

# 國立交通大學

電機學院 IC 產業研發碩士班

## 碩士論文

影像縮放內插方法及硬體化可行性之研究

Study of Interpolation Methods for Image Scaling and Possible Hardware  
Implementation



研究生：高文淵

指導教授：莊仁輝 教授

中華民國九十六年一月

# 影像縮放內插方法及硬體化可行性之研究

## Study of Interpolation Methods for Image Scaling and Possible Hardware Implementation

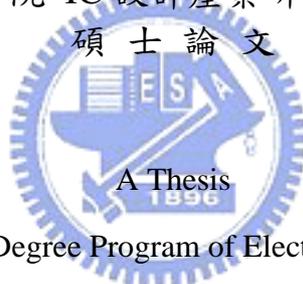
研究生：高丈淵

Student : Chang-Yuan Kao

指導教授：莊仁輝

Advisor : Jen-Hui Chuang

國立交通大學  
電機學院 IC 設計產業研發碩士班  
碩士論文



Submitted to Degree Program of Electrical Engineering

National Chiao Tung University  
in partial Fulfillment of the Requirements  
for the Degree of  
Master  
in

Industrial Technology R & D Master Program of  
Electrical and Computer Engineering College

Jan 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年一月

# 影像縮放內插方法及硬體化可行性之研究

學生：高丈淵

指導教授：莊仁輝 博士

國立交通大學電機學院產業研發碩士班

## 摘 要

本論文研討之影像縮放內插演算法及其可行之硬體化技術乃近年來影像科技發展之趨勢所需。不論是在日常家庭生活早已佔有不可或缺地位的電視機或是辦公室常見的電腦顯示器，已在沉默中由數位薄型螢幕取代了傳統的類比陰極射線管(CRT)顯示器；而人手一部的行動電話，亦拜平板顯示技術進步所賜，其功能早已進展到可播放數位影像。然而如何在這些大小尺寸不一的顯示媒體上顯示同一解析度的影像或是在一固定解析度顯示器上顯示不同解析度圖像訊號便成為了一個重要的研究議題。

理論上，在有限頻寬信號分析的研究中，應用 Sinc 函數作內插是最理想的內插選擇，運用到影像縮放處理時，則是將影像看成二維的信號，分別從水平方向及垂直方向進行內插。然而因其函數特性過於複雜而使其無法在硬體化實現上得到經濟的實作方式。一般來說，對於影像以內插方式增加或降低其解析度基本常用的內插核心不外乎有最鄰近像素內插(Nearest Neighbor)、雙線性(Bi-Linear)內插及雙立方(Bi-Cubic)內插。其中又以雙立方內插核心函數之特性與 Sinc 最為接近。

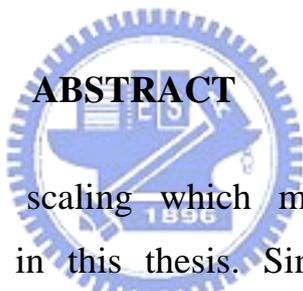
我們選擇雙立方內插核心函數為我們硬體實作之研究，理由除了其函數特性外，我們也以大量的圖像做為分析演算法之優劣的基礎，我們所得到的數據亦支持了學理上的說法。在影像縮放內插演算法之硬體化可行性之研究上，我們以硬體實作首要考量之經濟及處理速度為出發點，設計了查表式的影像縮放硬體架構，並針對所需解晰度轉換所需之查表容量做最佳化研究，以達經濟及處理速度之目的，其中對於內插函數權值的調整是目前已公開之文獻中未曾探討的領域。

# Study of Interpolation Methods for Image Scaling and Possible Hardware Implementation

**student : Chang-Yuan Kao**

**Advisors : Dr. Jen-Hui Chuang**

Industrial Technology R & D Master Program of  
Electrical and Computer Engineering College  
National Chiao Tung University

The logo of National Chiao Tung University is a circular emblem with a gear-like border. Inside the circle, there is a stylized figure holding a torch, with the year '1959' at the bottom. The word 'ABSTRACT' is overlaid in bold black text across the center of the logo.

## ABSTRACT

Effective procedures for image scaling which may be suitable for hardware implementation are investigated in this thesis. Since the display technology is changing rapidly from analog to digital and CRT to flat panel display, there are often various resolution, in display specifications in our daily life, such as these for huge size TV or hand-held devices, such as cell phone. How to display an image with satisfactory in different display devices with different resolutions become big issue.

In this thesis, we study the image scaling technique, starting from on the interpolation kernel "Sinc" that signal processing theory regarded as "ideal," then other common methods like "Nearest Neighbor," "Bi-Linear" and "Bi-Cubic." Subsequently, the "Bi-Cubic" interpolation kernel for possible hardware implementation, not only because it is closer to "ideal" mathematically, but also supported by experimental data. We implement the hardware architecture by table-look-up approach and also discuss different ways of optimizing the table size.

# 致謝

本篇研究論文得以順利完成，期間經歷了相關文獻研究、研究工具選定、實驗之方法學習、與論文內容的編寫及修正。使本人在這兩年的學習過程中，深刻感受到在學術研究這條路上之艱辛以及需要耐心來克服種種阻礙。兩年的學習使個人受益非淺，首先個人要鄭重感謝指導教授 莊仁輝 博士，在個人的學習及研究期間，時刻給予學生的指導及支持，同時也感謝口試委員 韓欽銓 教授及 王才沛 教授的寶貴意見，使個人可在本論文之不足處得補強，讓本論文可以更加完整地完成。

在這兩年的碩士學習生涯中，亦感謝實驗室學長高肇宏及林泓宏在學業上的協助，使個人能有更加正確的研究態度。此外，更要感謝在這兩年來默默陪伴我、支持我的好朋友，是你給我信心及鼓勵，讓我再一次的學生生涯添加許多生活上的樂趣。最後要感謝我的父母，因為您們的支持，我才能無後顧之憂地會再次體驗學生生活，並取得碩士學位，感謝所有曾經幫助過我的朋友。



# 目錄

中文摘要.....	I
英文摘要.....	II
致謝 .....	III
目錄 .....	IV
表目錄.....	VI
圖目錄.....	VII
一、 緒論.....	1
1.1 簡介.....	1
1.2 相關文獻探討.....	2
1.2.1 專利文獻.....	3
1.2.2 適合硬體實作之相關文獻.....	3
1.2.3 較不適合硬體實作之論文研究.....	4
1.3 論文章節簡介.....	5
二、 影像內插演算法.....	7
2.1 影像內插縮放概述.....	7
2.2 演算法比較.....	10
2.2.1 影像縮放演算法之核心函數.....	10
2.2.2 Sinc 內插核心演算法 .....	11
2.2.3 Nearest Neighbor 內插核心演算法 .....	14
2.2.4 Linear/Bi-Linear 內插核心演算法 .....	16
2.2.5 Cubic/Bi-Cubic 內插核心演算法 .....	18
2.3 二維度 Cubic 內插放大演算法 .....	22
2.4 邊界處理.....	24
2.5 RGB/YUV 色域空間的選擇與比較.....	26
2.6 實驗比較結果.....	29
三、 Bi-Cubic 內插運算之硬體實作探討 .....	44
3.1 Fixed Point Model 之設計.....	44
3.2 Look-up Table 之建構.....	44
3.3 Look-up Table 容量大小 .....	46
3.4 Fixed Point Model 之內插運算.....	46
3.5 內插係數和之探討.....	49
3.6 數值運算精確度探討.....	51
四、 影像縮小.....	59
4.1 影像縮小觀念說明.....	59

4.2	Bi-Cubic 影像縮小 .....	61
4.3	Local Average 影像縮小 .....	63
4.4	實驗結果.....	68
五、	討論.....	73
5.1	所採用方法的總結.....	73
5.2	未來展望.....	73
	參考文獻.....	75
	附錄一.....	77



# 表目錄

表 2-1 以邊框與文字影像作不同演算法之放大誤差統計表 .....	33
表 2-2 以 Master 影像作不同演算法之放大誤差統計表 .....	34
表 2-3 以風景影像作不同演算法之放大誤差統計表 .....	35
表 2-4 以動物影像作不同演算法之放大誤差統計表 .....	36
表 2-5 實驗測試影像之比較表 .....	42
表 2-6 內插補點函數核心在其處理後影像之效果差異及在軟體及硬體實作上的 考量.....	43
表 3-1 Fixed-point 與 WS-A 權重(Weight)比較表格 .....	51
表 3-2 以灰階人物影像(圖 3-5)作 Fixed Point 與 Floating Point 放大之差值統計表 .....	53
表 3-3 以彩色風景影像(圖 3-6)作 Fixed Point 與 Floating Point 放大之差值統計表 .....	54
表 3-4 以彩色建築物影像(圖 3-7)作 Fixed Point 與 Floating Point 放大之差值統計 表.....	55
表 3-5 以彩色動物影像(圖 3-8)作 Fixed Point 與 Floating Point 放大之差值統計表 .....	56
表 3-6 以文字模式影像(圖 3-9)作 Fixed Point 與 Floating Point 放大之差值統計表 .....	57
表 3-7 以主要測試影像(圖 3-10)作 Fixed Point 與 Floating Point 放大之差值統計 表.....	58
表 4-1 自然風景影像之縮小誤差統計表 .....	69
表 4-2 主要測試影像之縮小誤差統計表 .....	70
表 4-3 環形條紋影像之縮小誤差統計表 .....	71
表 4-4 動物影像之縮小誤差統計表 .....	72

# 圖目錄

圖 2-1	三倍內插重新取樣 .....	8
圖 2-2	一維度線性內插補點 .....	10
圖 2-3	Sinc 函數 .....	12
圖 2-4	以 Sinc 函數為內插核心所內插出之函數圖形示意圖 .....	12
圖 2-5	波紋效應 (Ringing Effect) .....	13
圖 2-6	Sinc 函式作二維影像內插放大結果 .....	14
圖 2-7	Nearest Neighbor 函數 .....	15
圖 2-8	以 Nearest Neighbor 函數內插出之圖 2-1(a) 函數圖形 .....	15
圖 2-9	Nearest Neighbor 作一維影像內插結果 .....	16
圖 2-10	Linear(線性)函數 .....	17
圖 2-11	以 Linear 函數內插出之圖 2-1(a) 函數圖形 .....	17
圖 2-12	Bi-Linear 作一維影像內插結果 .....	18
圖 2-13	Cubic 函數 .....	19
圖 2-14	以 Cubic 函數為內插核心示意圖 .....	19
圖 2-15	一維度 Cubic 內核心函式補插區間示意圖 .....	20
圖 2-16	內插像素點與原始影像像素點位置關係(s)示意圖 .....	21
圖 2-17	Cubic Convolution 作一維影像內插結果 .....	21
圖 2-18	Raster Scanning 之顯示方式 .....	22
圖 2-19	二維度放大取樣示意圖，其中黑色圓圈代表原始影像像素點位置 .....	23
圖 2-20	垂直方向放大取樣示意圖 (綠色三角為垂直方向內插運算像素點) .....	23
圖 2-21	水平方向放大取樣示意圖 (紅色交叉為水平方向內插運算像素點) .....	24
圖 2-22	兩種不同的內插點放置 .....	26
圖 2-23	方法(1)，RGB 影像以 Bi-Cubic 放大 .....	27
圖 2-24	方法(2)，YUV 影像以 Bi-Cubic 放大 .....	28
圖 2-25	方法(3)，YUV 影像中，Y 以 Bi-Cubic、UV 以 Nearest Neighbor 作放大 .....	28
圖 2-26	以不同內插核心函數應用於 RGB color space 之結果 .....	29
圖 2-27	一般業界常用的 Pattern Generator 產生出的測試圖 .....	30
圖 2-28	放大實驗測試比較結果流程圖 .....	31
圖 2-29	誤差分析之影像取樣排列示意圖 .....	32
圖 2-30	邊框與文字影像 .....	32
圖 2-31	將圖 2-30 中 640×480 影像，以 Bi-Cubic 放大回 1280×1024 的局部結果 .....	33
圖 2-32	Master 影像 .....	34
圖 2-33	風景影像 .....	35
圖 2-34	動物影像 .....	36

圖 2-35 以部分影像為單位之內插演算法比較方法 .....	37
圖 2-36 以 8×8 為 window size，分析圖 2-30 以三種演算法放大的結果.....	38
圖 2-37 以 16×16 為 window size，分析圖 2-30 以三種演算法放大的結果.....	38
圖 2-38 以 8×8 為 window size，分析圖 2-32 以三種演算法放大的結果.....	39
圖 2-39 以 16×16 為 window size，分析圖 2-32 以三種演算法放大的結果.....	39
圖 2-40 以 8×8 為 window size，分析圖 2-33 以三種演算法放大的結果.....	40
圖 2-41 以 16×16 為 window size，分析圖 2-33 以三種演算法放大的結果.....	40
圖 2-42 以 16×16 為 window size，分析圖 2-34 以三種演算法放大的結果.....	43
圖 2-43 以 16×16 為 window size，分析圖 2-34 以三種演算法放大的結果.....	44
圖 3-1 浮點內插係數分佈圖 .....	45
圖 3-2 Cubic convolution 內插運算示意圖.....	47
圖 3-3 Fixed Point Model 乘法運算之程式(Pseudo code) .....	48
圖 3-4 調整過的 Fixed Point Look-up Tables.....	50
圖 3-5 差值分析所採用之灰階人物影像 .....	53
圖 3-6 差值分析所採用之彩色風景影像 .....	54
圖 3-7 差值分析所採用之彩色建築物影像 .....	55
圖 3-8 差值分析所採用之彩色動物影像 .....	56
圖 3-9 差值分析所採用之文字模式影像 .....	57
圖 3-10 差值分析所採用主要測試影像 .....	58
圖 4-1 一維度頻率域影像縮小，其中紅線為低通濾波器 .....	60
圖 4-2 空間域上的 Sinc 函數 .....	60
圖 4-3 影像放大所採用之 Cubic 核心函數.....	61
圖 4-4 以不同寬度之 Cubic Kernel 作影像縮小之比較.....	62
圖 4-5 一維度的 Local Average 影像縮小運算示意圖.....	65
圖 4-6 對於圖 4-4 (a)作 Local Average 影像縮小之結果.....	65
圖 4-7 空間域上的 Local Average 核心函數.....	65
圖 4-8 Local Average 縮小示意圖.....	66
圖 4-9 二維度 Local Average 演算法處理流程圖.....	67
圖 4-10 縮小實驗測試比較結果流程圖 .....	68
圖 4-11 自然風景影像 .....	69
圖 4-12 主要測試影像 .....	70
圖 4-13 環形條紋影像 .....	71
圖 4-14 動物影像 .....	72

# 一、緒論

由於近年來，平版型顯示器的大量生產，不論是家用電視或是人手一部的手持式裝置皆有大量的應用。平板型電視快速成長，源於 LCD TV 從小尺寸切入電視市場及 PDP TV 從大尺寸切入電視市場。而以目前趨勢來看，在突破傳統 CRT 電視之外型及畫面尺寸的瓶頸是平板型電視成之一大驅動力。此外，影音媒介（如 DVD）的普及及家庭劇院潮流的興起，亦有相關影響。而在各國數位廣播陸續開播後，預期又將掀起另一波平板型電視的成長，最後，平版型電視將成為家庭多媒體中心，成為串接家中數位裝置的中心。

而在未來產品發展趨勢方面，平版型電視在面板效能上可望持續改善，在相關視訊處理 IC 上也將朝多整合、高畫質及多媒體應用方面發展，而在相關連接介面上，為因應數位資訊的傳輸及保密需求，數位影像訊號逐漸取代類比影像訊號已是一股必然的潮流。而由於數位換機潮全球同步，各種不同尺寸顯示裝置將同時產生需求，因此如何廣化產品線，在這些顯示區域大小不一的顯示器顯示相同解析度的影像將是廠商未來發展的關鍵。而使用內插方法來轉換影像的解析度便是一種常見的解決方案。

## 1.1 簡介

本研究論文之主要研究內容，即在於針對將解析度大小不同的輸入影像，縮放成各種不同解析度的輸出影像，並討論影像處理後的品質問題以及找出最適合硬體實作的演算法並研究其最佳硬體實現化之方式來做討論，因此，本論文研究內容可分為兩個部份，即影像縮放演算法研究及其硬體實作之討論。

在影像放大的研究中，我們從取樣理論著手，探討以理想的 Sinc 函數作內插的放大效果，同時實驗不同的內插演算法，如 Nearest Neighbor、Bi-Linear、Bi-Cubic 等，加

以比較並且從這些方法中，找出具有最佳視覺效果的演算法，由 2-6 節的實驗結果顯示，以一般影像的放大來說，Bi-Cubic 是一個較為理想方法，其次是 Bi-Linear 與 Nearest Neighbor，然而對於在一些特別的測試影像，例如文字或有高影像反差特性的圖像，Nearest Neighbor 會是一個較優良的選項。因此我們是採用 Bi-cubic 作為此計畫影像放大研究的主要演算法，並且進一步討論硬體實作的相關問題。對於混合不同內插演算法的研究，我們希望可以探討出一個新的研究方向。而關於 Bi-cubic 的硬體實作，主要是以查表法來簡化硬體實作時的運算，並且以 C 程式語言，模擬硬體實作的情況，以建構出適合硬體使用，以及可轉換不同解析度的放大係數表，同時在 2-6 節中比較軟體 Bi-Cubic 運算與硬體作法之間的差異。

在影像縮小的研究方面，類似放大演算法研究，我們以縮小演算法的核心函數出發作討論，分析何種核心函數方適合於縮小的運算。由於直接對影像作重新取樣的縮小方法，如 Nearest Neighbor 或直接應用固定寬度的 Bi-Cubic 核心函數，皆會出現處理後影像因取樣密度不足所造成之失真的現象，故我們希望可以應用類似低通濾波的作法，在縮小運算的過程中，去除影像中高頻訊號後再作影像縮小取樣，以期能夠得到較符合學理與視覺感受的影像縮小結果，因此我們另外再提出了一種 Local Average 演算法之研究及實作。

## 1.2 相關文獻探討

在影像縮放文獻探討中，我們研究並探討了硬體實現影像縮放相關的專利，以及討論「適合硬體實作」及「較不適合硬體實作」的相關研究與一般常見的網路討論部分共 20 篇，在本小節中做概略說明。

## 1.2.1 專利文獻

參考文獻[1]提出一個影像縮放的架構，將來源影像在水平方向及垂直方向上進行縮放後形成一個目的影像，此專利設計出一個可在來源影像與目的影像時脈不同下的運作方式，藉由兩個不同的時脈 (Clocks) 處理與 Line Buffer 的設計，使得輸出的放大影像 Frame Rate，達到與輸入影像 Frame Rate 相同的播放速度。

參考文獻[2]於 2002 年被提出，主要重點在利用平行處理的架構對數位影像進行縮放。在平行處理的系統中有一常被運用到的「單一指令複合資料模式 (Single Instruction Multiple Data)」技術，可在同一時間處理大量的數據，因此可以大幅的增加資料處理的速度，運用在影像縮放上，也就有可能使用較複雜的運算得出較細緻的內插結果。

## 1.2.2 適合硬體實作之相關文獻



一般在影像縮放的軟體中，常採用的論文研究方法包含了 Sinc、Bi-Linear [3]、Bi-Cubic [4]、Nearest Neighbor [20]等作法，其基本原理皆是應用不同的核心函數 (Kernel Function) 來對影像作縮放，不同的核心函數的形式，即可得到不同的影像縮放效果，如 Lanczos [5]的影像縮放，即是 Sinc 核心函數的簡化實作方法，而 Mitchell-Netravali [6]、[7]所採用的核心函數，則可視為 Bi-Cubic 核心函數的廣義形式；這些基於核心函數的影像縮放方法，由於較為基本而簡易，所以也常被應用於軟、硬體影像縮放的實作。

在關於影像縮放的議題之討論中 [9]，以 Bi-Linear 作影像內插放大，是一個簡易同時效果可為大眾所接受的方法，以 Bi-Cubic 作影像內插放大，則是一個較 Bi-Linear 複雜一些，但效果較佳的選擇，即是在於討論以硬體實作 Bi-Cubic 影像放大相關的問題(詳見第三章)。

### 1.2.3 較不適合硬體實作之論文研究

除了上一小節所描述較基本的影像縮放方法之外，還有比較複雜，適合軟體實作的方法，如採用視覺上的幾何來取代核心函數的方式[10]作影像放大，這個方法能在放大影像的過程中保持影像的精確度，且放大後的邊緣看起來較平滑。而在論文[11]中，Storkey 提出了一個以機率混合模型（Probabilistic Mixture Model）作超解析度放大的方法，有別於以簡單濾波器作放大處理的作法，Storkey 所提出的混合機率法則是將每一個高解析度的點，由數個潛藏的點（Latent Nodes）混合計算而得出，而每個潛藏點則可視為對應到一塊同質的影像區域，這些潛藏點的機率模型，則是以信息傳遞（Belief Propagation）的方法計算得出，如此所作出的放大影像，其效果優於既有的濾波器處理方法，其放大的影像與真實大張的影像相比，有較小的平方誤差，放大後影像的紋路得以較好的方式呈現。

在 [12]、[13]的研究中，則是應用求解偏微分方程式的方法，來改善影像放大時鋸齒狀的狀況。其方法大致是在放大的影像上，建立格子點，並依照影像內容（如邊緣分佈），建構初始等高線與梯度函數，並驅動偏微方程求解的程序，以得出新的等高線分佈，如此修正過後的影像，會使得原本鋸齒的情況，藉由等高線的移動而變得較為平滑，但由於此類方法需求解偏微分方程，故計算量頗大，較不適合硬體實作。

