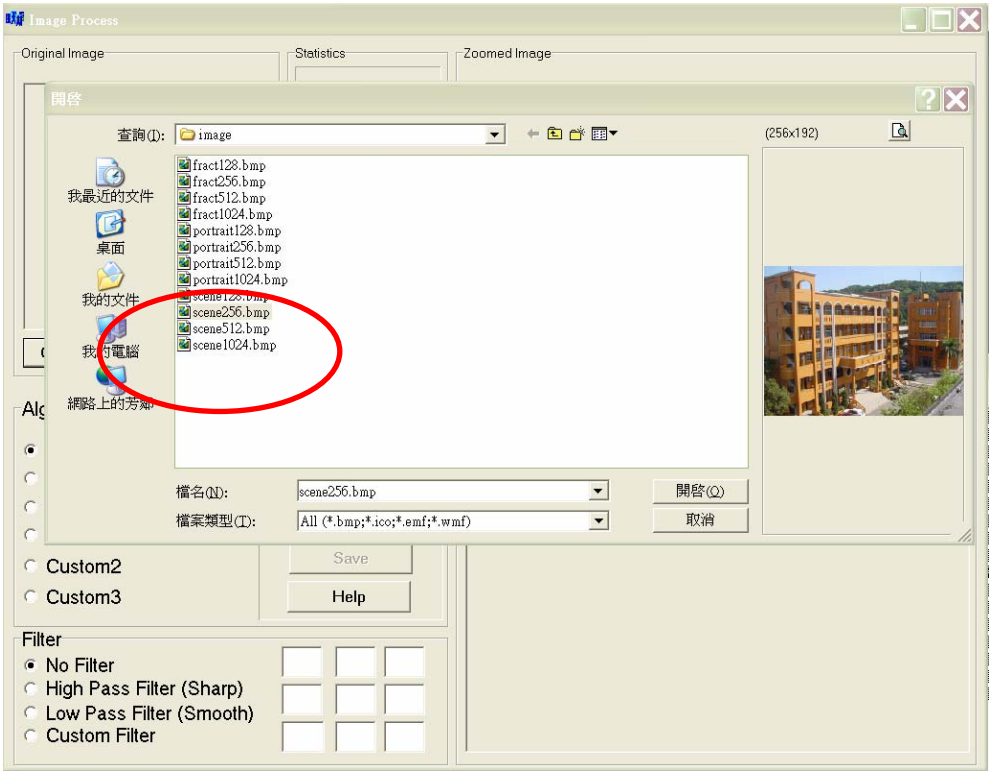


圖 25、載入圖檔

3、選取縮放演算法：共有六種縮放演算法可選用

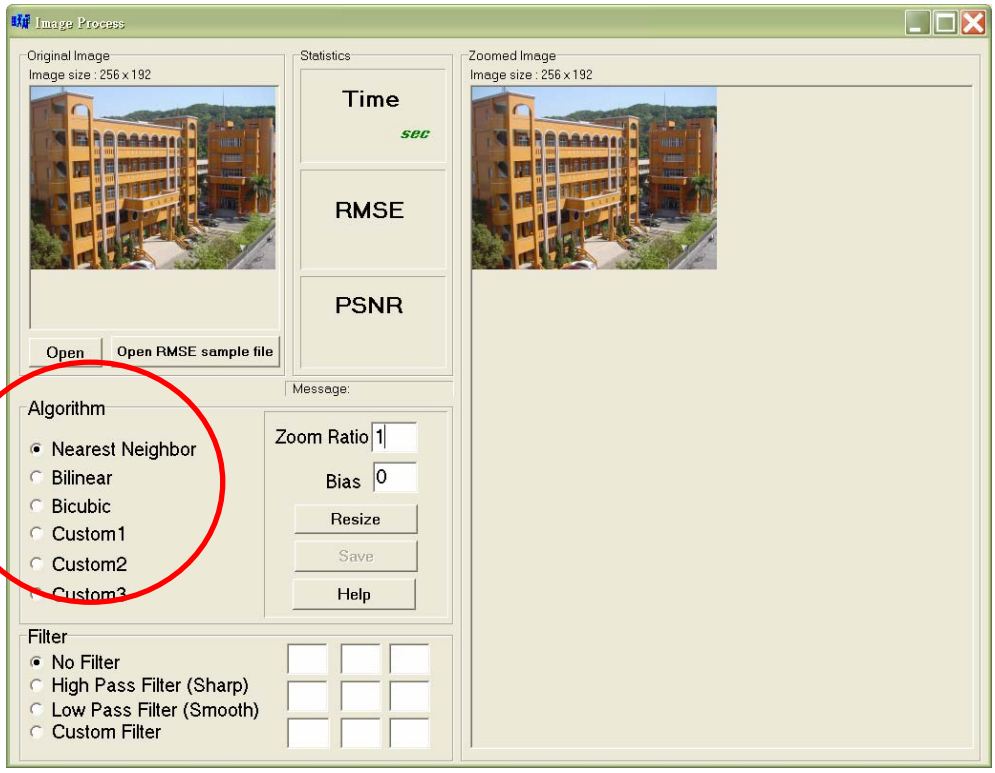


圖 26、選取演算法

4、設定縮放倍率(Zoom Ratio)及位移值(Bias)：倍率可爲放大(>1)或縮小(<1)，亦可爲小數。

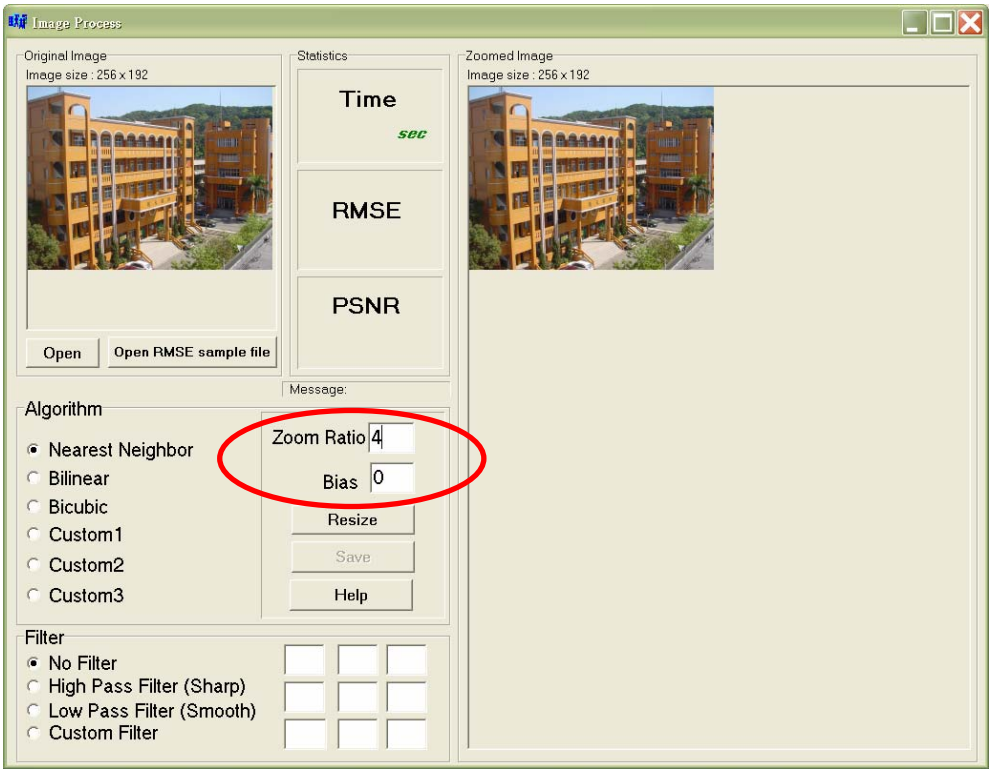


圖 27、設定縮放倍率

5、設定使用濾鏡：共有三種濾鏡可供使用。