此外，在影像處理的領域，對於放大比例很高的影像處理問題，又稱為超解析度問題。早期關於這方面的研究大多是從影像重建（Image Reconstruction）的觀點出發[14]，此類做法，基本上是假設存在一張真實的高解析影像  $H$ ，經過 Convolution 及 Down-Sampling 運算，這類運算轉換稱之為  $T$ ，成為我們所見到的低解析度影像  $L$ ，我們可以  $L = TH + Z$  的式子來表示這樣的關係，其中  $Z$  為雜訊或誤差向量。在[14]中假設這些轉換的過程是較簡單的線性轉換，以解出轉換矩陣的方式求解，不同的作法還包含利用最大化相似區間（Maximum Likelihood）求解，以得到一近似的高解析度影像  $H'$ ，Irani 等人在[15]中所提出的方法，即可快速求得這樣的  $H'$ 。但由於這樣的模型本身可能

包含無數解，所以雖然我們求得的  $H'$  在數值上正確，但對於人類的視覺感官來說卻可能會有相當不自然的感覺，如鋸齒狀及光暈的出現，對此，可行的改進模型是對所要求的  $H$  給予一先備知識 (Prior)，經由最大化事後機率 (Maximum A posteriori) 的計算，可得到一較為自然的影像。

為了想要得到更自然的影像，有些人提出了以學習方法為主的超解析度方法[16]、[17]、[18]，這類研究大都以全影像中的一小區塊 (Image Patch) 及相對應的高解析影像區塊的資料庫，在提升  $L$  的解析度時，先將  $L$  切割成小塊的區塊，然後從資料庫中找出最接近的低解析度區塊，再將相對應的高解析度區塊取回組成一高解析度的影像。這類方法的優點在於所得的高解析影像大多較為自然，且可在只有單張低解析影像時放大的倍數，其缺點則在於搜尋資料庫的時間成本以及應用範圍會跟資料庫的內容有關。

儘管上述的諸多方法，較不適用於硬體的實作，而多屬於軟體影像處理的研究結果，但其研究所得到的影像處理效能，卻是未來發展趨勢的重要指標。



## 1.3 論文章節簡介

本論文以下的章節內容安排如下：

第二章我們將探討常見的訊號內插演算法以及其數學上的模型與運用於影像處理的方法。常見的影像內插演算方法有 Sinc 內插演算法、最鄰近內插演算法(Nearest Neighbor Interpolation)、雙線性內插演算法(Bi-Linear Interpolation)、雙立方內插演算法(Bi-Cubic Interpolation)。此外，我們並將說明如何評估內插影像演算法的優劣，及影像縮放時的邊界處理及取樣間距選擇等細節問題。

第三章之重點著重於影像縮放演算法可行之硬體化研究，並進一步討論影像內插縮放硬體架構。與軟體不同的是，硬體實作重點在於計算負擔的簡化及定點數(Fixed-Point)的運算，因此我們在此部分的研究重點是針對查表法(Look-up Table)的架構及表格(Table)

的建置。

第四章之探討重點在於影像縮小，由基於前章節之研究及訊號處理理論，我們可由縮小影像處理的研究中，發展出一個優於傳統影像縮放的 Local Average 內插演算法。最末，第五章為本研究論文之研究成果討論總結，並提出未來可能的研究方向。



## 二、 影像內插演算法

本章節中，我們的討論以內插補點演算法為主題核心，並由內插核心函式原理解說開始，之後再進一步討論各種內插演算法的效能及實驗結果。

### 2.1 影像內插縮放概述

假設我們有一影像訊號，在我們欲將其以內插方式放大至 3 倍尺寸而不失去原有的影像品質(解析度)，我們必須找出其影像訊號於三倍頻率時的取樣值，如此才能夠維持放大後的影像品質。而如何產生在原有的像素之間的新像素是這個問題的關鍵。在這裡我們以作圖 2-1 說明，首先，我們考慮將一維度的數位訊號函式(圖 2-1(a))(以下稱為  $f$ ) 做三倍內插取樣(圖 2-1(b))，並將對應的連續訊號函式稱為  $g$  (圖 2-1(c))。  $f$  與  $g$  的關係式如下：

$$g(t) = f\left(\frac{1}{3}t\right)$$

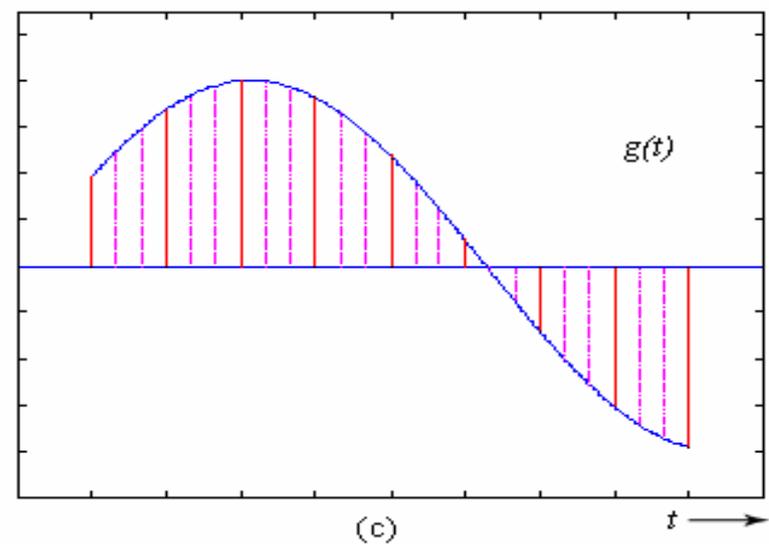
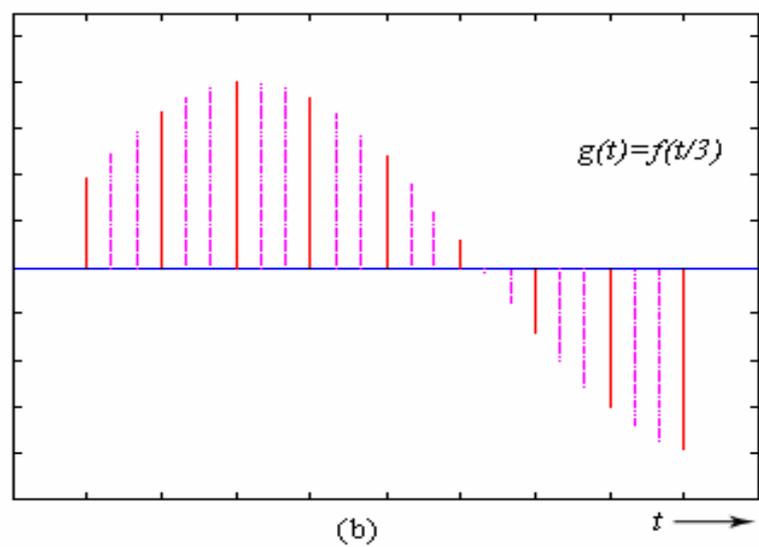
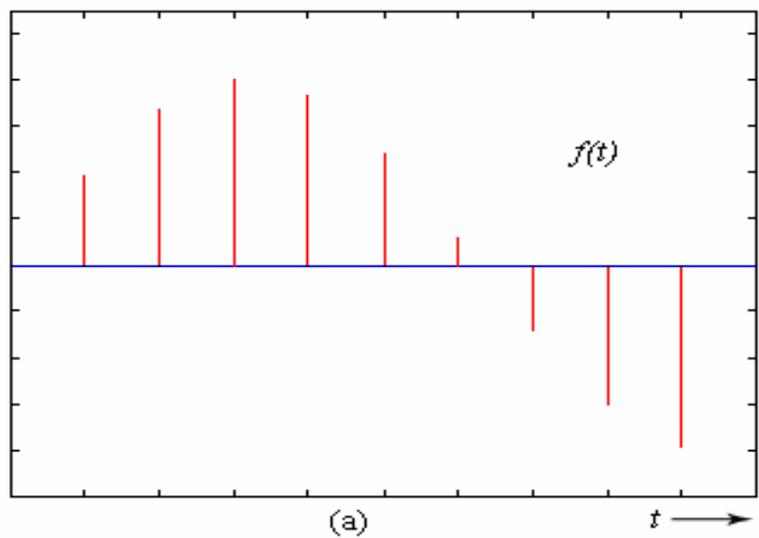


圖 2-1 三倍內插重新取樣。

在  $g(t)$  函數中， $t$  除以 3 餘 1 及餘 2 的取樣點 ( $t$  為整數)，於  $f$  函數上並無對應，因此若取樣時無假設任何限制條件，則沒有方法可以求出相鄰取樣樣本之間，訊號的變化方式。理論上，我們希樣重現  $f$  函數的連續變化特性於  $g$  函數上，由取樣定理可知，若信號被正確地取樣（例如原先的信號  $f$  的最高頻率小於取樣頻率的一半），我們就能夠做到「理想」的重建。然而在現實情況中，由於取像設備的頻寬有限，而景物的變化卻無一定規則可循，因此，當我們有一張需被放大的影像時，此一輸入影像的不一定符合充分取樣的情況。實際上，我們往往需要對非理想取樣的影像作內插補點處理，並必須同時得到良好的影像放大效果，這是本研究希望達到的目標。除此之外，以硬體處理內插補點時，我們不希望將空間域（Spatial Domain）轉換至頻率域（Frequency Domain）處理後，再轉換回空間域，以避免運算複雜度過高的問題。因此本研究論文主要僅探討影像縮放於空間域的處理。

以下，我們將輸入的影像訊號稱為來源影像（Source Image），並以函式  $f(x)$  表示；而放大後的影像稱為目標影像（Goal Image）並以函式  $g(x)$  表示，下列數學式則說明了來源影像與目標影像之間的函數關係[4]：

$$g(x) = \sum_k f(x_k) \cdot u\left(\frac{x - x_k}{h}\right)$$

其中  $g(x)$  為目標影像在座標  $x$  處的取樣強度值； $f(x_k)$  為目標影像在座標  $x_k$  處的取樣強度值； $u\left(\frac{x - x_k}{h}\right)$  表示內插補點的核心函數，也就是在  $f(x_k)$  處的影像取樣強度對於  $g(x)$  的影響度；或稱為權重值； $h$  為在來源影像上每次重新取樣（Re-Sampling）增加的距離。圖 2-2 為一維度線性內插放大（Linear Interpolation）的圖示，至於二維度放大的情形，將在章節 2.3 討論。

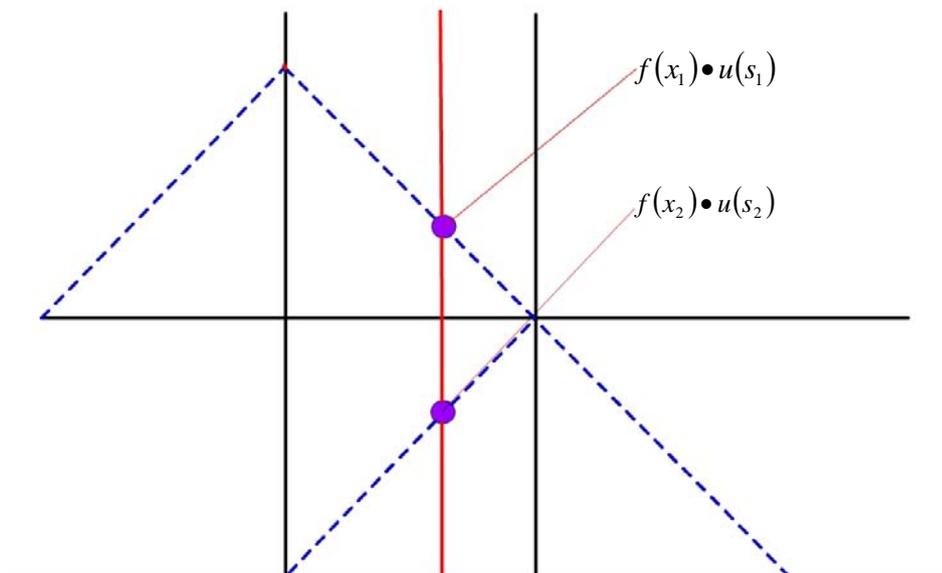


圖 2-2 一維度線性內插補點， $u(s)$  為線性內插補點演算核心函數 ( $s = x - x_k$ ;  $h = 1$ )，可視為內插取樣點之參考點的權重比值。

## 2.2 演算法比較



在二維度的內插放大演算法中有幾種常見的核心函數，如 Sinc、Nearest Neighbor、Bi-Linear、Bi-Cubic，因此在本節中，我們將會探討以上這四種演算法的主要特性及影像內插作法，並且討論其內插效果。

### 2.2.1 影像縮放演算法之核心函數

我們研究內插演算法時，首先必須設定的條件就是：放大影像的內插點位置，若對應到原始影像的像素點位置，則它們的值必須一致。此性質可以下列數學形式表示，假定  $f$  是原始影像的方程式，且  $g$  是內插放大後的影像方程式，則  $g(x_k) = f(x_k)$ ，其中  $x_k$

是位於相同位置的內插取樣點。進一步來看，內插方程式  $g$  可寫成以下 Convolution 的形式：

$$g(x) = \sum_k c_k u\left(\frac{x-x_k}{h}\right) \quad (2.1)$$

其中， $g(x)$  為內插所得的影像方程式， $u$  為內插核心函數， $x_k$  為內插取樣點， $c_k$  為在  $x_k$  處的取樣資料影像強度，即  $f(x_k)$ ， $h$  為取樣密度（取樣區間），由於上述  $g(x_k) = f(x_k)$  的限制，故  $u(0)=1$  且  $u(1)=0$ 。

## 2.2.2 Sinc 內插核心演算法

一般來說，在訊號分析的理論中，應用 Sinc 函數最理想之內插演算法的選擇，運用於影像放大處理時，則是將影像視為二維的離散信號，分別從水平方向及垂直方向進行內插。為了說明上的方便，以下我們將以一維的信號來解釋 Sinc 內插方法。

在理論上，最理想之內插演算法則通常是以利用傅立葉轉換（Fourier Transform），將原本的空間域信號轉換到頻率域上，再依據放大的倍率，將所需的較高頻率的值補上（補 0 值），再將之轉換回原本的空間。

Sinc 內插也可利用 Sinc 函數直接在原本的空間上做訊號的內插，Sinc 函數的定義如下：

$$\text{sinc}(x) = \begin{cases} 1, & x = 0 \\ \frac{\sin(\pi x)}{\pi x}, & x \neq 0 \end{cases} \quad (2.2)$$

從上式我們可以看出 Sinc 函數在除了 0 以外的整數點上值均為 0，而其變化的強度有愈來愈小的趨勢，請參考圖 2-3。

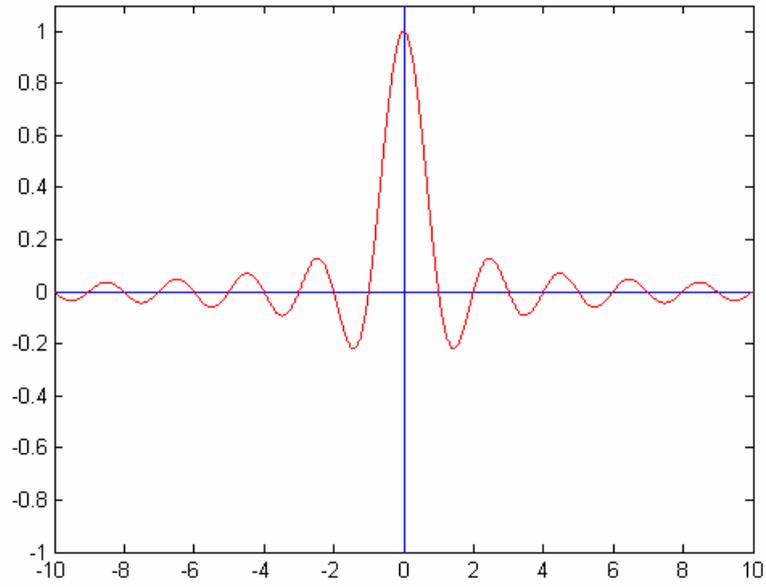


圖 2-3 Sinc 函數。

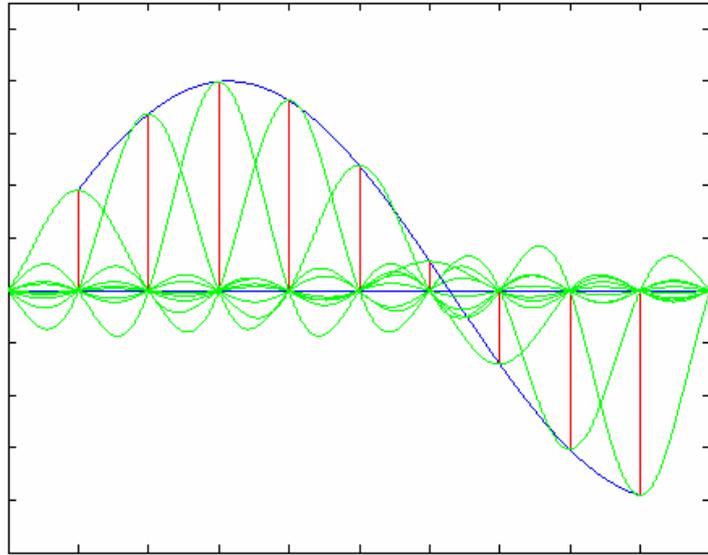


圖 2-4 以 Sinc 函數為內插核心所內插出之函數圖形示意圖。

利用 Sinc 函數來做內插時，我們一次看一個新的取樣點  $x'$ ，以其為原點，將每一個原有取樣點  $x_n$  對於  $x'$  的相對位置  $(x_n - x')$ ，代入 Sinc 函數算出  $x_n$  對  $x'$  的影響再乘

上  $f$  在  $x_n$  上的值後相加，所得的值  $\sum_n f(x_n) \text{sinc}(x_n - x')$  即為  $x'$  上的內插值，可參考圖

2-4。透過 Sinc 核心函數作影像放大的處理，在理論上是一個最佳的選擇，但由於取樣點為固定的，其所包含的資訊是有限的，因此我們並無法得知更高頻率的訊號強度，所內插出的放大影像也會較模糊，同時由於 Sinc 函數內插所需之參考像素涵蓋所有像素點（理論上為無限多），因此放大的影像可能產生波浪狀的不自然圖形，稱為波紋效應（Ringing Effect），在圖 2-5 與圖 2-6 中所提供的例子，是一個以 Sinc 核心函數，作影像二維放大處理的結果，在二維的影像放大中，我們可以清楚的看到波紋效應所造成的不自然結果。

由於 Sinc 函數的涵蓋區域為無限大，因此在硬體上仍難以實現。許多其他的內插法便採用不同的函數來取代 Sinc 函數，常見的有 Nearest Neighbor（最接近像素內插）、Bi-Linear（雙線性內插）、Bi-Cubic（雙立方內插）內插法等等，我們將在以下的小節中作討論。



圖 2-5 波紋效應（Ringing Effect）。



圖 2-6 Sinc 函式作二維影像內插放大結果。

### 2.2.3 Nearest Neighbor 內插核心演算法

最鄰近內插法 (Nearest Neighbor Interpolation)，是最簡單的內插方法，依據從內插影像座標系的插補位置，找出原始影像中最相近點的影像強度值，作為補點之影像強度值。其作法為：

1. 找出放大後影像像素座標與原始影像座標之關係。
2. 以原始座標之最鄰近像素灰階值為放大後影像像素灰階值。

這種內插補點方法並不是真的計算影像強度值，而只是複製已存在像素點的影像強度值。因為它不改變影像強度值，所以原影像的影像特徵更易被保留。對於一維的最近鄰內插演算法來說，需要計算出的新像素點強度值所需參考的原影像像素點強度值為二個點。同理，對於二維最近鄰內插演算法來說，參考的原影像像素點強度值為四點。以一維度的 Nearest Neighbor 內插法而言。其核心函數  $u(s)$  數學式可表示為：

$$u(s) = \begin{cases} 0, & |s| > 0.5 \\ 1, & |s| \leq 0.5 \end{cases} \quad (2.3)$$

其中  $s$  為需被內插出的像素點與原始影像格點距離。圖 2-7 為 Nearest Neighbor 之內插核心函數之繪圖；圖 2-8 為以 Nearest Neighbor 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。其一維影像放大的結果如圖 2-9 所示。

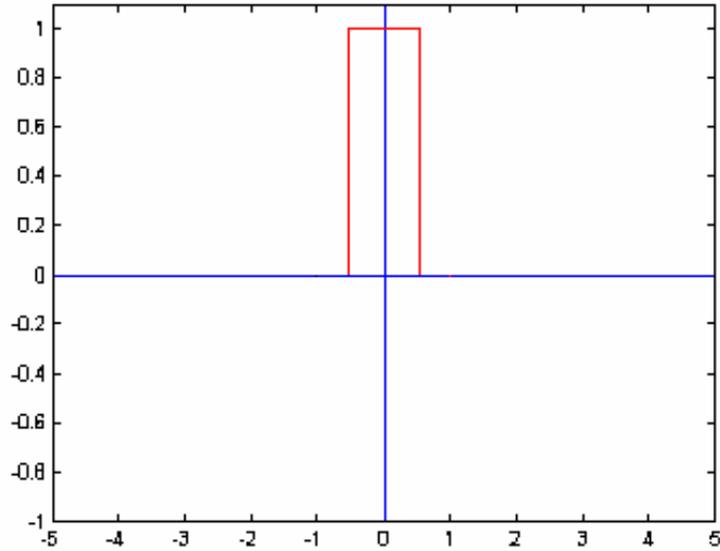


圖 2-7 Nearest Neighbor 函數。

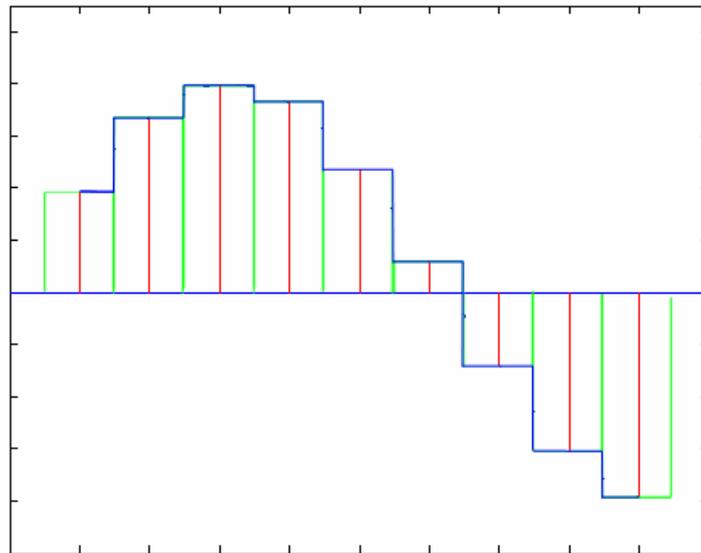


圖 2-8 Nearest Neighbor 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。



圖 2-9 Nearest Neighbor 作一維影像內插結果。

## 2.2.4 Linear/Bi-Linear 內插核心演算法

線性內插 (Linear Interpolation) 演算法以原始輸入影像座標系中的兩個已知像素點的影像強度值 (Intensity) (可參考圖 2-2)，乘以其權重值來計算出欲補點之影像強度值。