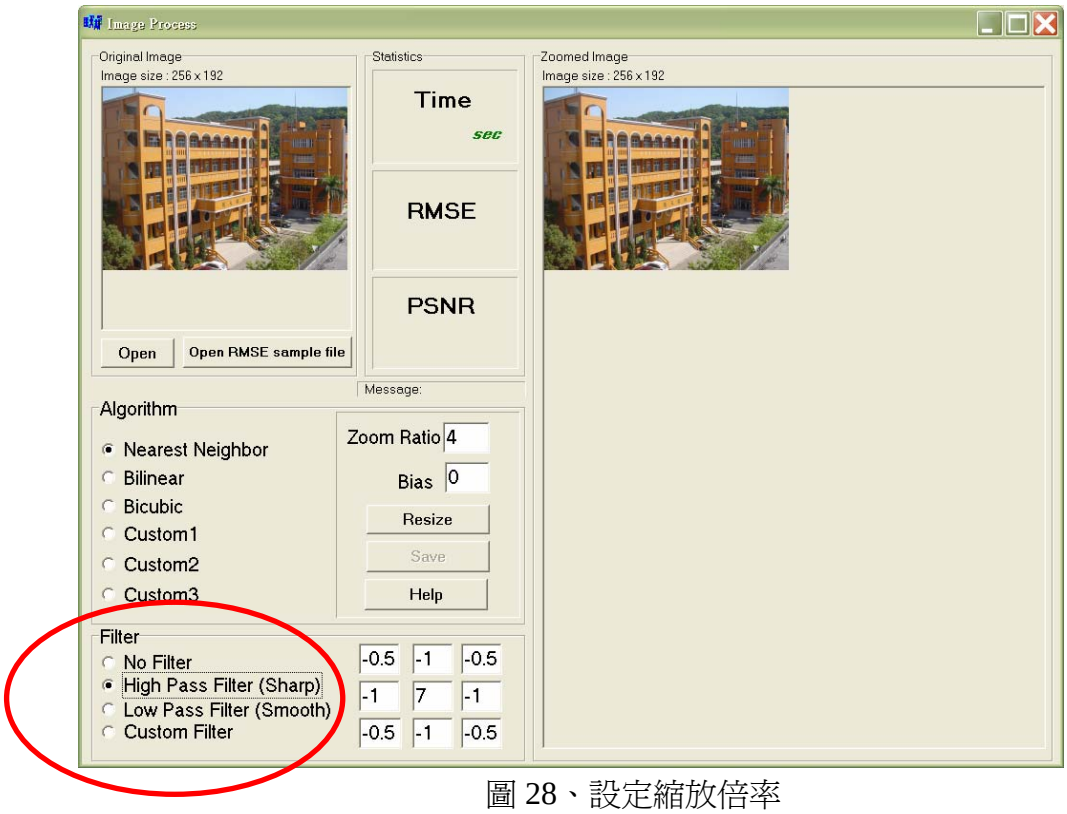


圖 28、設定縮放倍率

6、執行縮放功能：按下「Resize」按鈕即可執行縮放，並可顯示執行時間。

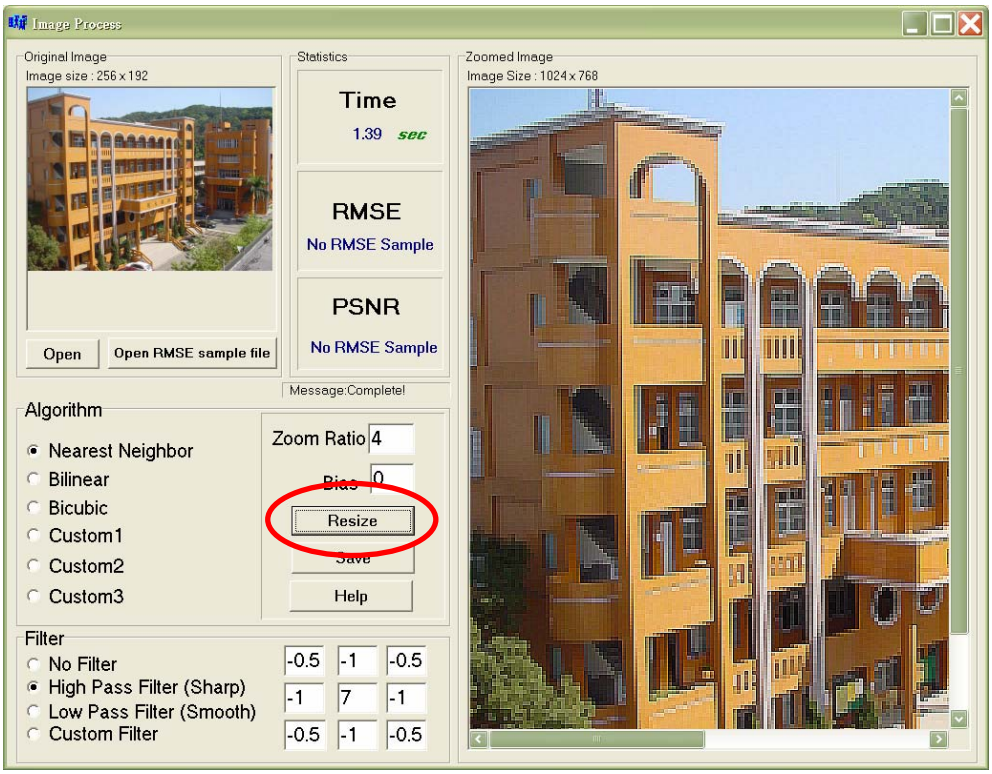


圖 28、執行縮放功能

7、若需計算 RMSE、PSNR 值，請先按下「Open RMSE sample file」按鈕，開啓計算 RMSE 所需的參考影像(解析度須同縮放後的影像)後，再按「Resize」即可。