如果推廣此一觀念，則雙線性內插演算法 (Bi-Linear Interpolation) 之作法為分別在垂直及水平兩個方向應用兩個線性內插補點。以單一維度 (水平或垂直方向) 而言，其內插核心函數  $u(s)$  數學式可表示為：

$$u(s) = \begin{cases} 0, & |s| \geq 1 \\ 1 - |s|, & |s| < 1 \end{cases} \quad (2.4)$$

其中  $s$  為被內插出的像素點與原始影像格點距離。圖 2-10 為 Linear 之內插核心函數之繪圖；圖 2-11 為以 Linear 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。其一維影像放大的結果如圖 2-12 所示。

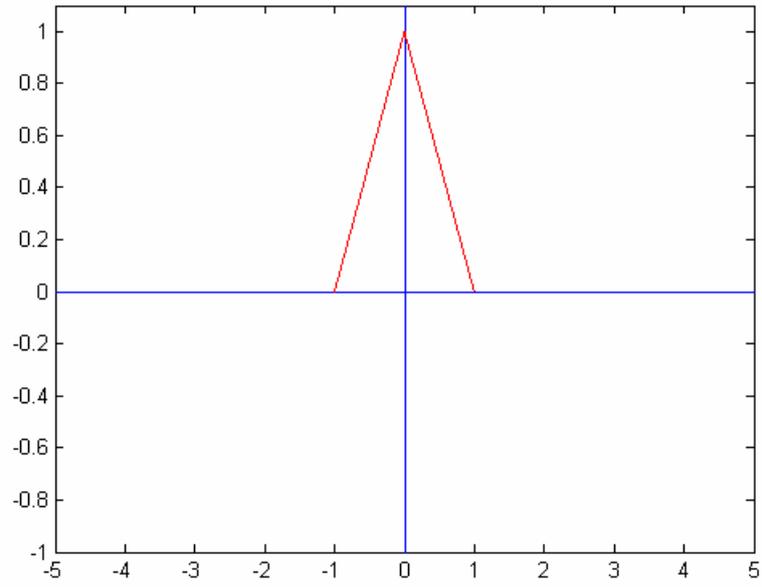


圖 2-10 Linear(線性)函數。

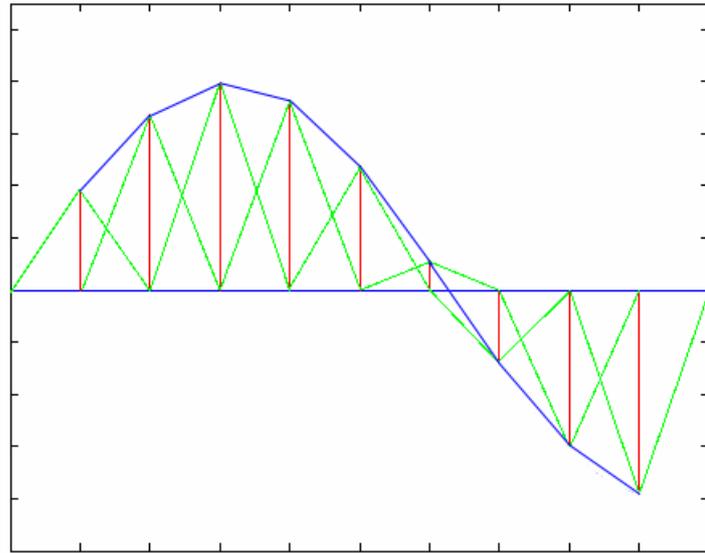


圖 2-11 為以 Linear 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。



圖 2-12 Bi-Linear 作一維影像內插結果。

## 2.2.5 Cubic/Bi-Cubic 內插核心演算法

立方內插 (Cubic Convolution Interpolation) 演算法之做法依然是參考原始影像中的像素值，與 Nearest Neighbor 及 Linear/Bi-linear 不同的是，立方內插參考更多的原始影像像素值。以單一維度而言，立方內插參考四個已知原始像素點之影像強度值，乘以其權重值來計算出欲補的點之影像強度值。推廣至二維度內插則需參考原始輸入影像十六個已知像素點的影像強度值 (Intensity)。

以單一維度內插放大而言，其核心函數  $u(s)$  數學式可表示為：

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1, & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2, & 1 < |s| < 2 \\ 0, & 2 < |s| \end{cases} \quad (2.5)$$

其中  $s$  為需被內插出的像素點與原始影像格點距離，圖 2-13 為 Cubic 之內插核心函數之繪圖；圖 2-14 為以 Cubic 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。其核心函數的插補區間曲線則如圖 2-15 所示，而關於此內插核心之立方函數之推導細節請見附錄一。

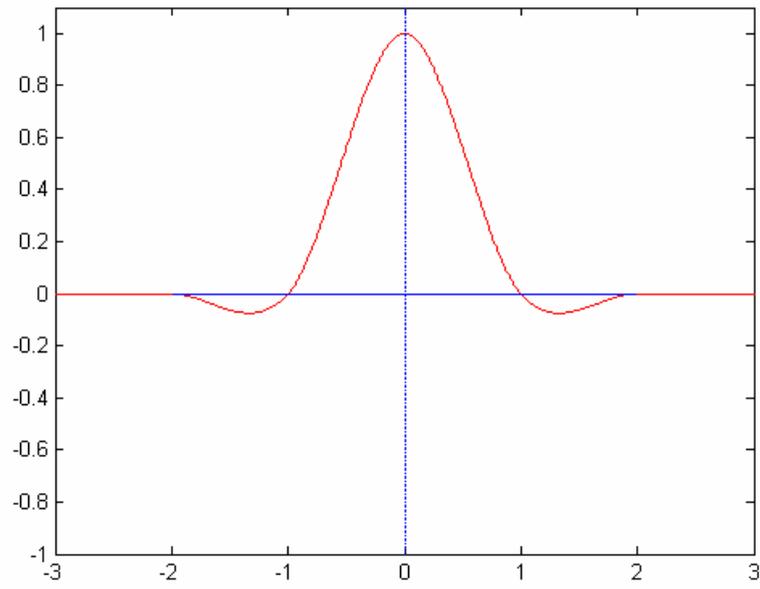


圖 2-13 Cubic 函數。

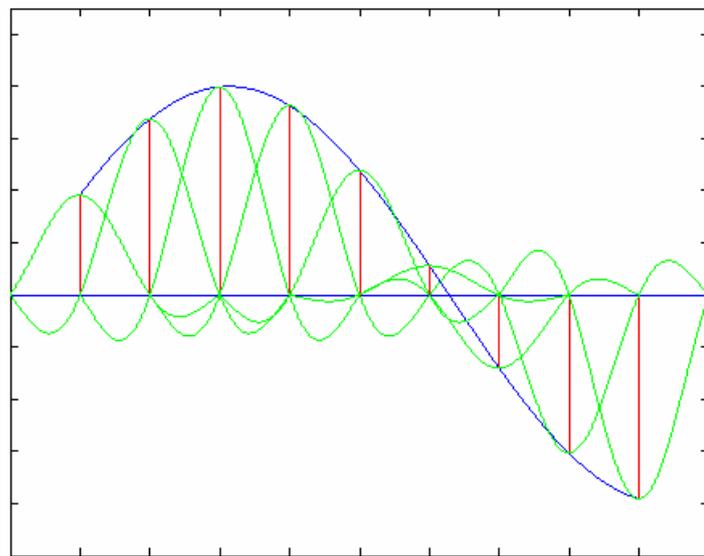


圖 2-14 為以 Cubic 為內插核心，對一數位離散訊號作內插插補後的圖形繪圖。

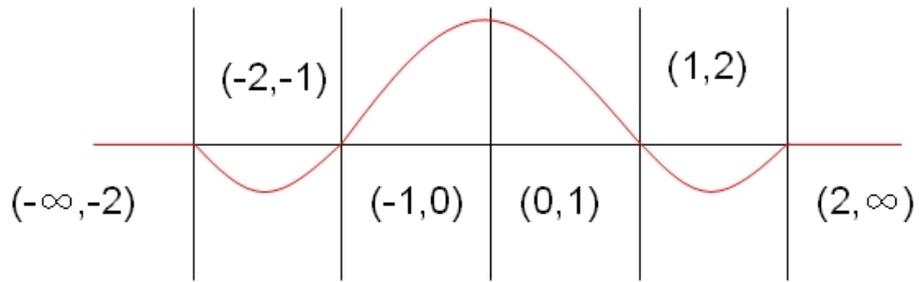


圖 2-15 一維度 Cubic 內核心函式補插區間示意圖。

在圖 2-16 中，我們以圖示的方式，來表示在每個一維的 Bi-Cubic 內插放大運算中，相鄰的原始影像像素點  $x_{k-1}$ 、 $x_k$ 、 $x_{k+1}$ 、 $x_{k+2}$ ，對於所要內插出來的像素點  $x$ ，其影響的比重係數值。而後透過 Weighted Convolution 的運算，我們便可以內插出新影像點  $x$  的亮度值。其一維影像放大的結果如圖 2-17 所示。



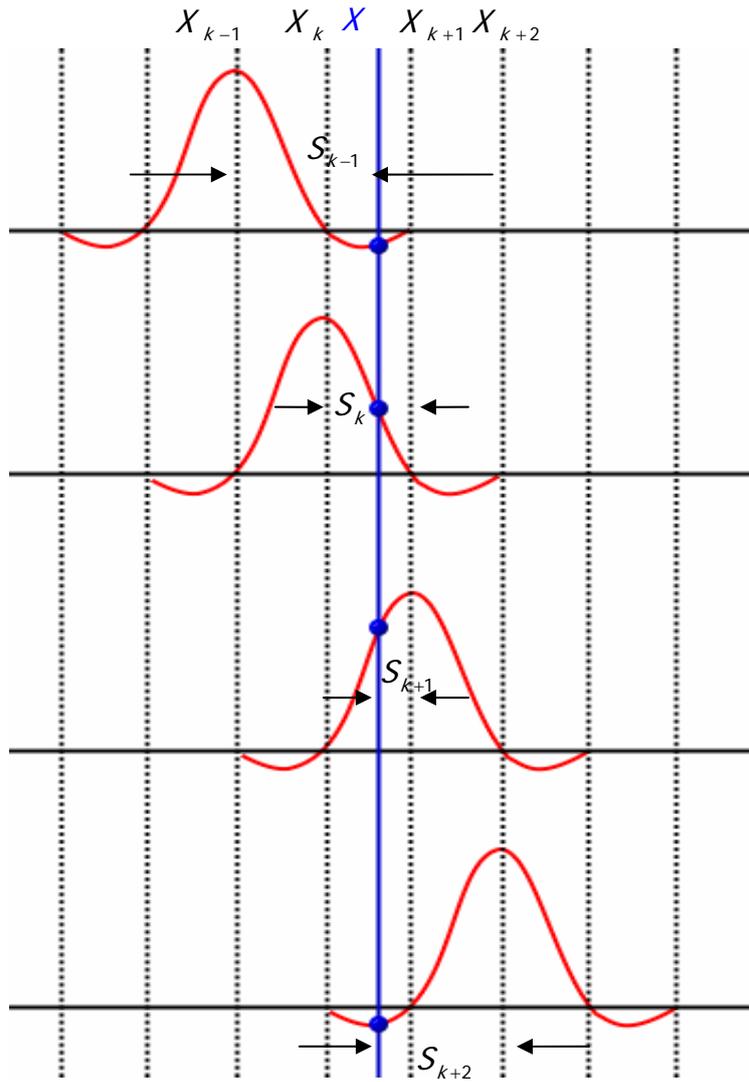


圖 2-16 內插像素點與原始影像像素點位置關係(s)示意圖。



圖 2-17 Cubic Convolution 作一維影像內插結果。

## 2.3 二維度 Cubic 內插放大演算法

由於 Cubic 演算法有比較完整的數學推演(附錄一)，同時亦被普遍應用於一般商用的影像處理軟體，如 Photoshop，其效果優良，故我們將進一步針對 Bi-Cubic 其推廣至二維度影像放大的處理作討論，以一個直觀的處理方式來實作本內插插補核心（如圖 2-19 到圖 2-21 所示），相類似的二維影像放大處理方式，亦可見於[1]中，但這種影像之二維度處理的架構，為 Bi-Cubic 內插運算的一個常見而且實用的設計方式。

類似於[1]的設計，我們在實作 Cubic Convolution 時，因其內插函數特性為：在一個維度上需考慮四個參考點，所以在二維度上就必需參考原始 16 個點（ $4 \times 4$ ）。然而在硬體架構上，一般的顯示方式為 Raster-Scanning(圖 2-18)的資料的讀入方式，故以 Scan Line 為單位，共需要五條 Line Buffer，前面四條應用於垂直方向內插暫存，最後一條用於水平方向內插暫存。



圖 2-18 raster-scanning 之顯示方式。

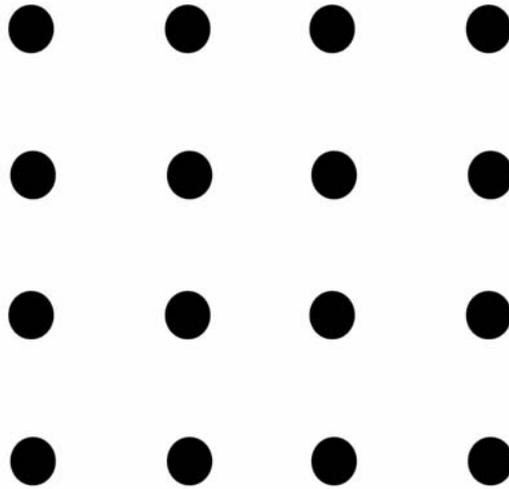


圖 2-19 二維度放大取樣示意圖，其中黑色圓圈代表原始影像像素點位置。

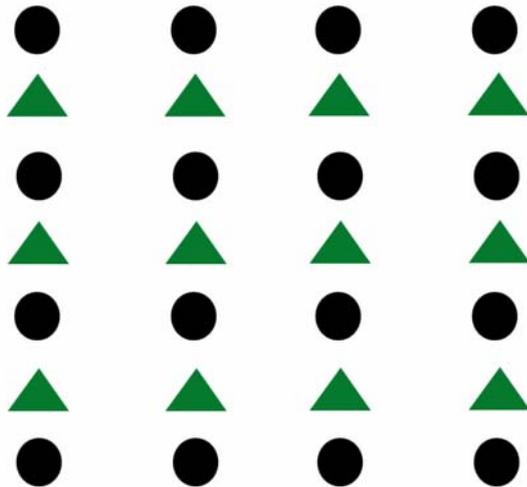


圖 2-20 垂直方向放大取樣示意圖（綠色三角為垂直方向內插運算像素點）。

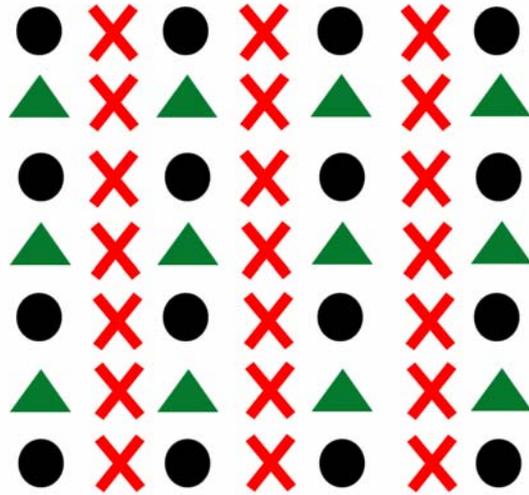


圖 2-21 水平方向放大取樣示意圖（紅色交叉為水平方向內插運算像素點）。

在作內插運算時，假設需補插入的目標影像之像素點，位於來源影像的第二及第三取樣點之間，即座標(0,1)區間（參考圖 2-15）。若我們先進行一維度的 Cubic Convolution 運算，則須參考來源影像的四個取樣點，以產生四個三角形的內插點（參考圖 2-20）。而後，再進行第二個維度的 Cubic Convolution 內插放大，其方法與第一個維度的內插作法類似，如此，我們便可以計算出輸出影像的內插點，其結果如圖 2-21 之叉叉點所示。

## 2.4 邊界處理

對於影像放大過程中，邊界處理的問題，會關連到兩個問題的探論，一是影像放大過程中，內插點的擺放位置；二是如何猜測原始影像以外區域的像素顏色值，以作為計算內插點時的參考資訊。對於這兩個問題的處理，我們將舉一個由 4 個點放大到 8 個點的內插放大處理，來作說明。

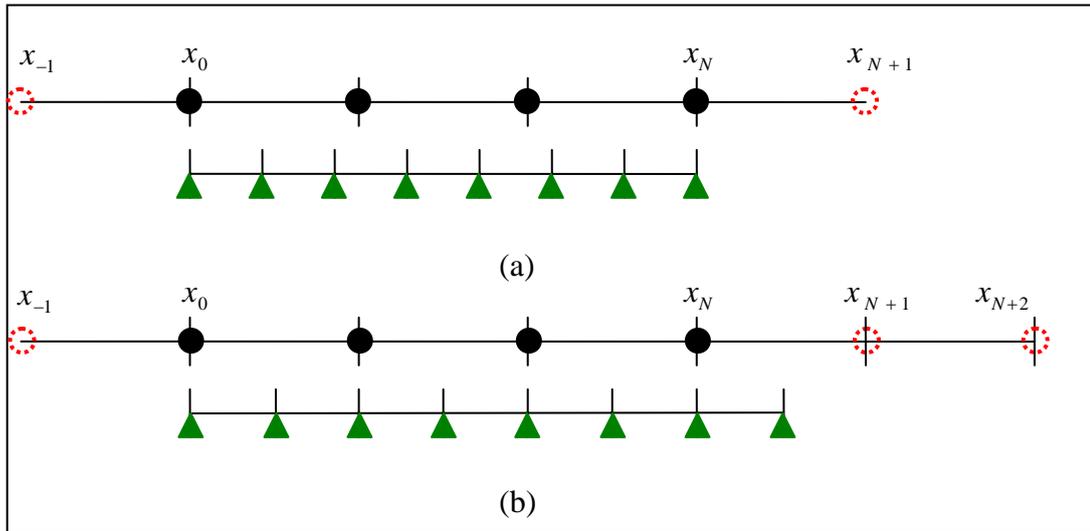


圖 2-22 兩種不同的內插點放置 (a) 前後對齊的放置 (b) 僅前方對齊的放置

關於影像放大過程中，內插點擺放位置的選擇，如圖 2-26 所示，當影像由 4 個點放大到 8 個點時，可以有兩種擺放的方式，第一種方式如圖 2-26(a)所示，內插點的擺放是將內插放大影像，以頭尾對齊原始影像頭尾像素點的方式作擺放，其中內插的綠色三角點，第一個和最後一個對齊原始影像上的第一個  $x_0$  和最後一個黑色圓點  $x_N$ ，因此，每個內插的綠色三角點，其間隔為  $\frac{3}{7}$ ，同時每個綠色三角點的內插運算，需要參考原始影像中，相鄰的四個圓形點的颜色值，方能夠求出一個新的綠色三角點颜色，因此我們需要推測原始影像之外，前後兩個增補點的颜色值  $I(x_{-1})$  和  $I(x_{N+1})$ ，才能夠完整的內插出所有的綠色三角點，故在邊界情況的處理上，我們需要額外考慮原始影像的外插計算，最簡單的方式即為令  $I(x_{-1}) = I(x_0)$  與  $I(x_{N+1}) = I(x_N)$ ，如此便能夠處理好放大過程中，邊界運算的情況；而比較複雜的外插方式則如 [4] 所採用的  $I(x_{-1}) = 3I(x_0) - 3I(x_1) + I(x_2)$  與  $I(x_{N+1}) = 3I(x_N) - 3I(x_{N-1}) + I(x_{N-2})$  外插運算，來推測外圍影像點颜色；實際上由於外插運算的目的在於猜測原始影像中，不存在的像素點颜色值，因此無論採用何種方式作運算，對於影像內插放的結果而言，並無絕對的優劣之分。

同理，如圖 2-22(b)所示，我們也可採用另一種不同的內插點擺放方式來作影像放大，其中我們僅把放大前與放大後的兩張影像，其最前端的第一個點對齊在一起，而後

則是以間隔 $\frac{4}{8}$ 的方式來放置每一個內插綠色三角點，在這種情況下，我們則需要在原始

影像的末端，增補兩個新的像素點顏色值 $I(x_{N+1})$ 和 $I(x_{N+2})$ ，方能夠完整的進行所有的內插點運算，同樣的我們也可以採用簡單的 $I(x_{-1})=I(x_0)$ 、 $I(x_{N+1})=I(x_N)$ 與 $I(x_{N+2})=I(x_N)$ 的方法來補上這些額外原始影像點的顏色值；比較這兩種不同的內插點擺放方式，如圖 2-22(a)所示的第一種方式，較常被一般如 Photoshop 的影像處理軟體所採用，而如圖 2-22(b)所示的第二種方式，乍看似乎比較麻煩，卻具有一個特出的優點，在於其內插點的間隔在一般的情況下，通常是一個可以約分的分數，以此例而言即為

$\frac{4}{8} = \frac{1}{2}$ ，此一可以約分的情況，有利於在硬體實作的過程中，節省 Look-up Table 建構所

需要記憶空間(詳見第三章說明)，簡化了硬體實作影像放大的複雜度並降低硬體成本。

在此一研究中，我們分別實作了上述兩種不同的內插點放置方式；為了將我們所實作的 Bi-Cubic 內插放大軟體模擬結果，與 Photoshop 等標準影像處理軟體的結果作比較，因此我們主要都是採用第一種方式的內插點放置，以求出軟體實驗的比較數據與相關結果；然而在第三章中，由於討論的是內插運算之硬體實作相關問題，因此，該小節的所有程式實作與實驗結果，皆是以第二種內插點放置方式來作討論與說明。

## 2.5 RGB/YUV 色域空間的選擇與比較

視訊影像相關的商品中，常常可以發現一般除了 RGB 色域的輸入輸出埠外，還有 YUV 色域的輸入輸出埠也相當常見。因此我們希望可以知道若對於亮度和色度分別以不同的影像放大演算法可否得到較佳影像品質，或是可以簡化演算法實現於硬體裝置上的複雜度。

RGB 模型中，每種色彩是以其紅、綠、藍的主要頻譜成份來顯現，而 YUV 的模型則不同，其色彩是由一亮度 (Luminance) 和兩個色彩度 (Chrominance) 組成，由於人眼對色度的敏感度不及對亮度的敏感度，所以許多視頻系統，在色度通道上，會作較疏的取樣 (相對亮度通道)，如此，便可以在不明顯降低畫質的同時，減少視頻信號所需的總頻寬。底下，我們將以比較傳統 YUV 色域空間與 RGB 色域空間，在作影像放大

時的異同，並對 YUV 色域空間的放大方法，作一實用可行的建議。

我們分別針對 RGB 及 YUV 兩種不同的輸入影像做內插放大，做以下實驗：

- (1). RGB : Bi-Cubic 。
- (2). YUV : Bi-Cubic 。
- (3). Y : Bi-Cubic 、UV : Nearest Neighbor 。

其中關於 YUV 影像的處理，我們對 Luminance Channel 以 Bi-Cubic 為放大核心演算法，對 Chrominance Channel 則採用 Nearest Neighbor，以降低運算複雜度。實驗的結果如圖 2-24 至圖 2-26 顯示，在放大倍率小於 3 倍之下，我們皆得到優良的視覺效果，亦即當 RGB 影像與 YUV 影像，皆以 Bi-Cubic 作放大時，其視覺效果類似，而對於 YUV 影像，以圖 2-26 所示，若 Y 以 Bi-Cubic、UV 以 Nearest Neighbor 作放大，相較於 YUV 皆以 Bi-Cubic 放大的方式，兩者的視覺效果亦相類似。

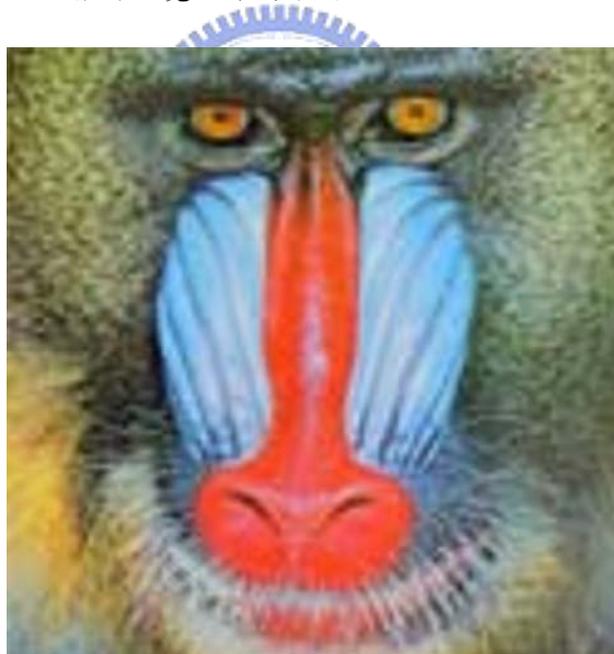


圖 2-23 方法(1)，RGB 影像以 Bi-Cubic 放大。

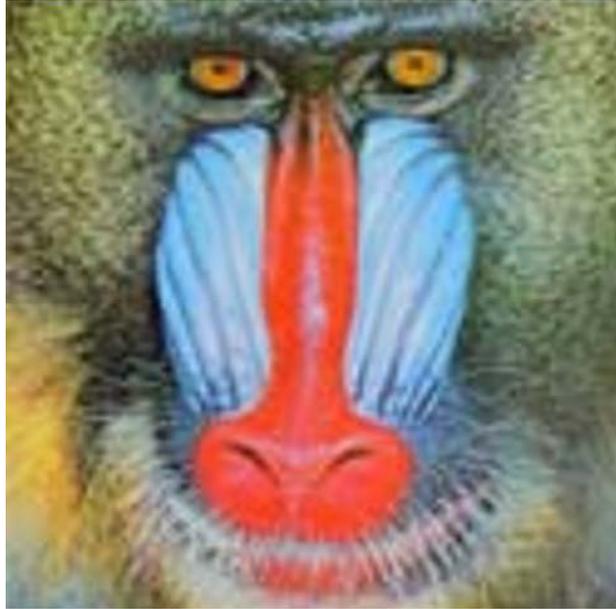


圖 2-24 方法(2)，YUV 影像以 Bi-Cubic 放大。

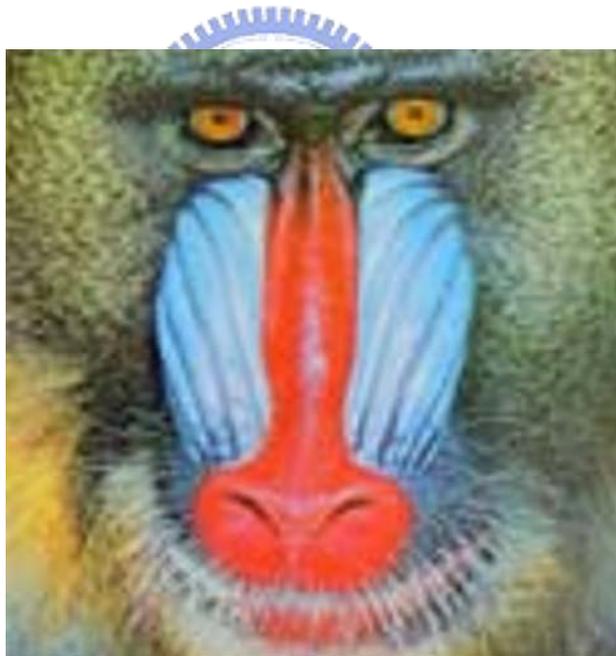


圖 2-25 方法(3)，YUV 影像中，Y 以 Bi-Cubic、UV 以 Nearest Neighbor 作放大。

關於在不同的 Color Channel 以不同的內插補點核心做補點的方式，僅適用於以 Luminance 和 Chrominance 來表示色彩的影像，如本例所實驗之 YUV Color Space,對於一般的 RGB Color Space 因會產生色偏問題(圖 2-25)故不適用本方法。

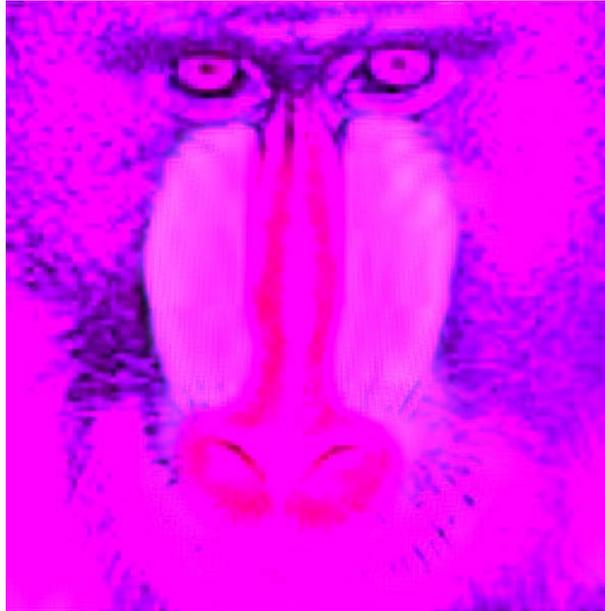


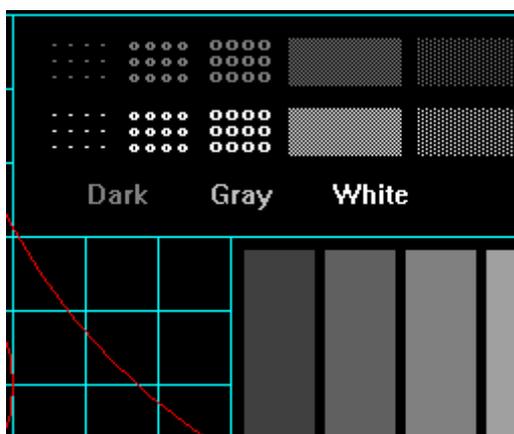
圖 2-26 以不同內插核心函數應用於 RGB color space 之結果，本例應用 Bi-Cubic 內插核心函數於 B channel；其餘二 channel 之內插核心函數為 Nearest Neighbor。

## 2.6 實驗比較結果

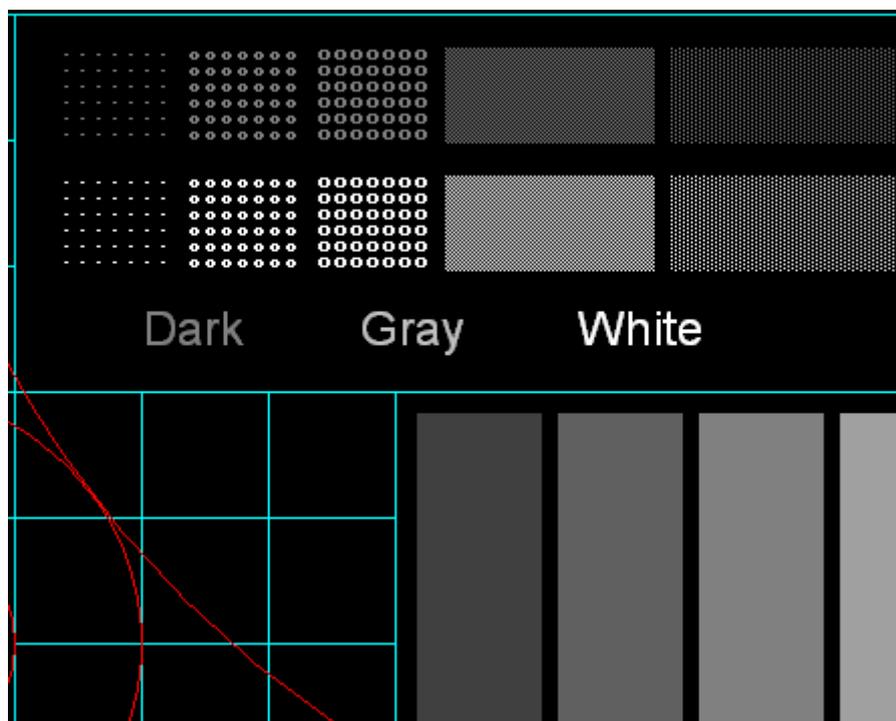


為了能夠定量地比較不同影像放大方法所做出來的效果，我們一開始執行一般的螢幕測試程式，以 Pattern Generation 的方式來產生不同解析度下的測試影像，包含了  $800 \times 600$ 、 $1024 \times 768$ 、 $1280 \times 1024$  等解析度的線條影像、文字影像、格線影像、曲線影像與邊界測試影像等，以作為 Ground Truth 的標準影像，並嘗試將如  $800 \times 600$  的標準影像，以不同的內插演算法放大到  $1280 \times 1024$ ，再與  $1280 \times 1024$  的標準影像作差值的分析比較，然而，此一比較方法實際上卻是行不通的，問題在於由 Pattern Generation 程式在不同解析度下所產生的影像，如  $800 \times 600$  與  $1280 \times 1024$ ，其兩張影像所含的內容不盡相同，故當把  $800 \times 600$  的影像放大到  $1280 \times 1024$  後，與原有  $1280 \times 1024$  的標準影像內容相較，會無法對齊一致，因此我們便無法以這樣的方式來作定量的影像放大誤差分析。舉例來說，若在  $800 \times 600$  的標準影像中，原本為 1 個像素寬的線段，放大到  $1280 \times 1024$  後，其寬度會變得大於 1 個像素寬，然而，在  $1280 \times 1024$  的標準影像中，其相對應比較的線段仍是 1 個像素寬，因此便發生了影像對準 (Image Registration) 的問題，此一問題在

Pattern Generation 的情況下，（標準影像的產生是由外在的程式所控制），變得難以控制(圖 2-29)。



(a)



(b)

圖 2-27 以一般業界常用的 Pattern Generator 分別在 SVGA(a)及 SXGA(b)解析度下產生出的測試圖，線狀圖形之寬度皆為 1 pixel。



圖 2-28 放大實驗測試比較結果流程圖。

由上述說明可知，在作影像放大誤差分析的過程中，如何選擇 Ground Truth 的標準影像，以及如何處理內插放大影像與標準影像的內容對準 (Registration) 問題，對於比較結果是否正確而言，是一個重要而根本的因素。對此，我們提出了如圖 2-所示的比較流程，對於一張大張的  $1280 \times 1024$  之 Ground Truth 影像  $I_0$ ，先將該圖縮小至  $800 \times 600$  的小圖，再分別用 Nearest Neighbor、Bi-Linear 以及 Bi-Cubic 放大回  $1280 \times 1024$  的大張影像  $I_1$ 、 $I_2$  和  $I_3$ ，之後再將三張影像與  $I_0$  作相減比較，求出影像差值的最大值及平均值與變異量，其中，關於第一個大張影像縮小的作法，我們是以簡單的 Nearest Neighbor 來進行，單純的對大張影像作重新取樣，丟掉部分的影像資訊而得出小張影像，作為放大的基準；而關於放大影像與原始標準影像的影像對準問題，我們則是採取了自行撰寫程式的方式來作控制，也就是無論是以 Nearest Neighbor 作縮小，或是以 Nearest Neighbor、Bi-Linear、Bi-Cubic 作放大，我們都是以如圖 2-22 所示的取樣方式，自行撰寫程式來對影像的第一點與最末點作對準，因此可以適當的控制影像對準的精確度，使得之後的誤差分析有一個公平的基準。在這樣的取樣排列之下，我們特別可以看到如圖 2-31、圖 2-33 所示，對於具有邊框的原始影像，其縮小與放大後的影像，邊框都會存在。

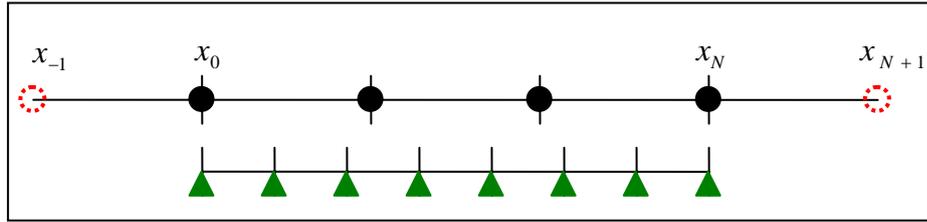


圖 2-29 誤差分析之影像取樣排列示意圖

關於影像放大誤差的分析結果如圖 2-31 至圖 2-35、表 2-1 至表 2-4 所示，由數值上的誤差統計可以看出，對於接近文字模式的測試影像，Nearest Neighbor、Bi-Linear、Bi-Cubic 的放大誤差都差不多，甚至是 Nearest Neighbor 有可能平均誤差會稍小一些，這是因為在文字模式中，影像顏色與圖樣的變化較少且較單調，較適合以 Nearest Neighbor 的放大方式來維持原本影像的亮度與單調的特性，但是對於自然影像而言，一般來說就會呈現出 Bi-Cubic 優於 Bi-Linear 與 Nearest Neighbor 的情況，這也符合一般在作影像放大時，所預期的情況，因此可以佐證，採用 Bi-Cubic 的核心函數作內插放大，是一個理想的選擇。



圖 2-30 邊框與文字影像：由原始 1280×1024 標準影像經 Nearest Neighbor 縮小到 640×480 的影像，用來作為後續放大回 1280×1024 的基準。

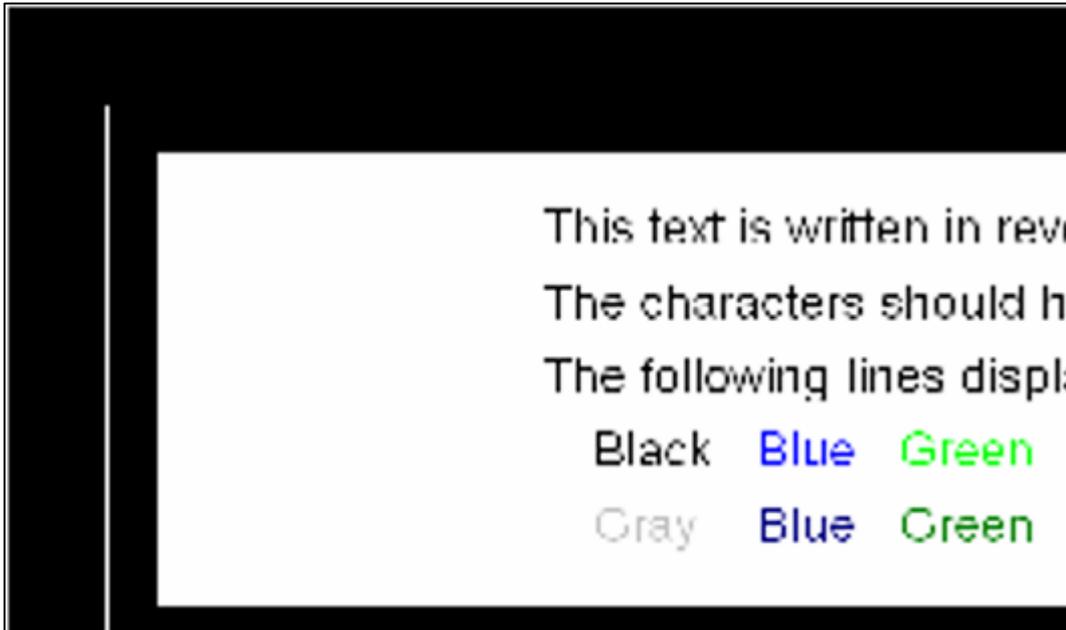


圖 2-31 將圖 2-30 中 640×480 影像，以 Bi-Cubic 放大回 1280×1024 的局部結果。



表 2-1 以邊框與文字影像作不同演算法之放大誤差統計表。

$ \Delta $	Nearest Neighbor	Bi-Linear	Bi-Cubic
Max	255.000000	255.000000	255.000000
Ave.	2.610062	2.916142	2.857430
Std.	2.175092	3.340302	3.084792

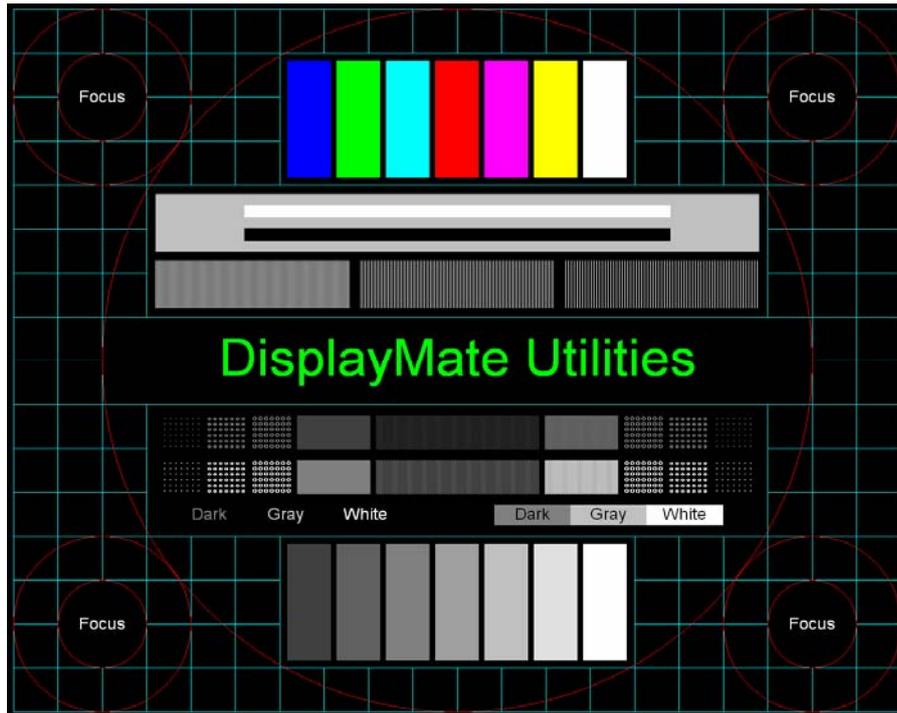


圖 2-32 Master 影像：原始 1280×1024 標準影像經 Nearest Neighbor 縮小到 800×600 的影像，再放大回 1280×1024。



表 2-2 以 Master 影像作不同演算法之放大誤差統計表。

$ \Delta $	Nearest Neighbor	Bi-Linear	Bi-Cubic
Max	255.000000	255.000000	255.000000
Ave.	17.363450	17.250752	17.278905
Std.	5.297565	5.337443	5.325322