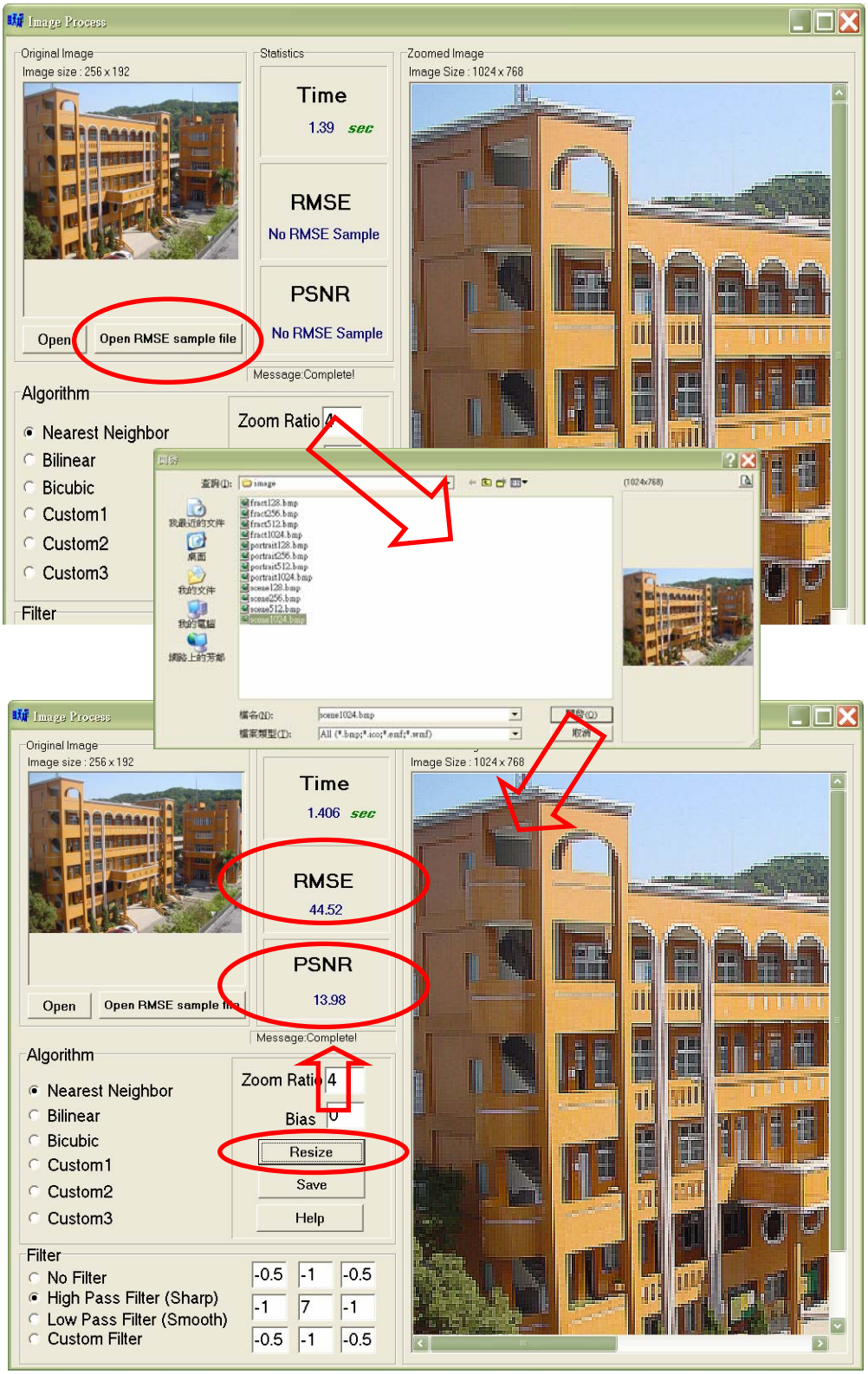


圖 29、計算 RMSE、PSNR 值

8、儲存功能：按下「Save」按鈕即可將處理後的影像存檔。

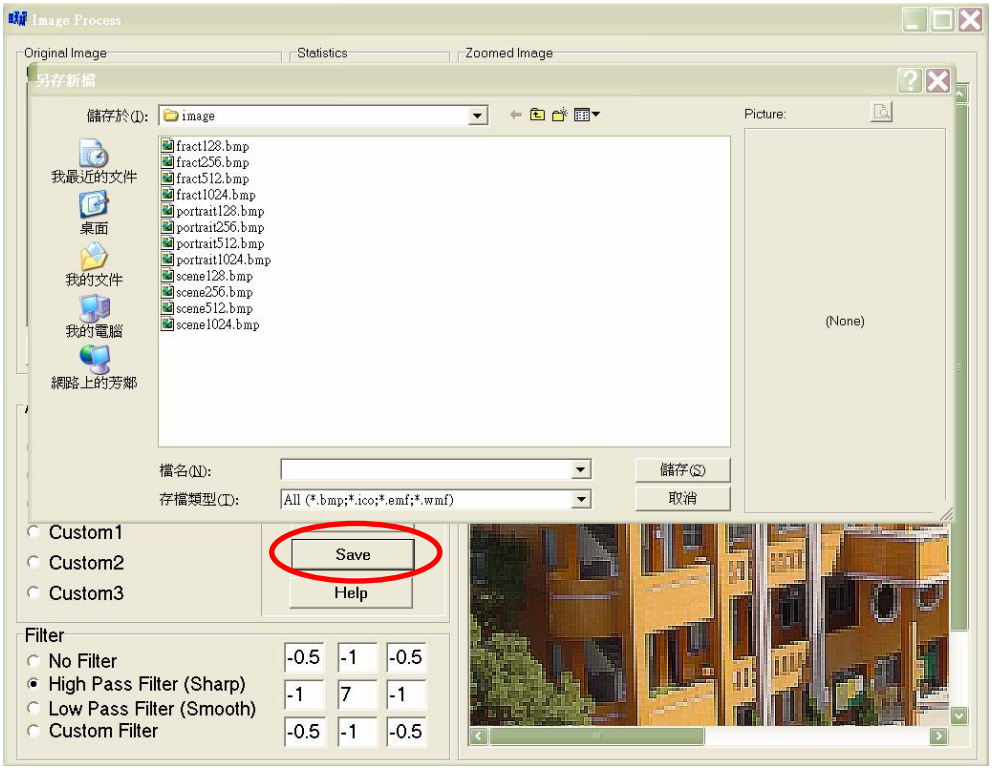


圖 30、儲存功能

9、輔助說明：按下「Help」按鈕可顯示本程式執行的輔助說明。

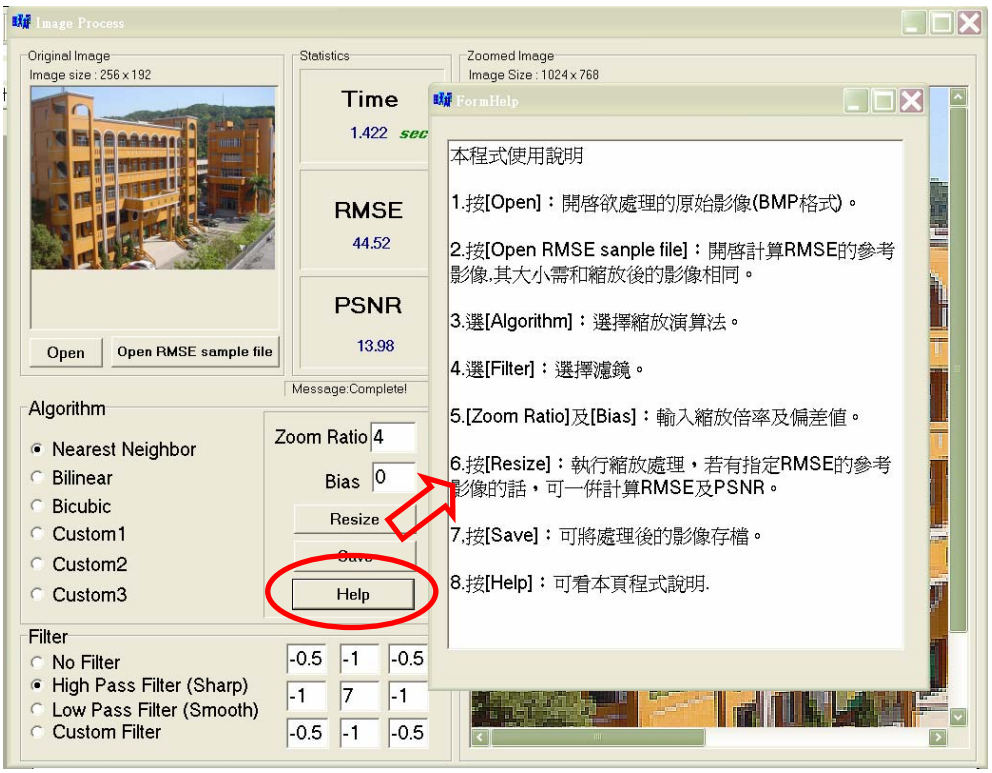


圖 31、輔助說明

陸、研究結果

一、測試影像

本次研究，我們準備了以下三種不同類型的影像供測試之用：

1.紋路細緻、色彩多變的影像**(影像細節較多)**：

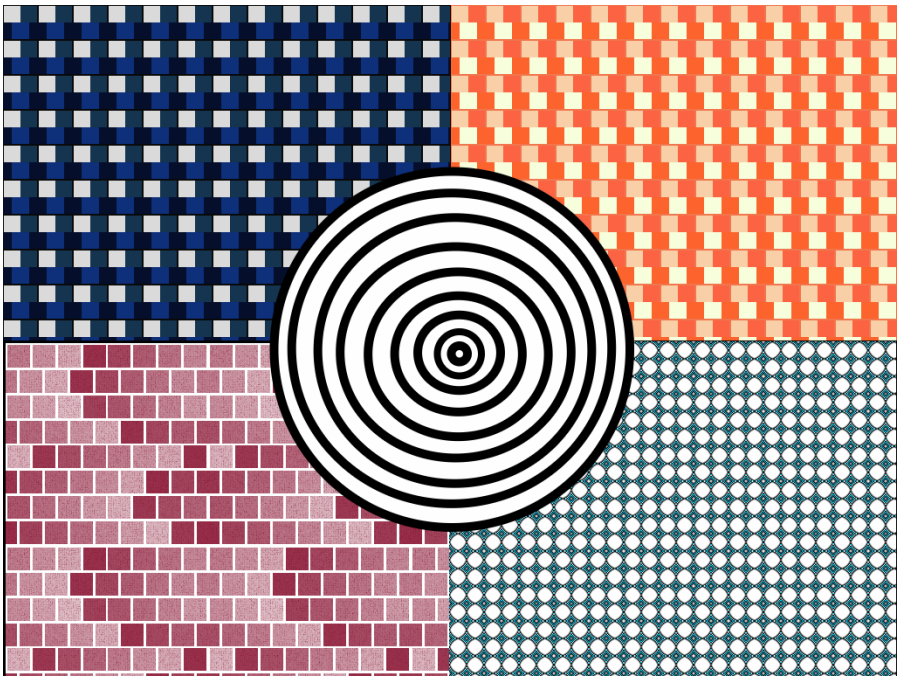


圖 32

2.線條狀的風景圖、建築物、文字**(影像細節適中)**：

圖 33

3.塊狀面積大、顏色漸層均勻的影像，如人像**(影像細節較少)**：

圖 34

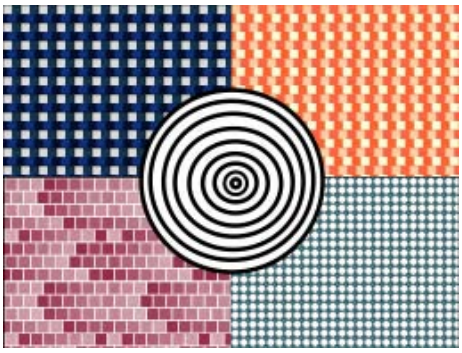
二、測試環境

本實驗在個人電腦上測試，其配備如下：
CPU：Intel Pentium 4　3.0GHz
RAM：768MB
作業系統：WindowsXP

三：測試數據

以上三種影像，經過我們的程式實際執行，所得的數據如下：

1、紋路細緻、色彩多變的影像(影像細節較多)：



演算法	RMSE	PSNR	Time(sec)
Nearest Neighbor	43.58	13.98	0.172
Bilinear	41.96	13.98	0.781
Bicubic	40.36	13.98	1.828
Custom1	41.96	13.98	0.859
Custom2	40.96	13.98	1.281
Custom3	46.31	12.04	0.563

表 1、圖片大小由 256x192→1024x768(放大 4 倍) ，不加濾鏡處理。

濾鏡	RMSE	PSNR	Time(sec)
High Pass Filter (Sharp)	40.29	13.98	1.562
Low Pass Filter (Smooth)	28.55	18.06	1.531

表 2、圖片大小(1024x768)不進行縮放

演算法	濾鏡	RMSE	PSNR	Time(sec)
Nearest Neighbor	High Pass Filter(Sharp)	72.25	9.54	1.500
	Low Pass Filter(Smooth)	40.21	13.98	1.531
Bilinear	High Pass Filter(Sharp)	43.32	13.98	2.187
	Low Pass Filter(Smooth)	42.64	13.98	2.203
Bicubic	High Pass Filter(Sharp)	42.43	13.98	3.312
	Low Pass Filter(Smooth)	40.83	13.98	3.313
Custom1	High Pass Filter(Sharp)	43.32	13.98	2.266
	Low Pass Filter(Smooth)	42.64	13.98	2.250
Custom2	High Pass Filter(Sharp)	39.58	13.98	2.703
	Low Pass Filter(Smooth)	41.87	13.98	2.734
Custom3	High Pass Filter(Sharp)	76.91	6.02	1.953
	Low Pass Filter(Smooth)	43.86	13.98	1.969