圖 2-33 風景影像：原始 1280×857 標準影像經 Nearest Neighbor 縮小到 800×600 的影像，再放大回 1280×857。



表 2-3 以風景影像作不同演算法之放大誤差統計表。

$ \Delta $	Nearest Neighbor	Bi-Linear	Bi-Cubic
Max	189.666667	137.666667	144.666667
Ave.	3.978078	3.642789	3.483795
Std.	4.761046	4.576241	4.497608



圖 2-34 動物影像：原始 1280×1024 標準影像經 Nearest Neighbor 縮小到 800×600 的影像，再放大回 1280×1024。



表 2-4 以動物影像作不同演算法之放大誤差統計表。

$ \Delta $	Nearest Neighbor	Bi-Linear	Bi-Cubic
Max	252.000000	252.000000	228.666667
Ave.	2.380312	2.380312	1.945215
Std.	3.772444	3.772444	3.028578

除了以上的比較方法外，為了進一步分析不同區塊於整張影像的特性差異，我們另外開發了以影像中的  $n \times n$  像素為評估單位的比較方法(如圖 2-26)。除了計算像素間的差異值大小外，我們將差異小的影像分別以紅色(Nearest Neighbor)、藍色(Bi-Linear)、綠色(Bi-Cubic)為遮罩後再貼回原影像以方便觀察其差異。我們嘗試了分別以  $1 \times 1$ ,  $2 \times 2$ ,  $8 \times 8$ ,  $16 \times 16$ ,  $32 \times 32$  為影像取樣比較的單位做比較，我們將  $8 \times 8$  及  $16 \times 16$  之比較後的結果圖 2-37~圖 2-44，於自然影像得到的結果依然是 Bi-Cubic 優於其餘兩種內插演算法(表 2-5)。