表 3、圖片大小由 256x192→1024x768(放大 4 倍)，再加上濾鏡處理。

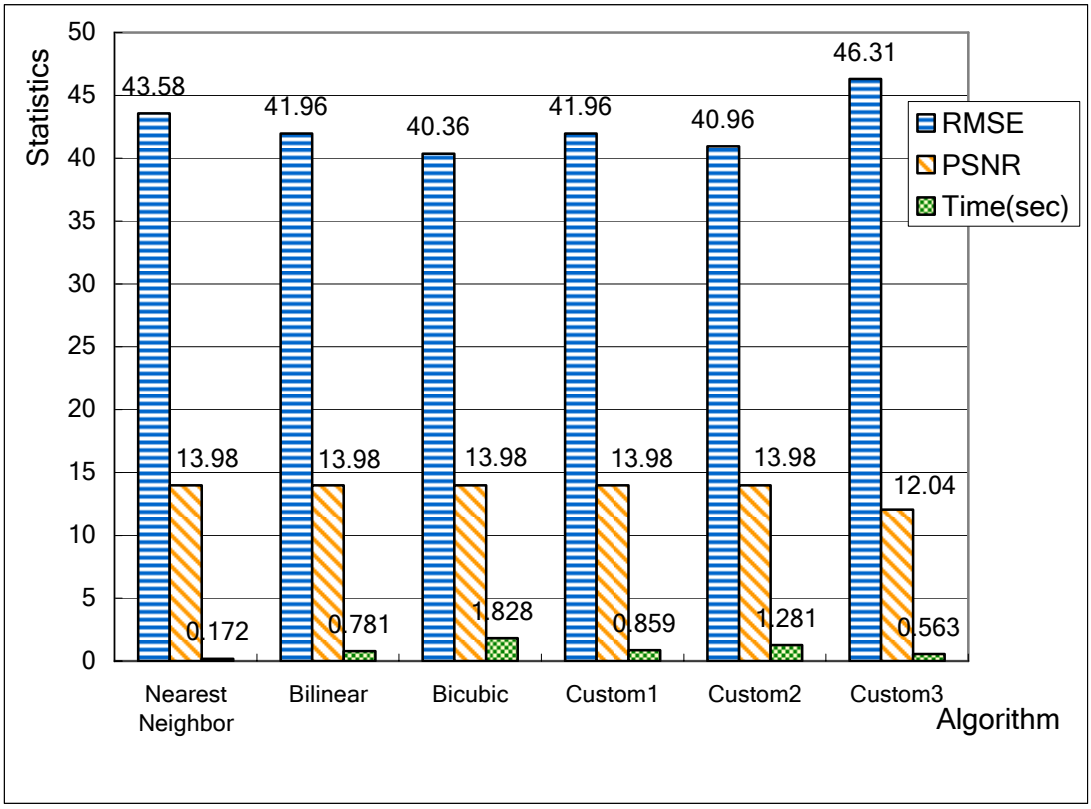


表 4、圖片大小由 256x192→1024x768(放大 4 倍) ，不加濾鏡處理。

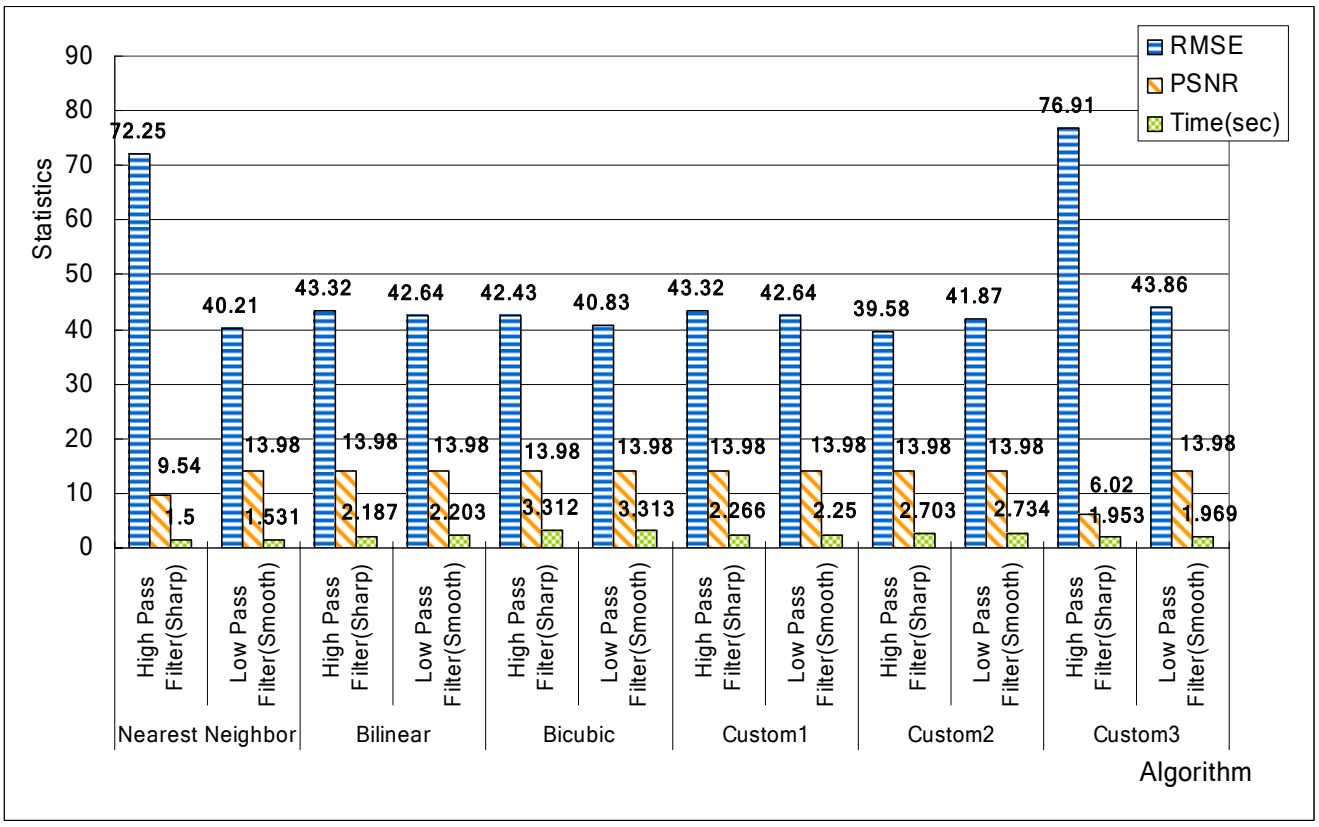


表 5、圖片大小由 256x192→1024x768(放大 4 倍)，再加上濾鏡處理。

2、線條狀的風景圖、建築物、文字(影像細節適中)：

演算法	RMSE	PSNR	Time(sec)
Nearest Neighbor	18.34	21.58	0.172
Bilinear	18.13	21.58	0.766
Bicubic	17.46	22.28	1.843
Custom1	18.13	21.58	0.844
Custom2	17.68	22.28	1.266
Custom3	20.06	20.83	0.563

表 6、圖片大小由 256x192→1024x768(放大 4 倍)，不加濾鏡處理。

濾鏡	RMSE	PSNR	Time(sec)
High Pass Filter (Sharp)	41.73	13.98	1.515
Low Pass Filter (Smooth)	10.38	26.85	1.5

表 7、圖片大小(1024x768)不縮放

演算法	濾鏡	RMSE	PSNR	Time(sec)
Nearest Neighbor	High Pass Filter(Sharp)	43.68	13.98	1.515
	Low Pass Filter(Smooth)	17.21	22.28	1.516
Bilinear	High Pass Filter(Sharp)	20.27	20.83	2.203
	Low Pass Filter(Smooth)	18.45	21.58	2.188
Bicubic	High Pass Filter(Sharp)	20.35	20.83	3.328
	Low Pass Filter(Smooth)	17.61	22.28	3.266
Custom1	High Pass Filter(Sharp)	20.27	20.83	2.234
	Low Pass Filter(Smooth)	18.45	21.58	2.250
Custom2	High Pass Filter(Sharp)	17.60	22.28	2.719
	Low Pass Filter(Smooth)	18.12	21.58	2.718
Custom3	High Pass Filter(Sharp)	41.11	13.98	1.968
	Low Pass Filter(Smooth)	19.19	20.83	1.954

表 8、圖片大小由 256x192→1024x768(放大 4 倍) ，再加上濾鏡處理。

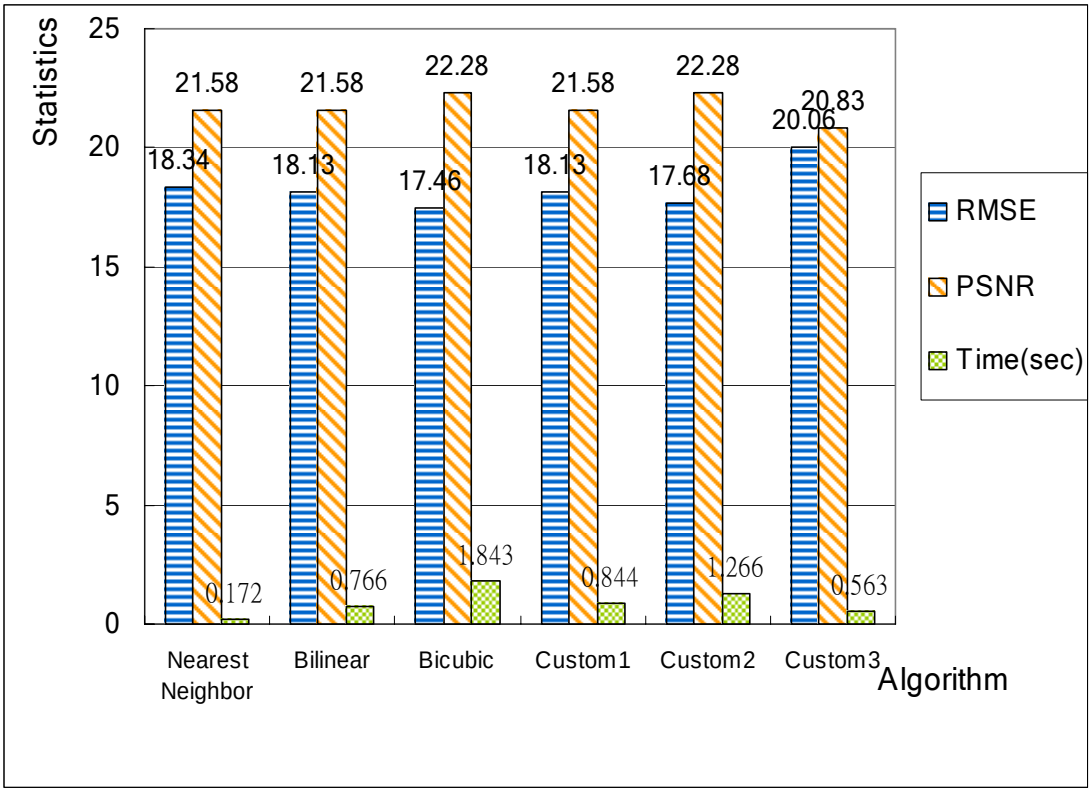


表 9、圖片大小由 256x192→1024x768(放大 4 倍)，不加濾鏡處理。

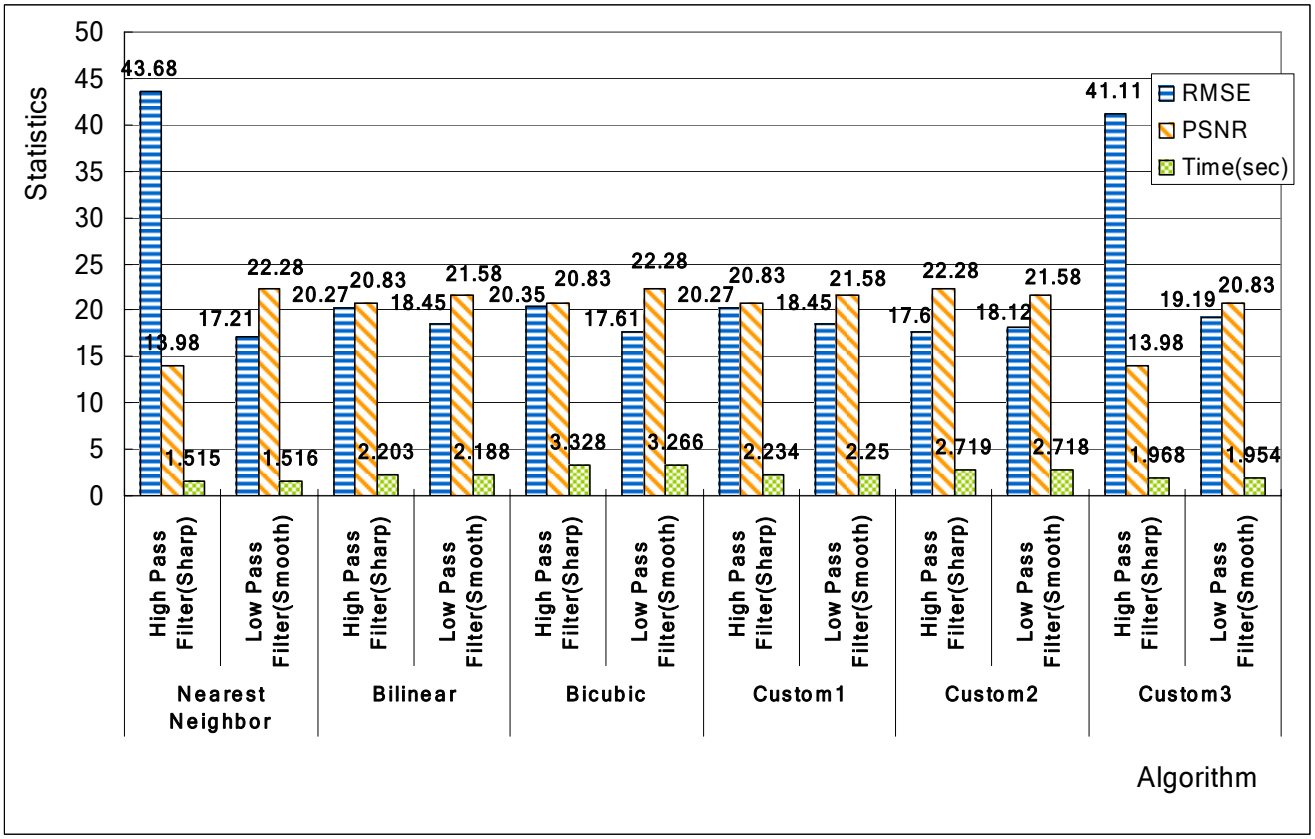


表 10、圖片大小由 256x192→1024x768(放大 4 倍) ，再加上濾鏡處理。

3、塊狀面積大、顏色漸層均勻的影像(影像細節較少)：

演算法	RMSE	PSNR	Time(sec)
Nearest Neighbor	12.67	25.11	0.172
Bilinear	11.95	26.02	0.781
Bicubic	12.19	25.11	1.828
Custom1	11.95	26.02	0.859
Custom2	11.91	26.02	1.281
Custom3	13.79	24.61	0.562

表 11、圖片大小由 192x256→768x1024 (放大 4 倍)，不加濾鏡處理。

濾鏡	RMSE	PSNR	Time(sec)
High Pass Filter (Sharp)	28.86	18.06	1.532
Low Pass Filter (Smooth)	6.36	31.36	1.531

表 12、圖片大小(768x1024)不縮放

演算法	濾鏡	RMSE	PSNR	Time(sec)
Nearest Neighbor	High Pass Filter(Sharp)	28.92	18.06	1.547
	Low Pass Filter(Smooth)	11.84	26.02	1.562
Bilinear	High Pass Filter(Sharp)	12.84	25.11	2.187
	Low Pass Filter(Smooth)	12.21	25.11	2.203
Bicubic	High Pass Filter(Sharp)	14.92	24.08	3.297
	Low Pass Filter(Smooth)	12.10	25.11	3.328
Custom1	High Pass Filter(Sharp)	12.83	25.11	2.281
	Low Pass Filter(Smooth)	12.21	25.11	2.281
Custom2	High Pass Filter(Sharp)	11.71	26.02	2.750
	Low Pass Filter(Smooth)	12.22	25.11	2.734
Custom3	High Pass Filter(Sharp)	26.74	18.06	1.984
	Low Pass Filter(Smooth)	13.16	24.61	1.985