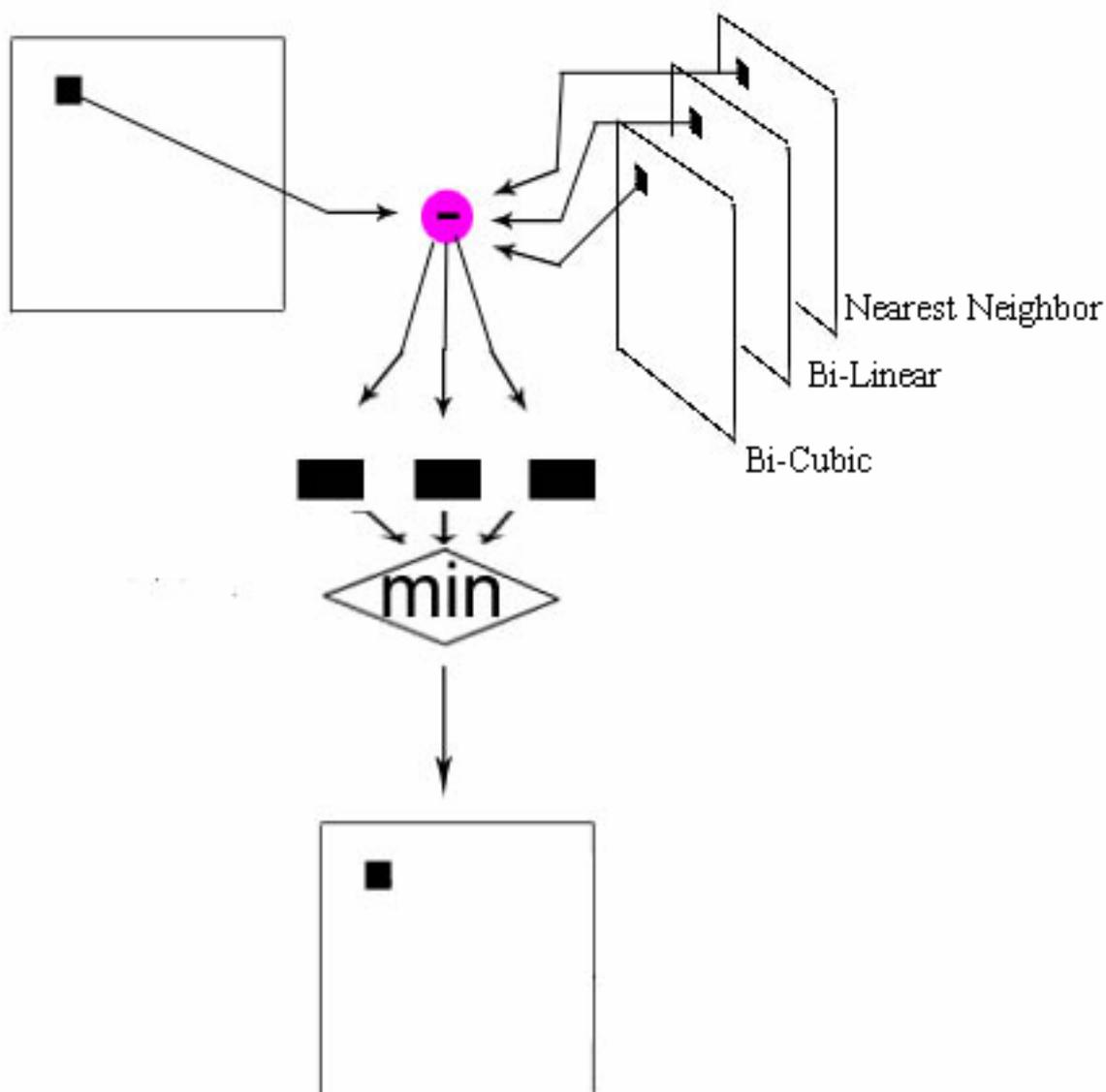


圖 2-35 以部分影像為單位之內插演算法比較方法

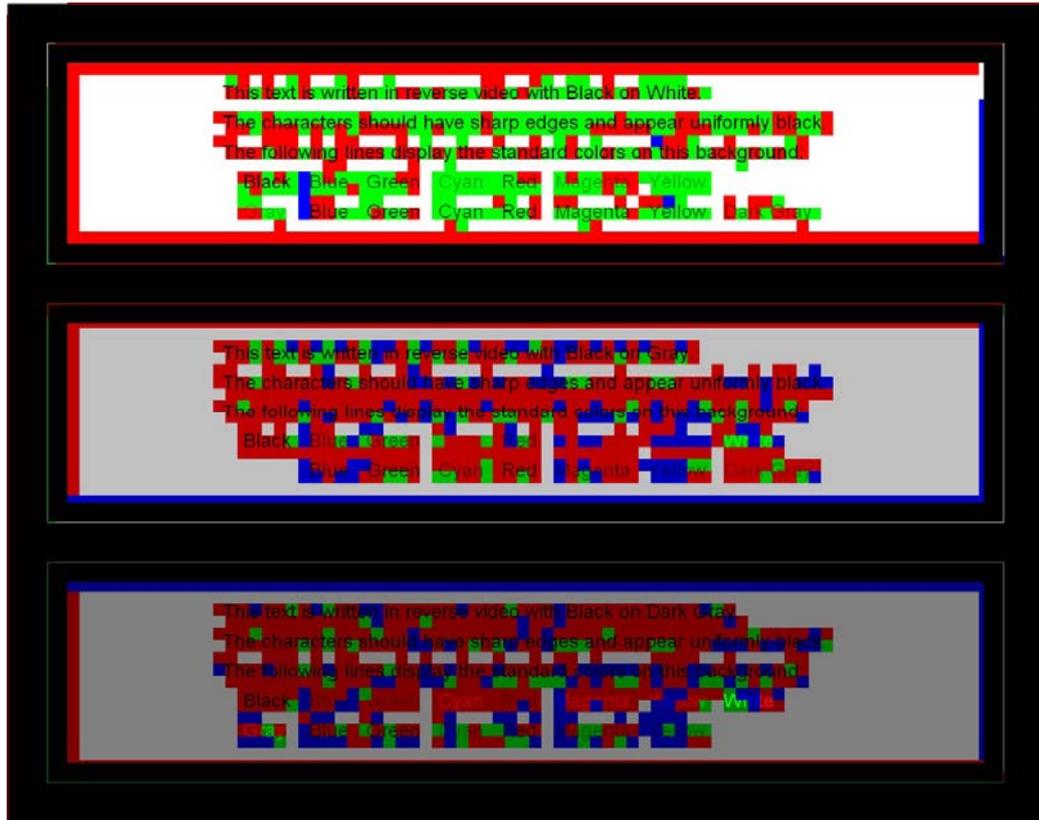


圖 2-36 以 8×8 為 window size，分析圖 2-30 以三種演算法放大的結果。

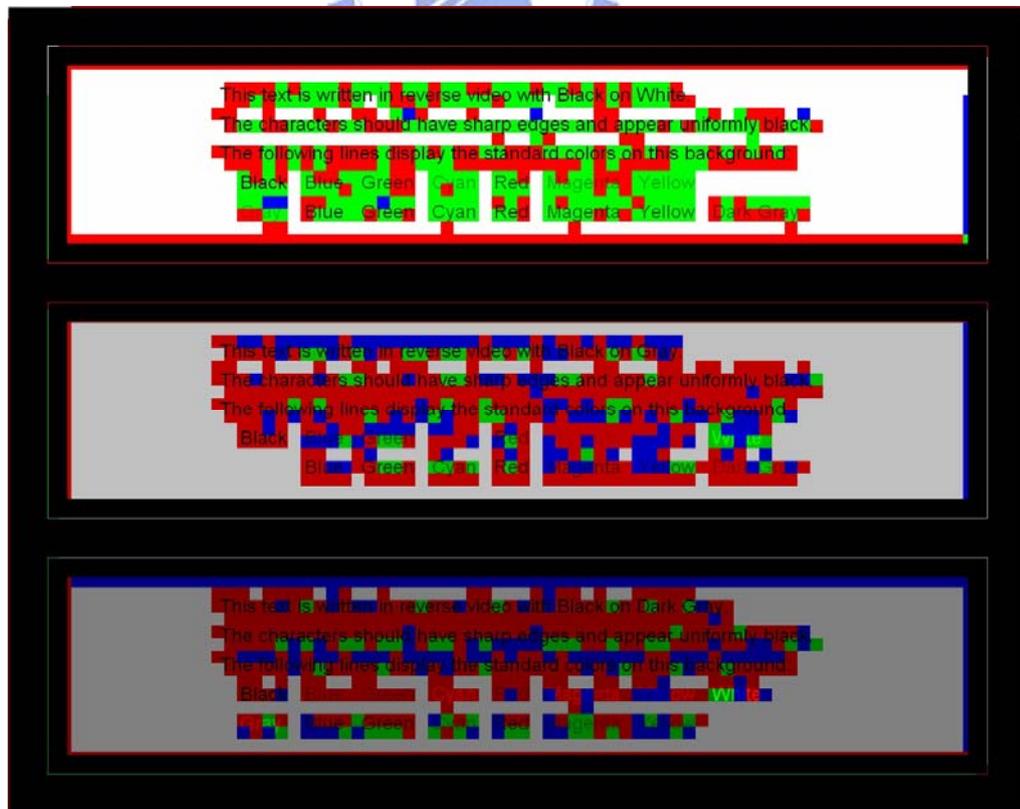


圖 2-37 以 16×16 為 window size，分析圖 2-30 以三種演算法放大的結果。

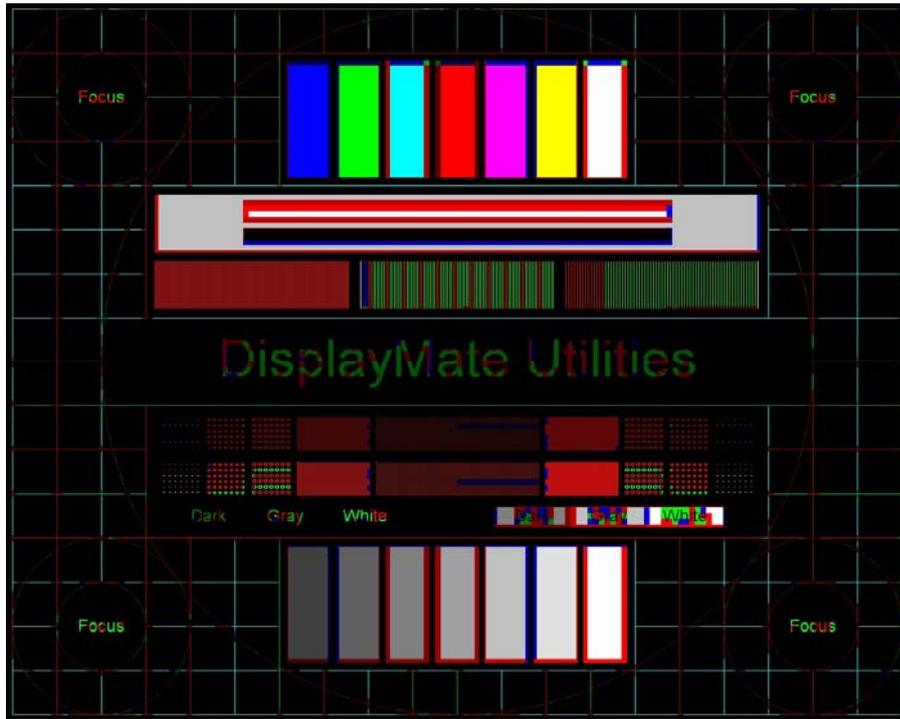


圖 2-38 以 8×8 為 window size，分析圖 2-32 以三種演算法放大的結果。

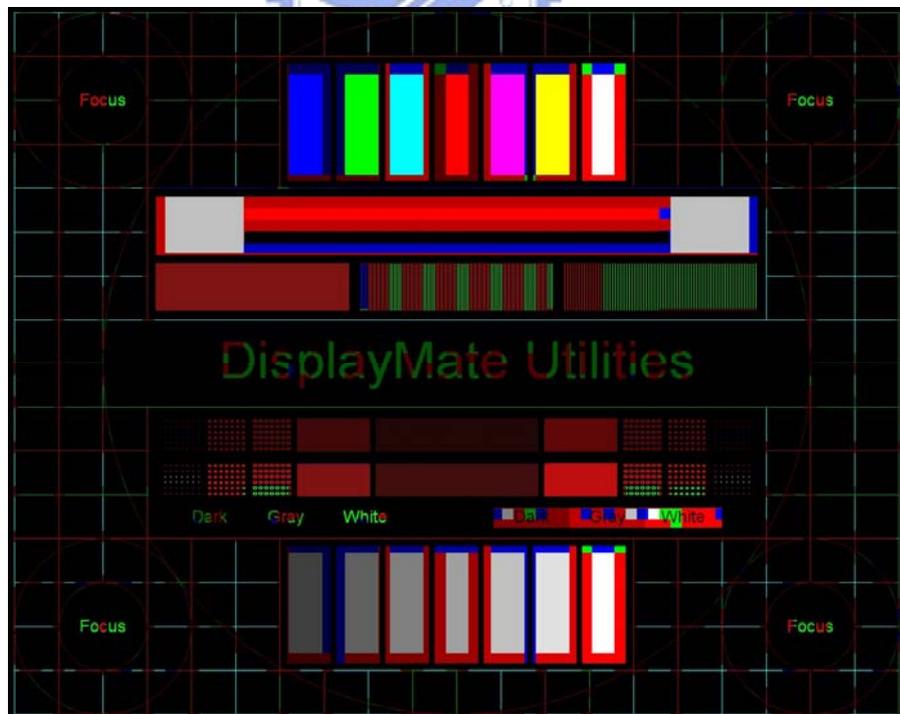


圖 2-39 以 16×16 為 window size，分析圖 2-32 以三種演算法放大的結果。

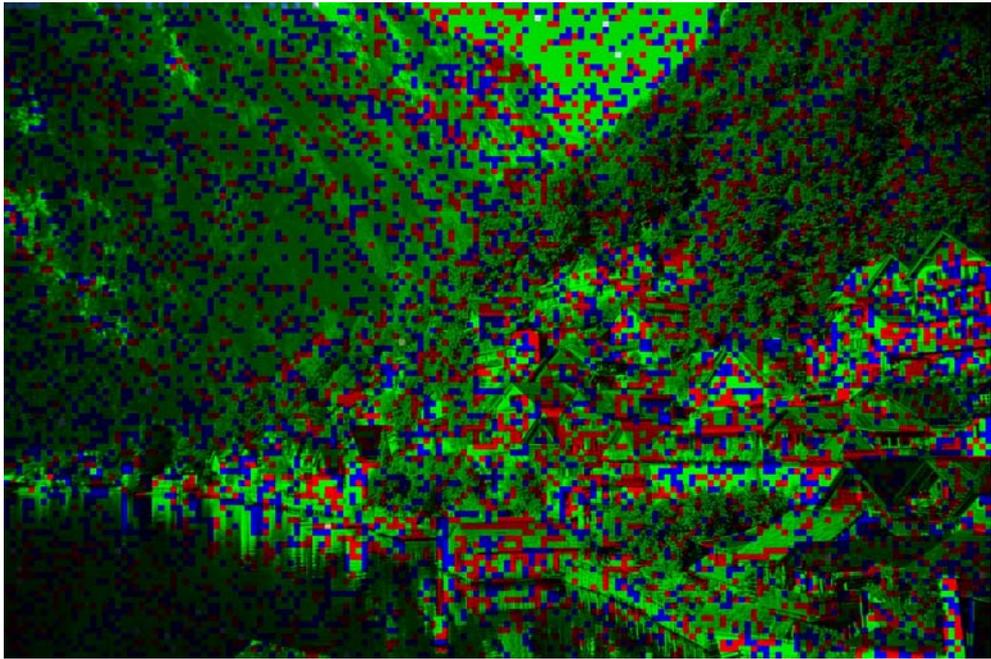


圖 2-40 以  $8 \times 8$  為 window size，分析圖 2-33 以三種演算法放大的結果。

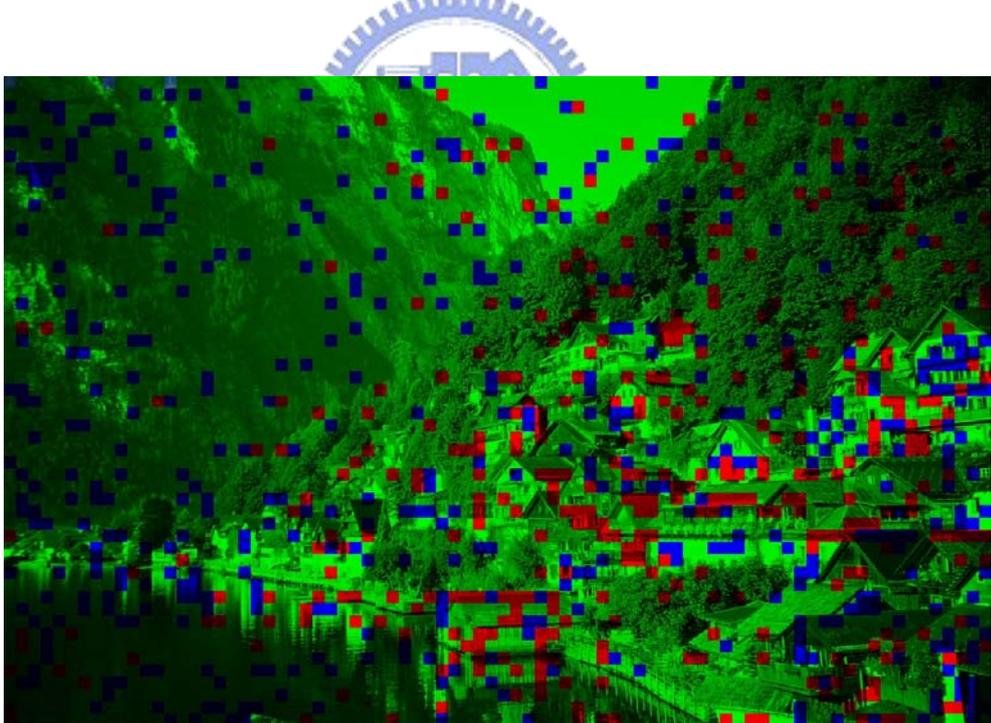


圖 2-41 以  $16 \times 16$  為 window size，分析圖 2-33 以三種演算法放大的結果。

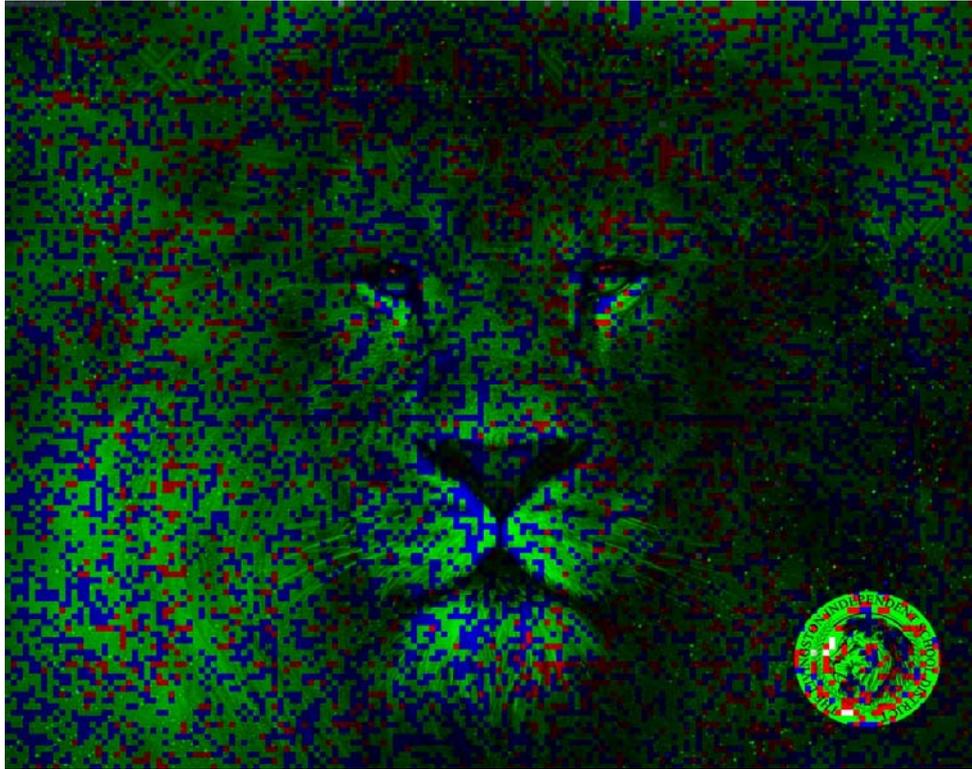


圖 2-42 以  $16 \times 16$  為 window size，分析圖 2-34 以三種演算法放大的結果。

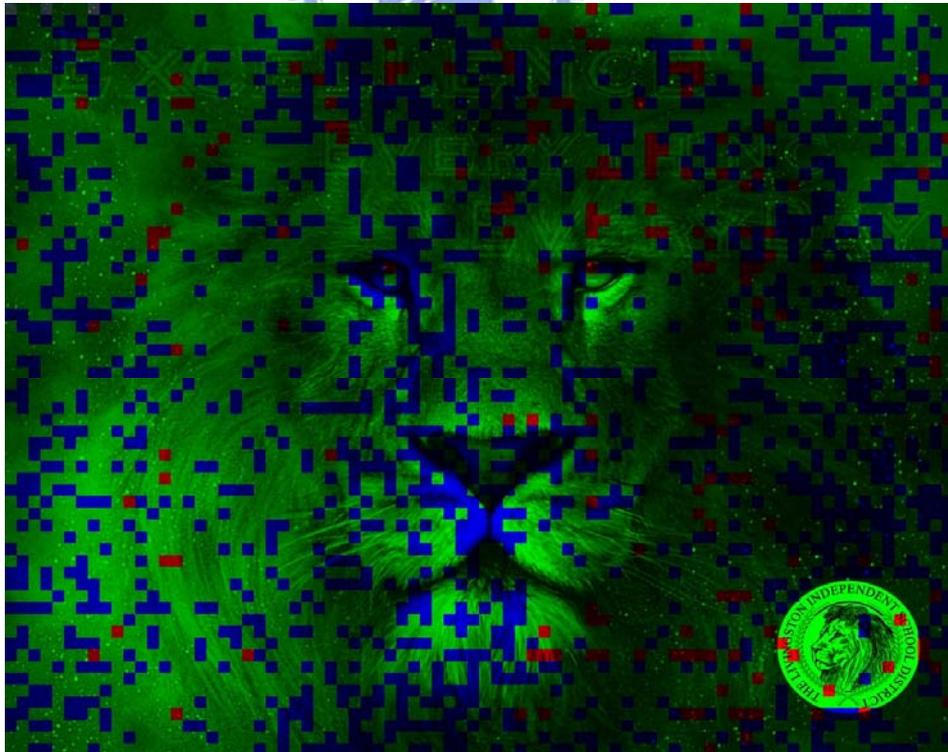
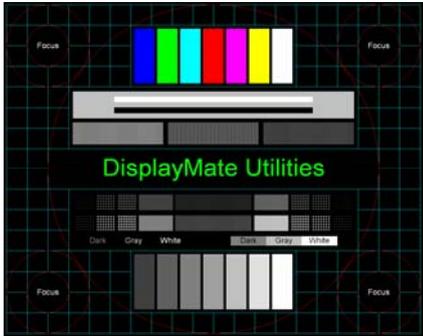


圖 2-43 以  $16 \times 16$  為 window size，分析圖 2-34 以三種演算法放大的結果。

表 2-5 實驗測試影像之比較表。

	
Text	Text
Nearest Neighbor	Nearest Neighbor
	
Landscape (Natural)	Landscape2 (Natural)
Bi-Cubic	Bi-Cubic
	
Animal (Natural)	Lenna (Natural)
Bi-Cubic	Bi-Cubic

最後，我們大約可以歸納出，以上所介紹的內插補點函數核心在其處理後影像之效果差異及在軟體及硬體實作上的考量(表 2-6)。

表 2-6 內插補點函數核心在其處理後影像之效果差異及在軟體及硬體實作上的考量。

內插核心函數	處理後影像效果	軟體實作之複雜度	軟體實作之暫存記憶體(cost)考量
Sinc	Ideal/Ringing	高	與輸入影像之解析度有關
Nearest Neighbor	Blocking Effect	最低	2 條 line buffer
Linear/Bi-Linear	Smooth	低	3 條 line buffer
<b>Cubic/Bi-Cubic</b>	<b>Clear</b>	<b>中</b>	<b>5 條 line buffer</b>



### 三、 Bi-Cubic 內插運算之硬體實作探討

在先前的章節中，我們討論了 Bi-Cubic 演算法的相關內容，而在此，我們則是著重在討論如何將 Bi-Cubic 演算法，以硬體方式實作的相關議題，其中主要包含了「定點模式 (Fixed Point Model) 運算之設計」與「數值運算精確度探討」兩個主題。

#### 3.1 Fixed Point Model 之設計

當以硬體實作數值運算演算法時，首先會考量的問題，即是能否以整數運算的方式，來表達浮點數的運算結果，此一部分即為 Fixed Point Model 的設計。對 Bi-Cubic 影像內插的運算而言，當我們採用 Fixed Point Model 進行硬體實作時，需要考量兩個部分，一是在建構 Bi-Cubic 內插係數的 Look-up Table 時，需要以多高精確度的 Fixed Point Model 來表示這些係數的數值；二是在作 Bi-Cubic 內插運算的過程中，需要進行哪些 Fixed Point Arithmetic 運算，方能正確的計算出最後的內插數值；對於這兩個部分的問題，我們將分別以一維影像放大的實例作討論，說明由解析度 240 放大到 1024 的過程，其定點(Fixed Point)計算的方法。

#### 3.2 Look-up Table 之建構

以硬體實作 Bi-Cubic 放大時，我們採用如圖 2-22(b)所示的取樣方式。由簡單的分數化簡可知，當一維影像解析度由 240 放大到 1024 時，兩個數值的比例為： $\frac{240}{1024} = \frac{15}{64}$ ，

這就表示了，我們需要建立的每個 Look-up Table，原則上是以 64 個數值為單位。再者，由於 Bi-Cubic 內插運算的係數有正負兩個部分，因此我們共需要建立兩組的 Look-up Tables，來處理一維的內插運算，此一內插比例的係數分佈如圖 3-1 所示，其中 64 個紅

點表示數值為正的係數，64 個藍點表示數值為負的係數；若我們仔細觀察此一分佈，會發現正負兩個 Look-up Tables 的第一個係數值，分別為 1 和 0，在內插的運算過程中，會用到這兩個係數計算出來（放大影像上）的內插點，其實是對應到原始輸入影像的像素點，故在硬體設計上，可省略此一不必要的內插計算，而直接將原始像素點的顏色值，複製到對應的內插放大影像上即可，也因此，這個兩值在實際建構 Look-up Tables 時，是不需要儲存的，故對於一維的內插運算處理，兩個 Look-up Tables 共需要儲存  $63 \times 2 = 126$  個係數值。此外，在建構係數為負的 Look-up Table 時，我們並非儲存負數（二補數）的數值，而是儲存其絕對值的量值部分，此一設計，有利於在硬體實作時，減少一個 Signed Bit 的儲存與運算處理。

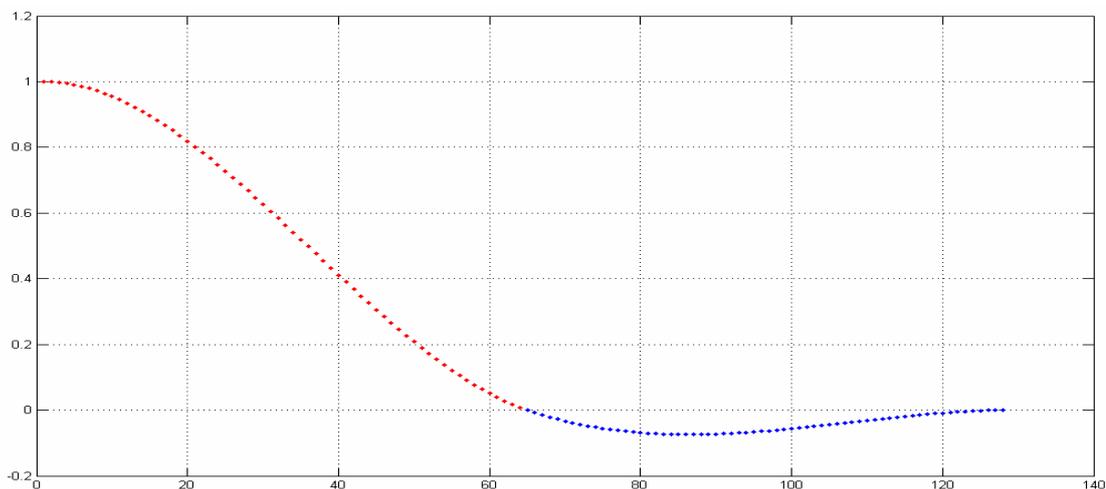


圖 3-1 浮點內插係數分佈圖。紅點表示係數值為正，藍點表示係數值為負。

建構 Lookup Table 的第二個步驟，我們的作法就是將圖 3-1 係數分佈的浮點數值（Floating Point Values），轉換成定點數值（Fixed Point Values）。在此，我們可以傳入參數“PRECISION”來選擇 Fixed Point Value 的精確度，分別為 8、12 或 16 位元，使得我們可以對不同精確度的情況下，建構 Look-up Tables 與進行 Bi-Cubic 內插運算，以進行不同精確度的誤差分析。由於所需要轉換儲存的內插係數數值皆小於 1，所以轉換後的 Fixed Point Values，其表示方式（以 8 位元精確度為例）為 .XXXXXXXX 的格式，裡頭所有的位元皆用來表示內插係數的小數部分（無整數與 Signed Bit 的部分）。這樣的作法亦常見於一般硬體實作上，並可節省定點數值所佔的記憶體空間。

### 3.3 Look-up Table 容量大小

以硬體實作 Bi-Cubic 影像放大時，對於每一種解析度轉換，我們都需要查表得出轉換運算的係數值，而當我們需要對多種不同的解析度進行影像放大時，則總共需要建構多大容量的 Look-up Table 才足夠呢？對此我們以一般常用到的水平方向的放大例子作討論，常見的水平方向解析度轉換包含了從 640/720/800/1024/1152 轉換 1280，對於這些解析度轉換，我們可以作分數化簡分析如：

$$\frac{640, 720, 800, 1024, 1152}{1280} = \frac{40, 45, 50, 64, 72}{80}$$

其中，我們可以看到分母的部分為 80，亦即只要透過建構一個容量大小為 $(80-1) \times 2$ 個單位的 Look-up Table，便可以將這些不同的解析度轉換到 1280；不過，在此我們還可以作更進一步的容量縮減，若考慮下列分數化簡：

$$\frac{640, 720, 800, 1024, 1152}{1280} = \frac{40, 45, 50, 64, 72}{80} = \frac{1}{2}, \frac{9}{16}, \frac{5}{8}, \frac{4}{5}, \frac{9}{10} = \frac{8, 9, 10}{16}, \frac{8, 9}{10}$$

我們可以發現，我們只需要用一個容量大小為 $(16-1) \times 2$ 的 Look-up Table，加上另一個容量大小為 $(10-1) \times 2$ 的 Look-up Table，便可以作上述的水平解析度放大，如此可以減少建構 Look-up Tables 所需的記憶體空間，同理，我們也可以依照這樣的分析方式，探討垂直方向解析度轉換所需的 Look-up Table 容量。

### 3.4 Fixed Point Model 之內插運算

在作內插運算時，以圖 3-2 所例，假設  $x$  為放大影像需求出的內插點； $x_0$ 、 $x_1$ 、 $x_2$ 、 $x_3$  分別為內插放大影像於原始影像的參考點，每個放大內插點的間隔為  $\frac{1}{64}$ ，若要計算

內插點  $x$  (與  $x_0$ 、 $x_1$ 、 $x_2$ 、 $x_3$  距離分別為  $\frac{75}{64}$ 、 $\frac{11}{64}$ 、 $\frac{53}{64}$ 、 $\frac{117}{64}$ ) 的顏色值，其算式為：

$$I(x) = I(x_0)w^-(10) + I(x_1)w^+(10) + I(x_2)w^+(52) + I(x_3)w^-(52) \quad (3.1)$$

其中， $I(x)$  表示  $x$  位置的原始影像顏色值，而  $w^+(\cdot)$  與  $w^-(\cdot)$  則為正負兩個 Look-up Tables 的係數值。而對於  $I(x_1)$  所相乘的係數  $w^+(10)$ ，其係數 Index 則可由  $[x_1 \times 64] - 1 = 10$  此一簡單的運算得出，需注意的是，此一 Index 計算中需要額外減 1，是因為在先前所建構的 Look-up Table 中，只儲存了 63 個數值的緣故，而另一個  $w^+(\cdot)$  Look-up Table Index 的值，則為  $63 - 10 - 1 = 52$ ；同理，我們也可以將這兩個 Indices 代入  $w^-(\cdot)$  Look-up Table 中，可直接得到其係數值，之後便可以將這 4 個係數值，和顏色值  $I(\cdot)$  進行相乘與加總的運算。

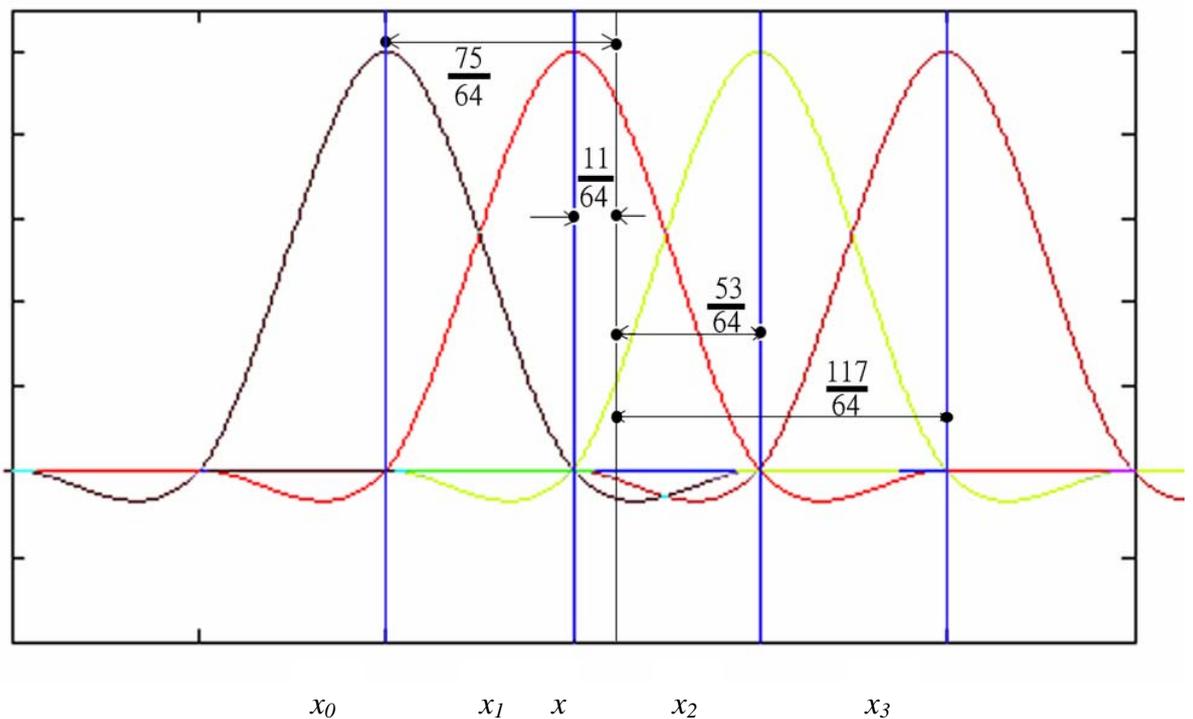


圖 3-2 Cubic convolution 內插運算示意圖

接下來，我們將對於 Fixed Point Model 內插運算中，乘法和加法的運算作說明。首先是乘法的部分，如圖 3-所示，乘法的運算是將 8 位元整數的顏色值  $I(\cdot)$  和 Fixed Point Value 的小數係數值  $w(\cdot)$  作相乘 (以圖 3-所示為例， $w(\cdot)$  的精確度為 8 位元)，所以我們

需要將  $I(\cdot)$  和  $w(\cdot)$  的數值，置放到  $8 \times 2 = 16$  位元的暫存器中，在進行整數乘法的運算，乘完後傳回 16 位元的 Fixed Point Value，其表示方式為 XXXXXXXX.XXXXXXXX 的格式。接下來則是簡單的 16 位元整數加法，需要將四個乘積值，依照其對應的正負號，先作兩個負的乘積值加法，再依序加上兩個正的乘積值，如此相加的結果即可得到內插點的亮度值圖 3-3 則為我們以軟體模擬此一運算的程式片段，其中的加法為了能夠符合不同精確度 (8/12/16 位元) 的計算，所以我們使用了 32 位元帶號整數 (long) 的加法，也因此模擬實作上，不需要考慮負數先加的順序。

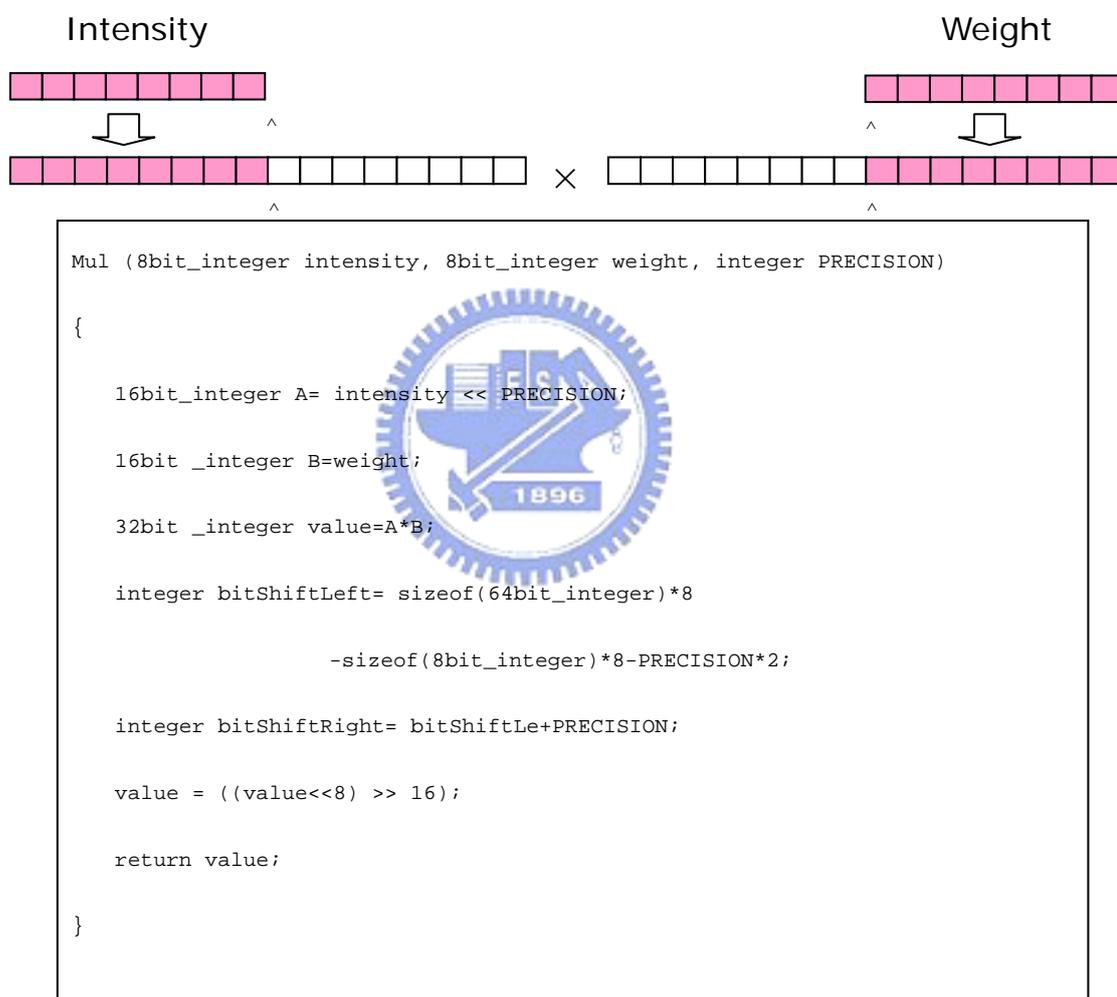


圖 3-3 Fixed Point Model 乘法運算之程式(Pseudo code)。

### 3.5 內插係數和之探討

在建構出 Fixed Point Look-up Table 之後，有一個需要注意的問題是，在影像內插放大的運算中，其四個係數的總和（理論上）皆必須為 1，其數學式為：

$$w(s_0) + w(s_1) + w(s_2) + w(s_3) = 1 \quad (3.2)$$

然而由於所建構出的 Fixed Point Look-up Table 其精確度可能會有誤差，其中  $\hat{w}(\cdot)$  為 Fixed Point 格式的內插係數，其總和可能大於、小於或剛好等於 1；若我們希望強制使得係數總和為 1，則可以在係數和的算式中補上一個差值  $\hat{w}_e$ ，因此係數和的算式可改寫成：

$$\hat{w}(s_0) + \hat{w}(s_1) + \hat{w}(s_2) + \hat{w}(s_3) + \hat{w}_e = 1 \quad (3.3)$$