表 13、圖片大小由 192x256→768x1024 (放大 4 倍) ，再加上濾鏡處理。

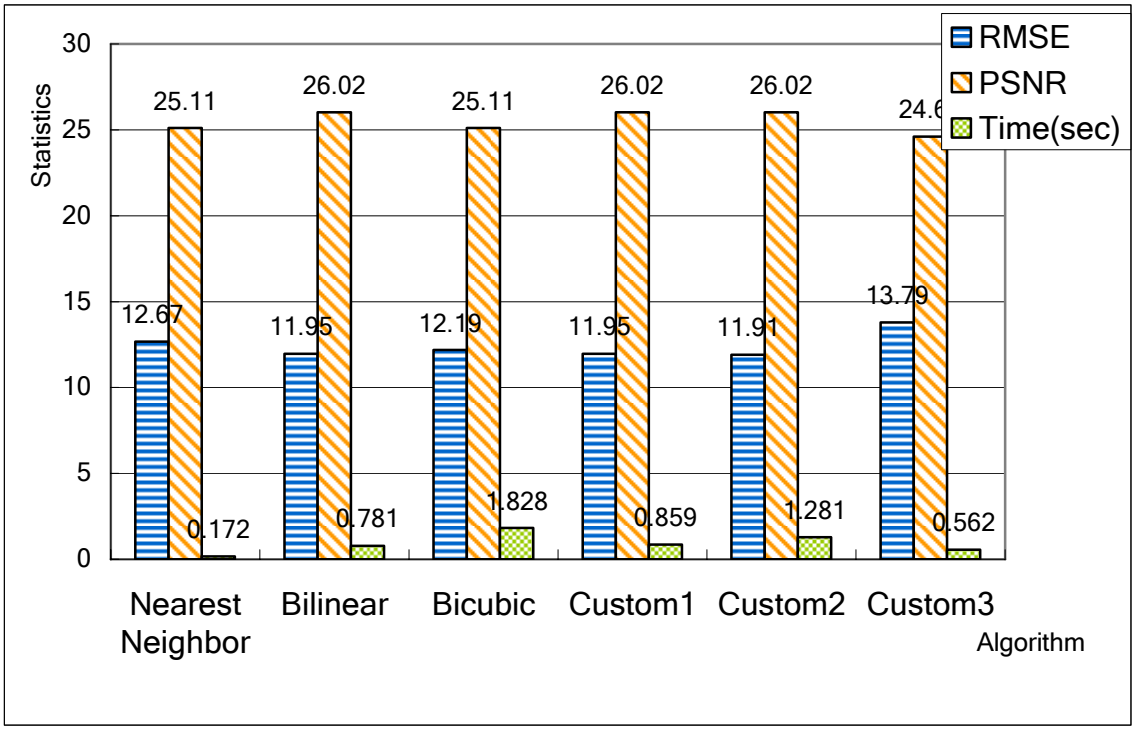


表 14、圖片大小由 192x256→768x1024 (放大 4 倍)，不加濾鏡處理。

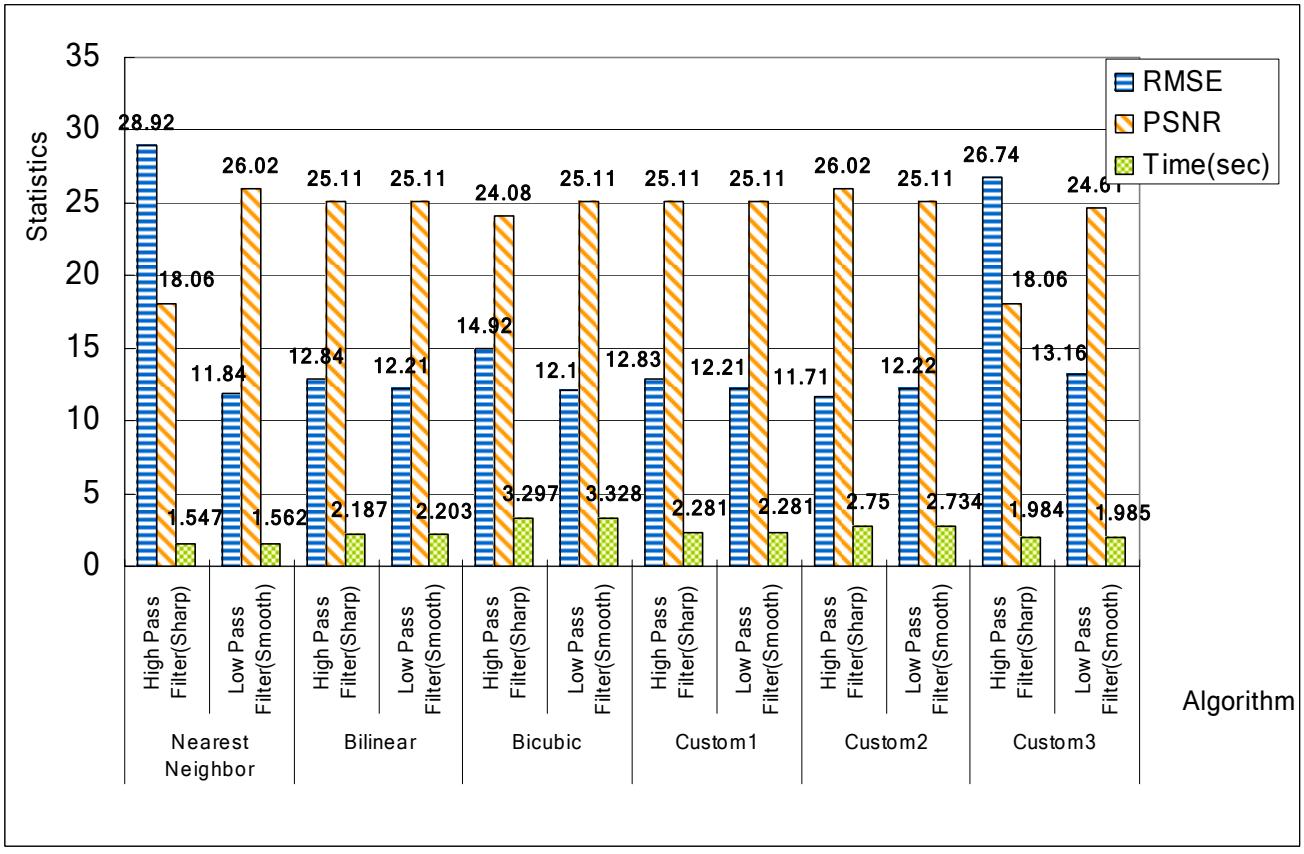


表 15、圖片大小由 192x256→768x1024 (放大 4 倍) ，再加上濾鏡處理。

柒、討論與結論

一、經由觀察發現，經由 Bilinear 與 Bicubic(含 Custom1、2、3)演算法所放大後的影像會產生位移偏差(Bias)。而此位移偏差值會影響 RMSE 的計算準確度(影響的幅度甚巨)。故本程式另外提供使用者調整影像位移偏差的功能，經實驗後發現放大 4 倍時 Bilinear 演算法(含 Custom1、2、3)需將偏差值調整爲 2，而 Bicubic 演算法則需將偏差值調整爲 1，如此計算出來的 RMSE 值才會正確。

二、以本專題所測試的三種影像來看，不管使用任一種演算法，可以發現細節越多的影像其放大後的效果越不理想，不論是視覺上的效果或是相關數據(RMSE、PSNR)都是如此。故這類影像的處理仍有很大改善的空間。

三、針對三種縮放演算法(Nearest Neighbor、Bilinear、Bicubic)的測試結果比較如下：不論以肉眼直接觀察，或比較測試數據（RMSE、PSNR），其結果均爲 Bicubic 優於 Bilinear，Bilinear 優於 Nearest Neighbor(僅第 3 個人像影像例外，Bilinear 略優於 Bicubic，推測原因爲此影像連續色調的範圍較大，而 Bilinear 所造成的模糊效果恰好較適於此類影像)。但 Bicubic 演算法的缺點爲所需時間較長，若處理大影像時所需的時間差距會更大。

四、針對自訂的三種縮放演算法的測試結果分析如下：

(一) 自訂方法一的品質與 Bilinear 演算法近似，但時間稍長。

(二) 自訂方法二的影像品質及數據接近 Bicubic 演算法，且時間較 Bicubic 演算法來的小，故可考慮用此演算法來取代 Bicubic 演算法。

(三) 自訂方法三的品質則較不理想，可看到類似 Nearest Neighbor 演算法的鋸齒狀，且數據及時間均不如 Nearest Neighbor 演算法，與原本設計此演算法時預估的有所差異，故不推薦此方法。

五、關於濾鏡的處理部份，適當地銳利化處理原本模糊的影像(如 Bilinear 演算法)，或柔化原本有鋸齒狀/馬賽克的影像(如 Nearest Neighbor 演算法)，就視覺效果而言能讓一般人感覺有明顯的差異(大部份是正面的影響)。但就實驗數據來看，在大部份的情形下銳利濾鏡和柔化濾鏡則都對數據造成負面的影響(例外情形：Nearest Neighbor 演算法加柔化濾鏡、Custom2 演算法加銳利濾鏡、Custom3 演算法加柔化濾鏡)，而其中又以細節較多的影像加上銳利濾鏡時所造成的數值負面影響最大。

六、承上例外情形之原因推測如下：

(一)、Nearest Neighbor 演算法較易產生鋸齒影像，故加上柔化濾鏡後有正

面的提升。

(二)、Custom2 演算法因爲實際上是 Bilinear 演算法加上均化(Average)的混合演算法，故加上銳利濾鏡後有正面的提升。

(三)、Custom3 演算法較易產生鋸齒狀(類似 Nearest Neighbor 演算法) ，故加上柔化濾鏡後有正面的提升。

七、關於銳利濾鏡的補充：本程式所採用的銳利濾鏡其效果並非適合每一張影像。因此我們利用自訂濾鏡的功能測試歸納出以下幾種不同的銳利濾鏡，使用者可以自行選擇合適的濾鏡。

0	-1	0
-1	5	-1
0	-1	0

此濾鏡銳利程度較緩和

-1	-1	-1
-1	9	-1
-1	-1	-1

此濾鏡銳利程度較強

-10	-10	-10
-10	79	-10
-10	-10	-10

此濾鏡可得到影像的輪廓線

八、雖然本次專題並未研究出較具突破性的的影像縮放方法，但也讓我們對各種數位影像處理的觀念與技巧方法有了較深入的了解，並能歸納出各種演算法的特性來加以應用。此外 Borland C++ Builder 這套開發工具強大的功能亦令我們印像深刻。以前我們僅會使用一些套裝的影像處理軟體（如：PhotoImpact、Photoshop、S-spline（PhotoZoom）、PhotoCap……等），在經過本專題的過程後，如今我們對於軟體的開發與影像處理均已有了初步的認知，相信未來在做類似的開發研究上會更加得心應手！

捌、未來展望

「影像無失真縮放」的境界，一直是眾人追求的理想和目標。對於身爲學生的我們而言，並沒有因爲科技的進步，而有能力購置高倍率的數位相機。因此，我們嘗試以現有知識，設計出本專題程式，希望能將現有的影像品質作進一步地提升。本專題程式的功能或許不足以媲美市面上的套裝軟體，但應已可以應付一般的需求。在應用方面，如目前的照相手機其像素值並不高，或許加上適當的影像處理的功能，就不必再爲了低像素而不佳的效果而傷腦筋。以本專題中我們對

於影像結構、像素資料的存取所累積的知識，若能再配合影像處理相關知識的精進，在未來我們可以更進一步朝向動態(即時)影像縮放與影像輪廓分析辨識的應用領域來發展。

玖、參考資料及其他:

1. 葉秋城、王文豪，影像處理與分析，碁峰資訊，1995
2. 施威銘、李青亮等四位，高中電腦，旗立出版社，2005/3
3. 洪國勝等四位，C++ Builder 6 程式設計快樂上手，旗標出版社，2002/8/20
4. 施威銘主編，數位相片編修聖經，旗標出版股份有限公司，2005/2/3
5. 陳映中，2000/6，類神經網路漸進式影像傳輸方法，國立中山大學機械工程研究所碩士論文
6. 江偉傑，2004/6，數位影像法於變形測量，國立雲林科技大學機械工程系碩士論文
7. 林達德，數位影像處理基本原理，國立台灣大學生物機電系相關論述
8. 黃一民，2003/7/15，超解析技術用於影像序列放大之研究，國立成功大學碩士論文
9. 杰克網電腦繪圖教室，http://www.ctk.com.tw/jackweb/graphic/
10. 洪燕竹的Homepage，http://andrew.csie.ncyu.edu.tw/Doc3A.htm
11. 影像縮小處理(縮小処理のナゾ)，http://www.nnet.ne.jp/~hi6/lab/resize/
12. Bicubic Interpolation for Image Scaling，http://astronomy.swin.edu.au/~pbourke/colour/bicubic/
13. Comp.Graphics.Algorithms Frequently Asked Questions Introduction and Contents，http://exaflop.org/docs/cgafaq/
14. CompuPhase，http://www.compuphase.com/graphic/scale1errata.htm
15. Dr.Dobb's Software Tools for the Professional Programmer，http://www.ddj.com/articles/2002/0205/
16. efg's Image Processing，http://www.efg2.com/Lab/Library/ImageProcessing/Algorithms.htm
17. EGES510:Final Project，Image Restoration Using Bilinear Interpolation，Gwyneth Holston，2004/11/10
18. Image Scaling with Bresenham ，Dr.Dobb Jounal.，2002/5
19. Pontus Rendah，http://www.iue.it/Personal/Researchers/pontus/work.htm
20. The Java Developers Almanac 1.4，http://javaalmanac.com/egs/java.awt/TransformImage.html
21. XtremeDSP，Developing Image Processing，Daniel E.Michek，2004

中華民國第四十五屆中小學科學展覽會

評　　語

高中組 生活與應用科學科

最佳團隊合作獎

040811

你可以再靠近一點－數位影像縮放之研究

國立桃園高級中學

評語：

團隊合作默契甚佳。作品所採用之方法及圖例，非常豐富。建議事項包括(1)可以就不同應用向度，歸納最佳影像處理方法，並以統一且具代表性之圖例示範，效果將會更佳；(2)資料的表現方式及結論的淬取，若能再加以深入且活潑化，將可以不辜負豐富的內容，及長期的努力成效。