亦即只要我們可以將  $\hat{w}_e$  補在原本 Fixed Point 係數  $\hat{w}(s_0)$ 、 $\hat{w}(s_1)$ 、 $\hat{w}(s_2)$ 、 $\hat{w}(s_3)$  的任一個，便可以達到維持係數總和為 1 的效果。在學理上，若我們將  $\hat{w}_e$  補在四個係數中絕對值最大的那一個，即補在：

$$\max(\hat{w}(S_0), \hat{w}(S_1), \hat{w}(S_2), \hat{w}(S_3)) \quad (3.4)$$

之中，則可以使得該係數對於運算結果的影響程度

$$\frac{\hat{w}_e}{\max(\hat{w}(S_0), \hat{w}(S_1), \hat{w}(S_2), \hat{w}(S_3))} \quad (3.5)$$

最小，依照此一原則，我們可以調整出一個可使得係數和為 1 的 Fixed Point Look-up Table，此一方法我們稱為補值方法 A（以下稱為 WS-A）。以先前解析度由 240 放大到 1024 為例，其調整過的兩個 Look-up Tables 分佈如圖 3-所示，其中綠色點為初始計算出來的 Fixed Point 係數值，而兩個黑色×點則表示 Look-up Tables 所不儲存的兩個整數係數，而紅色與藍色三角形點則表示原來綠點係數值，需要調整到紅點或藍點的位置，方能夠使得內插運算的係數總和維持在 1。由該圖可以看出，差值  $\hat{w}_e$  皆是補在正的 Look-up Table 中， $\hat{w}(\cdot)$  數值較大的半部，表 3-1 為 WS-A 方法所改變係數值之比較。

另一種補償係數和之差值的方式，所考慮的算式如下所示：

$$I(x_0)\hat{w}(s_0)+I(x_1)\hat{w}(s_1)+I(x_2)\hat{w}(s_2)+I(x_3)\hat{w}(s_3)+\hat{I}\hat{w}_e \quad (3.6)$$

我們可以在影像放大的過程中，考慮針對不同的影像亮度分佈，動態的將差值  $\hat{w}_e$  補在  $I$  值最小的數值上，亦即對於內插放大的運算，補上：

$$\hat{I} = \min(I_0, I_1, I_2, I_3) \quad (3.7)$$

的數值，使得差值  $\hat{w}$  對亮度乘積所造成的影響最小，此一方式需要動態的找出四個原始參考影像的最小亮度，因此是個較複雜的方法，我們稱之為補值方法 B (WS-B)。實際上，差值  $\hat{w}_e$  的來源，是由於  $\hat{w}(s_0)$ 、 $\hat{w}(s_1)$ 、 $\hat{w}(s_2)$ 、 $\hat{w}(s_3)$  在 Fixed Point Model 的表示法中，皆不夠精確所累積出來的誤差，因此對於在 A、B 兩種補值方法中，將差值  $\hat{w}_e$  補在單一係數或單一乘積的作法，皆屬於人為處理上的一個選擇，而無特定的好壞之分。

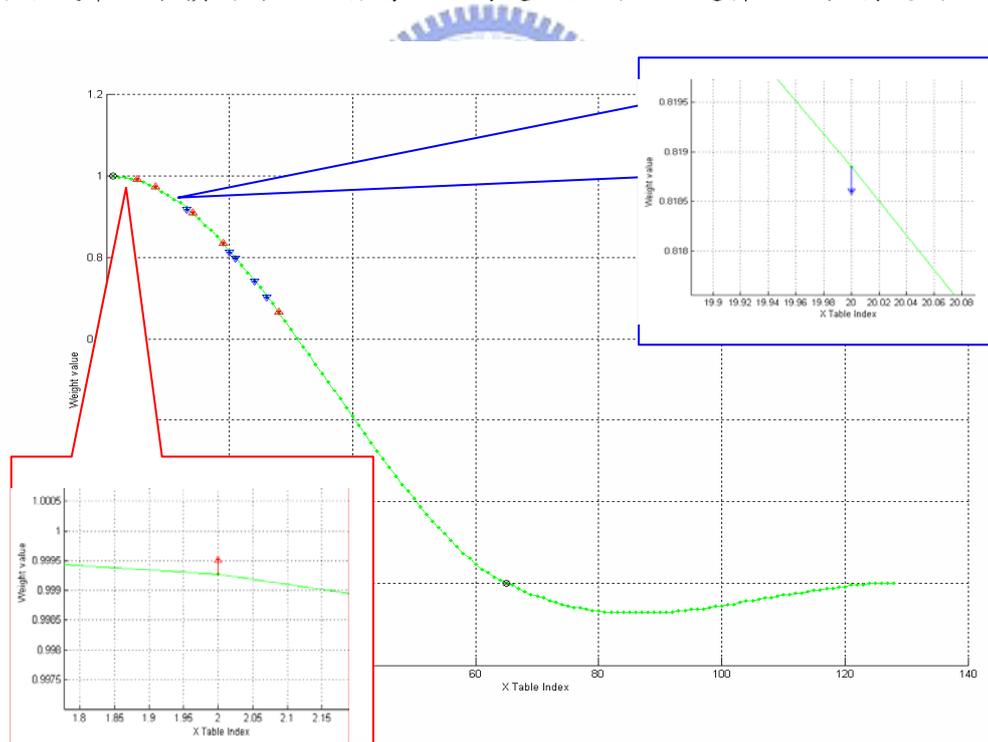


圖 3-4 調整過的 Fixed Point Look-up Tables。其中綠色點為原始的 Fixed Point 數值，紅色三角點則為調升過後的 Fixed Point 數值，藍色三角點則為調降過後的 Fixed Point 數值，以此調整過的 Look-up Tables 作內插放大運算，可使 Bi-Cubic 內插運算的任四個係數和皆為 1。

表 3-1 Fixed-point 與 WS-A 權重(Weight)比較表格。

index	4	7	12	13	18
fixed-point weight	0.988	0.969	0.922	0.906	0.832
WS-A weight	0.992	0.9723	0.918	0.910	0.836
index	19	20	23	25	27
fixed-point weight	0.816	0.801	0.746	0.707	0.664
WS-A weight	0.813	0.797	0.742	0.703	0.668

## 3.6 數值運算精確度探討

在 Fixed-Point Model 的實驗中，以圖 3-的影像為例，我們將此一灰階人物影像由原來的 240×320 放大到 1024×768 此一非整數倍數的解析度，同時探討在 Fixed-Point Model 中，使用 8/12/16 位元的精確來模擬硬體影像放大，其結果與 Floating Point Model 的差異為何。對此，我們比較了不同精確度與是否採用補償 Weight Sum 不足 1 共六種方法，其中補償 Weight Sum 不足 1 的實作方法中，又可分為補值方法 A (WS-A) 和補值方法 B (WS-B) 兩種，我們分別統計其 Fixed Point Model 的影像放大結果與 Floating Point Model 計算結果的統計量，包含了最大差值、平均差值與差值的變動大小三種統計量，其結果參見表 3-2。

對於各類不同影像(如圖 3-到圖 3-10)，其 Fixed Point Model 與 Floating Point Model 的各種差值統計分析實驗結果，則如表 3-2 到表 3-7 所示。我們歸結其數據比較結果，可以看到無論是採用 WS-A 或 WS-B 的方法，其差值皆比不作補值方法的情況來得好，也就是說補償 Weight Sum 不足 1 的作法是必須的，而對於 WS-A 與 WS-B 兩種方法的比較，則沒有一定的好壞，此一結果符合我們的預期。再者，若在沒有作 Weight Sum 補償的情況下，增加 Fixed Point Model 計算的位元數，如表 3-2 到表 3-5 所示，大致可以看出差值變小的趨勢；然而若是在有作 Weight Sum 補償的情況下，當我們增加 Fixed

Point Model 計算的位元數，進行不同位元精確度的影像放大，其差值的大小變化則沒有一定的趨勢，這是因為在作 Weight Sum 補償時，我們刻意改變了原來 Bi-Cubic 的內插運算方式，以人為的方式增補或減少小部分的內插數值，故此時位元精確度已不是主要影響內插結果的因素。若我們以圖例作進一步的觀察，如以自然影像的放大來看（如表 3-2 到表 3-5），在有作 Weight Sum 補償的情況下，8 位元精確度看來已經足夠，但若以高反差的影像放大來看（如表 3-6 與表 3-7），12 位元精確度會是一個比較適合的選擇。





圖 3-5 差值分析所採用之灰階人物影像 (240×320 放大到 1024×768)。

表 3-2 以灰階人物影像(圖 3-5)作 Fixed Point 與 Floating Point 放大之差值統計表。

$\Delta$	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	5.000000	2.000000	2.000000	3.000000	2.000000	2.000000	3.000000	2.000000	2.000000
Ave.	1.667441	1.024532	0.950471	0.917901	0.925133	0.925096	0.902248	0.925838	0.924979
Std.	0.697315	0.544652	0.542882	0.546806	0.542156	0.542365	0.545131	0.542144	0.542361



圖 3-6 差值分析所採用之彩色風景影像 (320×214 放大到 1024×768)。

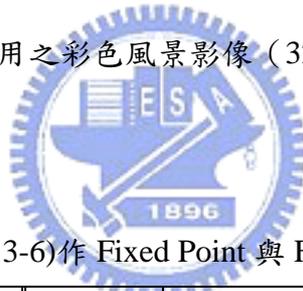


表 3-3 以彩色風景影像(圖 3-6)作 Fixed Point 與 Floating Point 放大之差值統計表。

Δ	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	8.000000	7.666667	7.666667	7.666667	7.666667	7.666667	7.666667	7.666667	7.666667
Ave.	1.772376	1.030220	0.965732	0.948574	0.960132	0.960723	0.954169	0.960729	0.960707
Std.	0.730598	0.530344	0.527321	0.541111	0.527263	0.527230	0.542146	0.527267	0.527230



圖 3-7 差值分析所採用之彩色建築物影像 (640×512 放大到 1200×1000)。

表 3-4 以彩色建築物影像(圖 3-7)作 Fixed Point 與 Floating Point 放大之差值統計表。

$\Delta$	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	10.333333	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
Ave.	1.672319	1.038270	0.978705	0.935284	0.941588	0.942283	0.912530	0.940994	0.941891
Std.	0.638821	0.543343	0.539863	0.553560	0.537805	0.537696	0.549517	0.537676	0.537655



圖 3-8 差值分析所採用之彩色動物影像（640×512 放大到 1200×1000）。

表 3-5 以彩色動物影像(圖 3-8)作 Fixed Point 與 Floating Point 放大之差值統計表。

$\Delta$	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	11.666667	11.333333	11.333333	11.666667	11.333333	11.333333	11.666667	11.333333	11.333333
Ave.	1.672805	1.017782	0.969907	0.953216	0.957990	0.958414	0.924221	0.956689	0.957943
Std.	0.632781	0.521097	0.518133	0.542380	0.517750	0.517676	0.536835	0.517663	0.517647

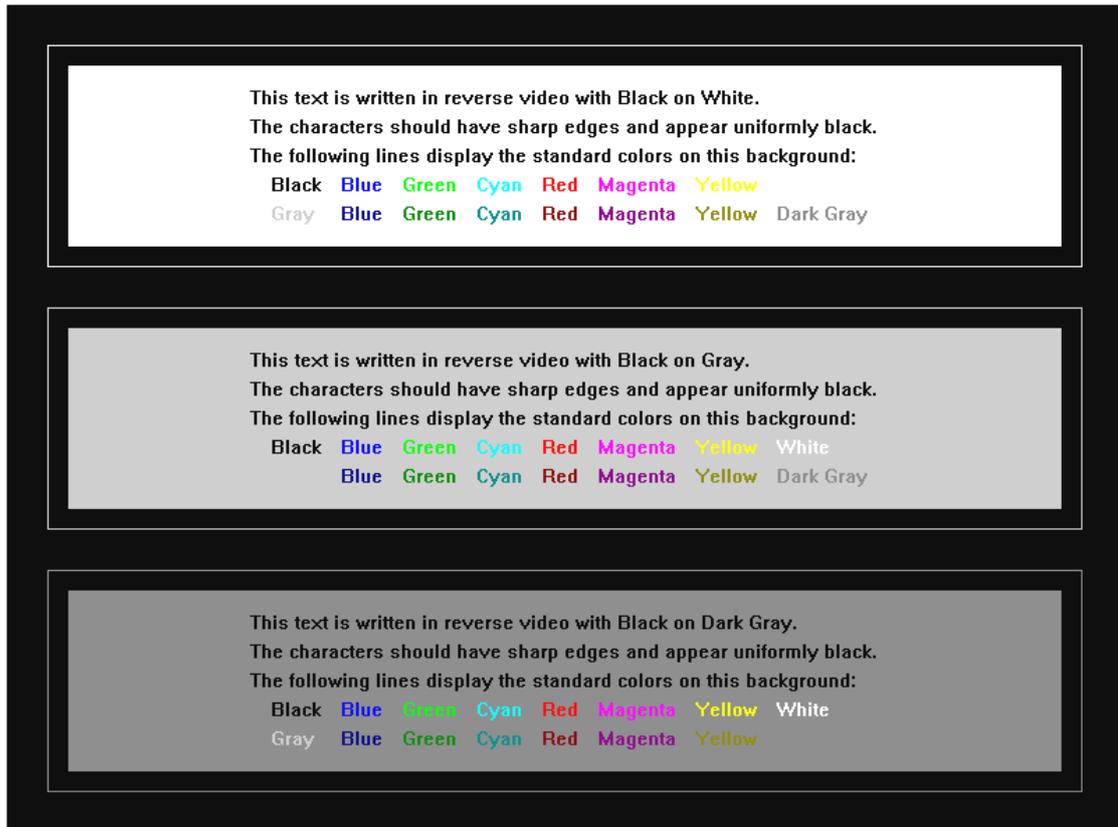


圖 3-9 差值分析所採用之文字模式影像（800×600 放大到 1024×768），此一影像包含了邊框、高對比的區塊與彩色文字。

表 3-6 以文字模式影像(圖 3-9)作 Fixed Point 與 Floating Point 放大之差值統計表。

$\Delta$	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	28.000000	28.000000	28.666667	28.666667	28.666667	28.666667	28.666667	28.666667	28.666667
Ave.	1.278334	0.881012	0.132401	0.139866	0.130497	0.132401	0.141374	0.130497	0.132401
Std.	1.361419	1.069023	0.768553	0.787627	0.767619	0.768553	0.788569	0.767619	0.768553

(註記：由於測試影像包含高對比的區塊與文字，故此表中 Max 的差值較大。)

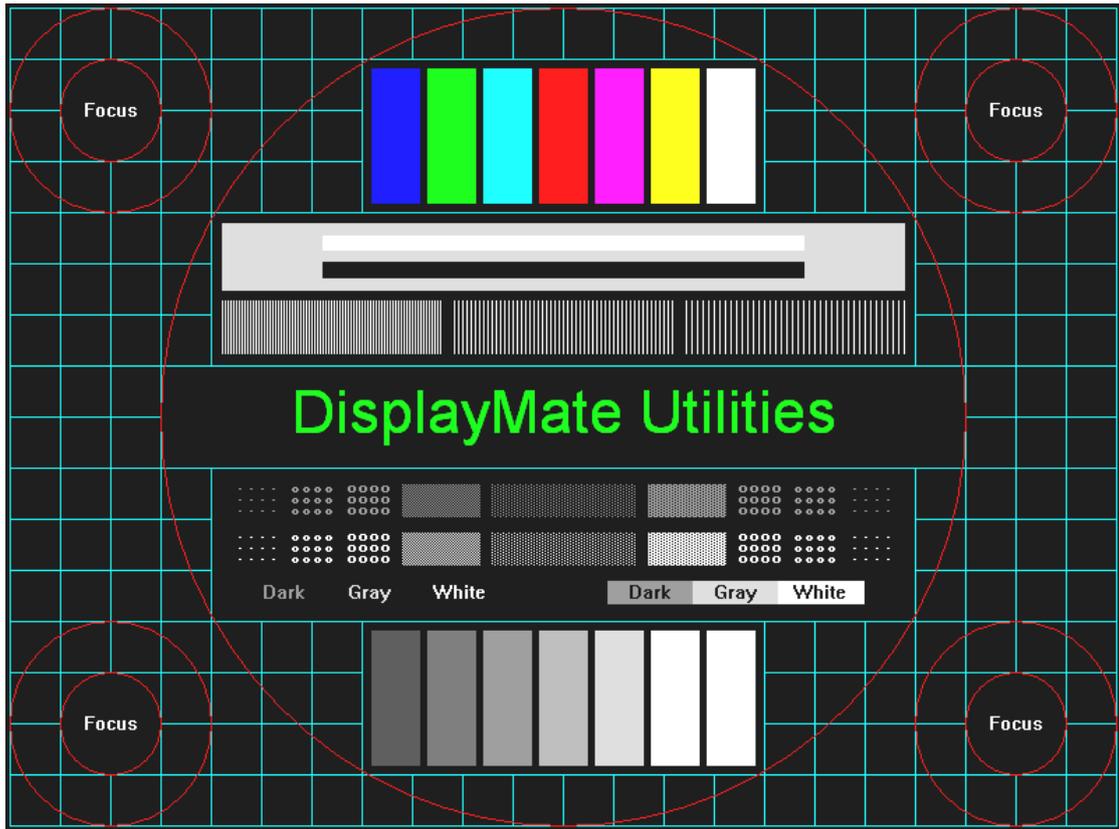


圖 3-10 差值分析所採用主要測試影像(800×600 放大到 1024×768)，此一影像包含邊框、圓框、高對比區塊、彩色文字、不同的圖樣分佈與灰階變化。

表 3-7 以主要測試影像(圖 3-10)作 Fixed Point 與 Floating Point 放大之差值統計表。

$\Delta$	8-Bit	12-Bit	16-Bit	8-Bit+	12-Bit+	16-Bit+	8-Bit+	12-Bit+	16-Bit+
				WS-A	WS-A	WS-A	WS-B	WS-B	WS-B
Max	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	29.000000	30.000000	30.000000
Ave.	0.500901	0.337772	0.120993	0.151871	0.122964	0.120993	0.154383	0.122964	0.120993
Std.	1.118656	0.888949	0.710746	0.750449	0.712640	0.710746	0.750863	0.712640	0.710746

(註記：由於測試影像包含高對比的區塊與文字，故此表中 Max 的差值較大。)

## 四、 影像縮小

本章的討論主要集中於影像縮小的處理，在此我們將先探討影像縮小的基本概念，接著，如放大處理核心函數一般，相同地我們引進縮小處理的核心函數概念，並進一步討論各種縮小演算法的效能差異。在實作上，我們除了再次沿用與影像放大研究相同的 Bi-Cubic 演算法外，以數位訊號處理的觀點，另外還實作了 “Local Averaging” 縮小演算法，並比較兩者影像縮小的效果。

### 4.1 影像縮小觀念說明

以單一維度(1D)的影像訊號來說，無論在空間域 (Spatial Domain) 或是在頻率域 (Frequency Domain) 上，都具有相同的取樣寬度大小，如影像在空間域有 2048 個像素寬，則經過頻率轉換後的頻寬，也有 2048 個寬度範圍，因此若要將影像在空間域上縮小，必然在頻率域上也要縮小至相同大小；對於縮小的處理，如參考文獻[19]所指出的普遍標準作法，是利用一個理想的低通濾波器 (Ideal Low Pass Filter)，將影像的高頻成份去除留下低頻的成份，如圖 4-1 所示，當一維度影像從 2048 縮小至 1024 時，在頻率域中可視為將原始的頻率分佈乘以低通濾波器，此低通濾波器頻率範圍是從 -512 到 512 且其大小為 1.0，如此便可以在頻率域上得到我們所需要的縮小影像，此一過程對應到空間域上，則是將原始影像與一個 Sinc 核心函數 (如圖 4-2 所示) 作迴旋積 (Convolution)，結果即是縮小後的影像。

在上述的影像縮小作法中，我們可以觀察到，理想低通濾波器的頻率範圍會隨著影像縮小的倍率而變動，在空間域中，低通濾波器所對應的 Sinc 函數，其寬度也會隨著縮小的比例而變動 (見圖 4-2)，舉例來說，當頻率域的訊號透過低通濾波縮小一半時，在空間域中，對應於低通濾波器的 Sinc 函數，其寬度則為兩倍。由於理想低通濾波器與 Sinc 函數，在硬體實作上難以實現，所以我們改採用其他與 Sinc 函數相近的核心函數，

作為影像縮小的處理，在下面章節中，我們將作不同縮小核心函數的說明與比較。

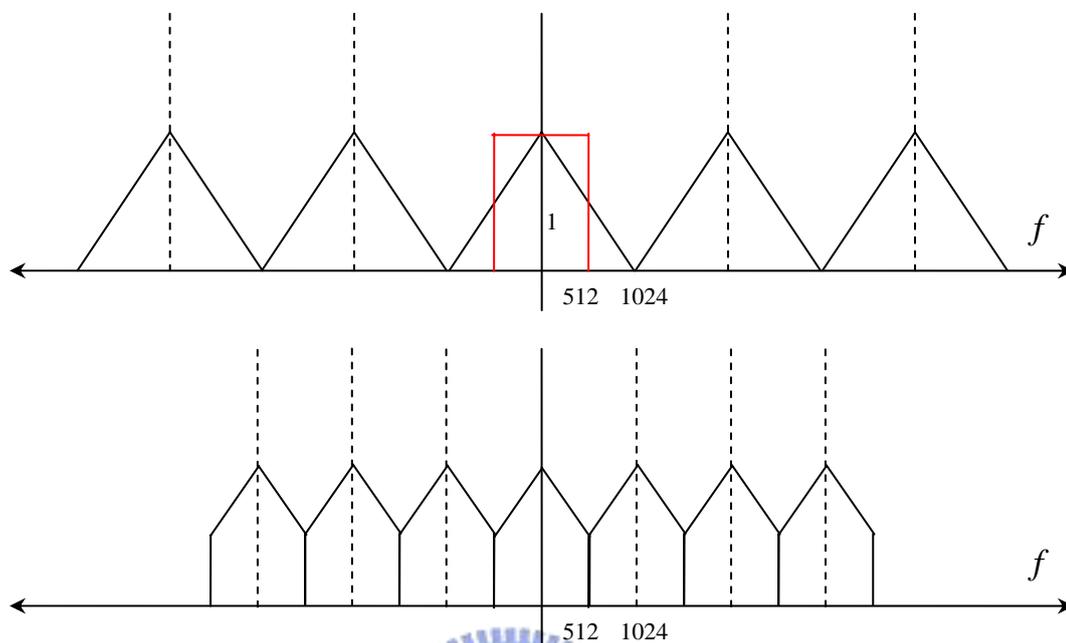


圖 4-1 一維度頻率域影像縮小，其中紅線為低通濾波器。

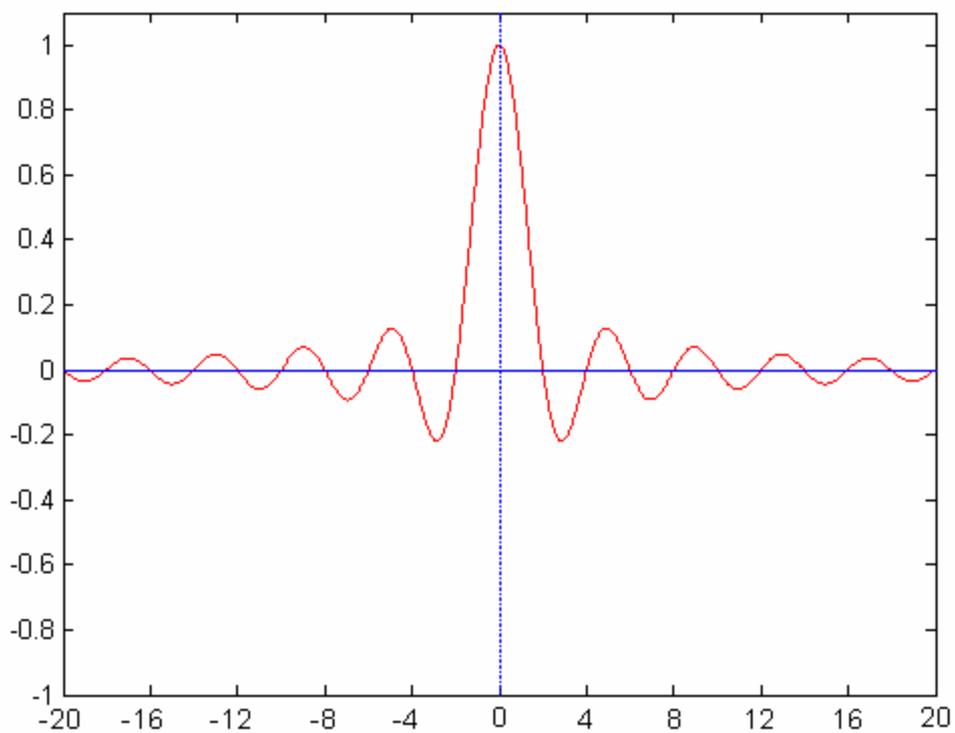


圖 4-2 空間域上的 Sinc 函數

## 4.2 Bi-Cubic 影像縮小

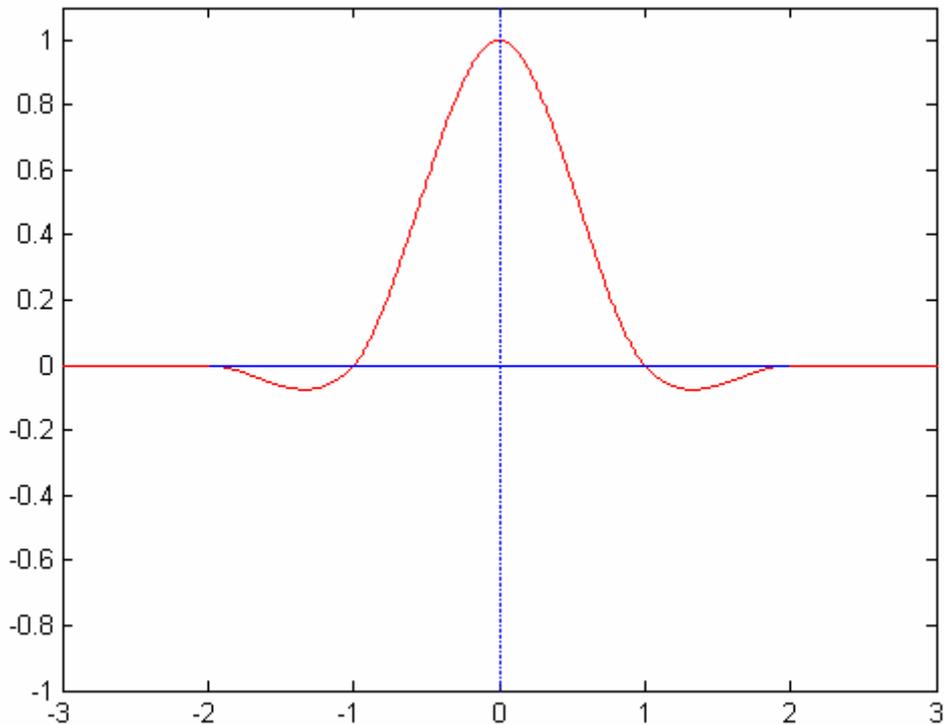
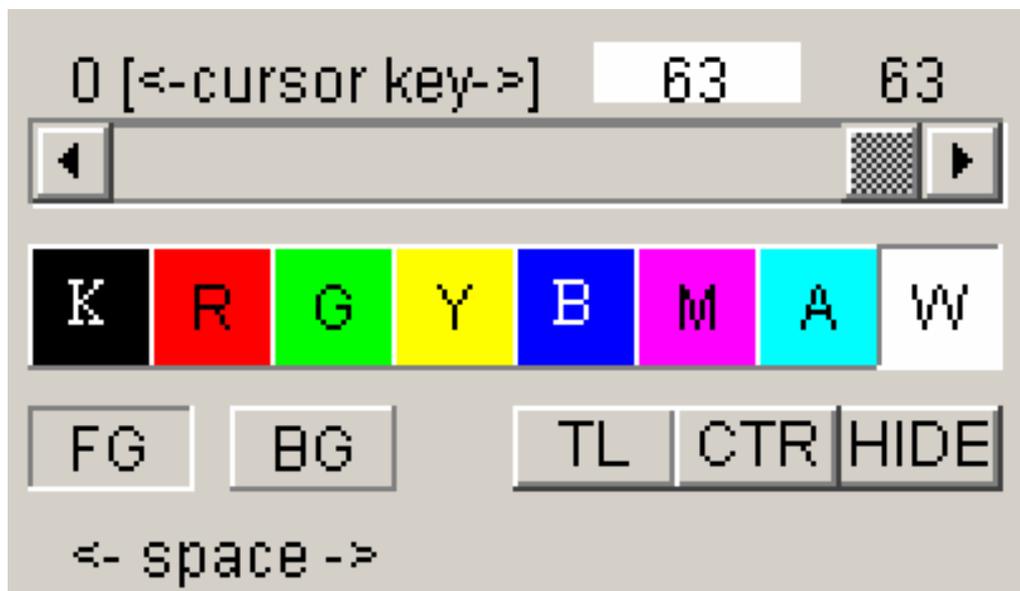


圖 4-3 影像放大所採用之 Cubic 核心函數。

如同放大演算法的處理，我們依然可以用 Cubic 核心來近似 Sinc 核心函數，然而縮小與放大不同的是，用在內插放大的 Cubic Kernel，如圖 4-3 所示，其寬度為固定的（在 -2 到 +2 之間），而用於縮小演算法的 Cubic 核心函數，函數特性相同，但寬度則是隨著縮小的比例而改變，假設（一維）縮小比例為兩倍時，則 Cubic Kernel 與橫軸的交點將會變成 -4、-2、+2、+4，也就是寬度變為兩倍；此亦為一般影像處理軟體的縮小作法。透過寬度可隨比例改變的 Cubic Kernel 作縮小運算，其二維影像縮小的結果如圖 4-4 所示，其中圖 4-4 (a) 為原始影像內容，圖 4-4 (b) 為 Photoshop 縮小處理結果，其縮小比例為原圖由 1024×768 縮小到 640×480，縮小比例在 2 倍以內，圖 4-4 (c) 則是以固定寬度的 Cubic Kernel 作縮小之比較，我們可以看到，圖 4-4 (c) 在文字部分的縮小結果，雖然

顯得對比較強烈，但文字的完整性卻不如圖 4-(b)的效果，其失真程度較大。

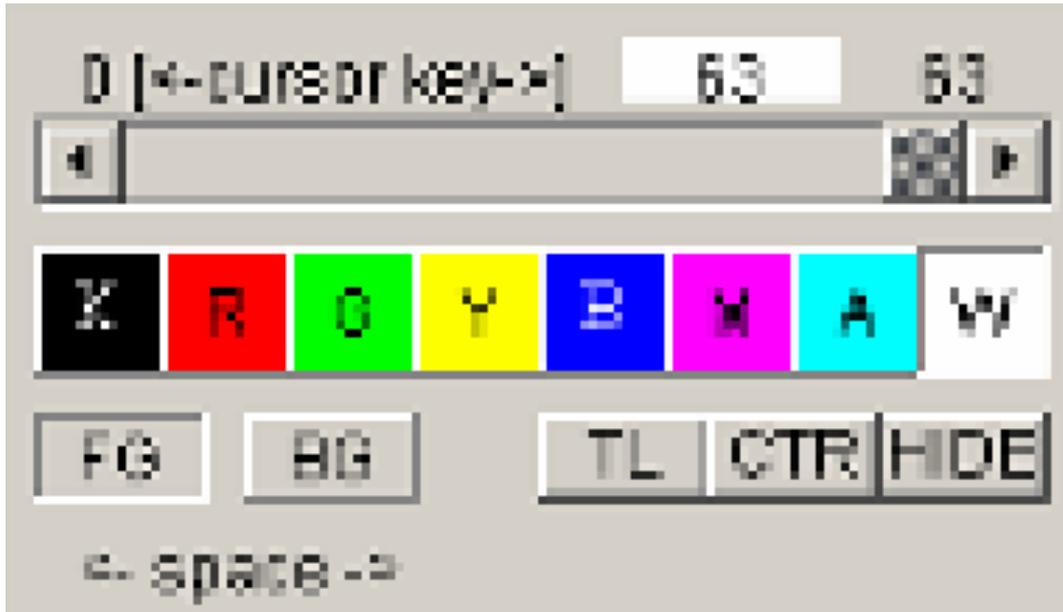


(a)



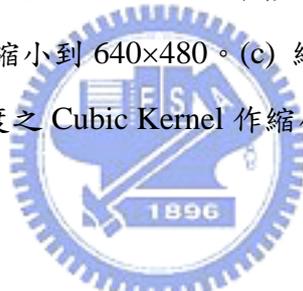
(b)

圖 4-4 以不同寬度之 Cubic Kernel 作影像縮小之比較，原始影像解析度為 1024×768，縮小到 640×480。(a) 原始影像 (1024×768) 之文字部分 (b) 縮小影像 (640×480) 之文字部分，以 Photoshop Bi-Cubic 方法作縮小。



(c)

圖 4-4 (續)以不同寬度之 Cubic Kernel 作影像縮小之比較，原始影像解析度為 1024×768，縮小到 640×480。(c) 縮小影像 (640×480) 之文字部分，以固定寬度之 Cubic Kernel 作縮小



## 4.3 Local Average 影像縮小

我們已知在作影像縮小時，需應用可變寬度的核心函數來作運算，方能夠得到比較接近原始影像的縮小結果，而對於硬體設計來說，實作可變寬度的 Cubic Kernel，其複雜度較高，因此，我們選用另一個可變寬度的核心函數，可以透過簡單的平均即可得出不錯的影像縮小效果，此一方法稱之為 Local Average。

Local Average 影像縮小的基本概念如圖 4-所示，當我們需要將一維六個點的影像訊號縮小成三個點時，則類似影像放大的情況，在考慮縮小取樣點中心位置與原始影像頭尾像素點中心對齊的取樣分佈下，我們計算每個縮小影像的取樣點，對應到原始影像點的寬度範圍（在此例中，寬度值為  $\frac{6-1}{3-1} = 2.5$ ），而後透過平均此一範圍內的原始影像

點亮度值，即可計算出縮小影像點的亮度；以第一個縮小取樣點的亮度計算為例，其亮

度值即為  $0.75 \cdot I(-1) + 1 \cdot I(0) + 0.75 \cdot I(1)$ ，其中  $I(\cdot)$  為原始影像點的亮度，為了方便邊界條件的處理，我們令超出原始影像邊界的  $I(-1) = 0$ 、 $I(6) = 0$ ，同理類推，我們便可以計算出整個縮小影像的亮度。

基於此一 Weighted Average 的處理概念，我們可以得出 Local Average 的核心函數如圖 4-5 所示，此一核心函數的寬度是隨著縮小比例的不同而改變，因此當縮小越多倍時，其寬度越寬，對應到作平均運算的原始像素點也就越多，而具有 Low-Pass Filtering 的效果，同時 Local Average 的核心函數，是一個比 Bi-Cubic 更為簡單的核心函數，故在實作上，也具有較為簡便的優勢。以圖 4-4 (a) 之影像，作 Local Average 縮小，結果如圖 4-6 所示，我們可以發現透過 Local Average 作影像縮小的效果，與 Photoshop 作 Bi-Cubic 縮小的效果類似。

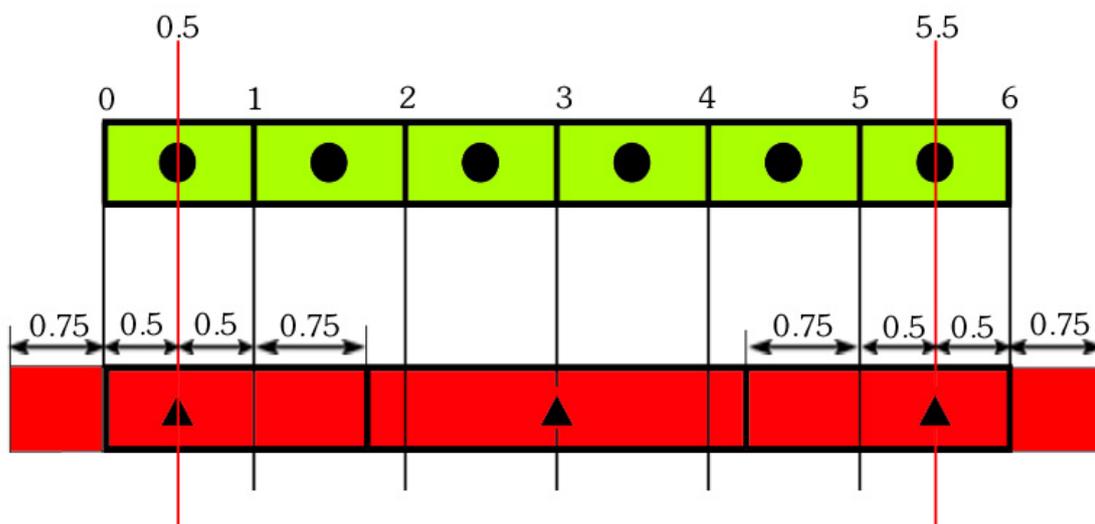


圖 4-5 一維度的 Local Average 影像縮小運算示意圖



圖 4-6 對於圖 4-4 (a)作 Local Average 影像縮小之結果。

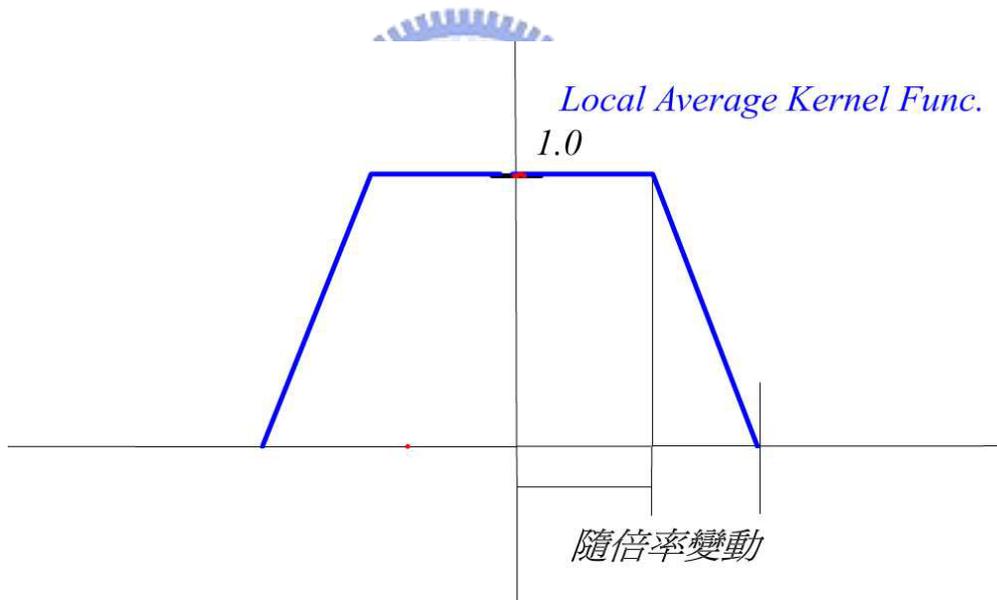


圖 4-7 空間域上的 Local Average 核心函數。

關於二維的 Local Average 處理流程，以圖 4-8 所示之圖例來作說明，其設計構想為：  
 $H$  為縮小處理後影像之高、 $W$  為縮小處理後影像之寬、 $Y_{Hi}$ 、 $Y_{Lo}$ 、 $X_{Hi}$ 、 $X_{Lo}$  分別為參考 window 於原始影像之邊界座標。 $bufX[]$ 及  $bufY[]$ 為暫存運算權值記憶體。對於每個縮小影像上的點，都有對應於原始二維影像的平均範圍，由  $X_{Lo} \sim X_{Hi}$ 、 $posY_{Lo} \sim Y_{Hi}$  所包含，透過  $X_{Lo} \sim X_{Hi}$ 、 $Y_{Lo} \sim Y_{Hi}$  此一已知範圍，我們便可依序計算每個平均範圍（即

二維原始影像小區域) 內，個別兩個維度的平均係數。圖 4-8 例中 X 方向係數即 BufXweight[] 為 0.4、1、1、0.4，而 Y 方向係數即 BufYweight[] 為 0.4、1、0.23，我們只要儲存這兩個維度的平均係數，如此便可以將 XLo~XHi、YLo~YHi 範圍內的每一個點，分別乘上兩個方向的係數，進而算出區域內的總和與平均，也就是說我們可將區域內的平均係數與區域內的原始影像亮度作 Weighted Convolution，如此即可計算出每個縮小影像取樣點的亮度，整個計算流程如圖 4-9 流程區塊所示；此一縮小設計的特點在於，透過二維小區域 Region Buffer 的使用，儲存原始影像的區域顏色值，加上兩個一維的平均係數計算，我們便可以快速的求出縮小影像。

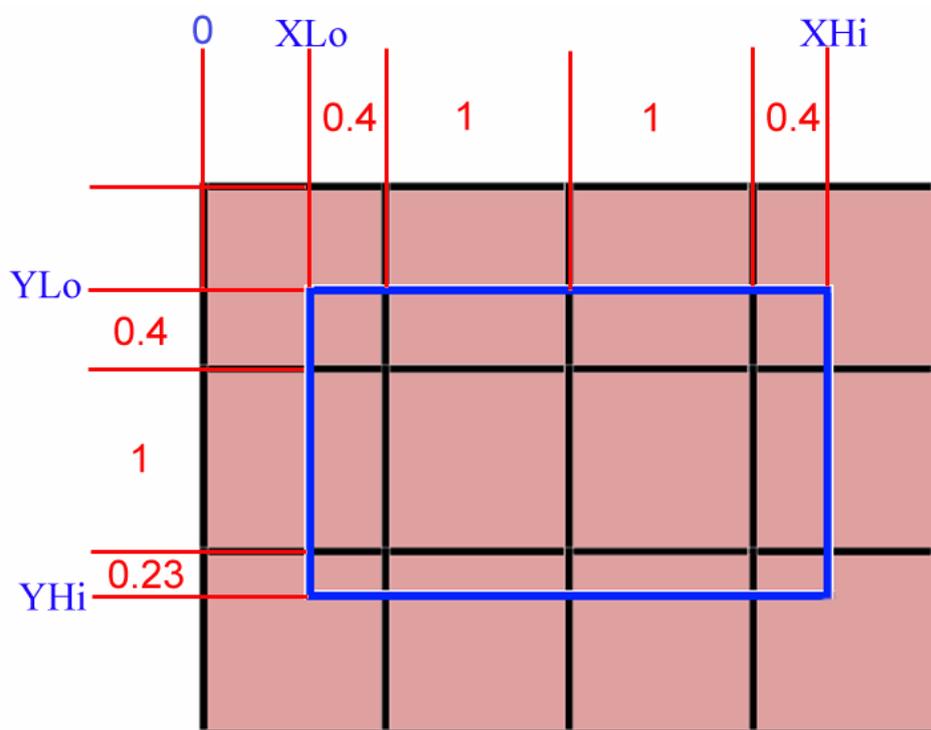


圖 4-8 LocalAverage 縮小示意圖，其中紅色阿拉伯數字代表影像像素縮小後所佔遠使影像像素之比例。

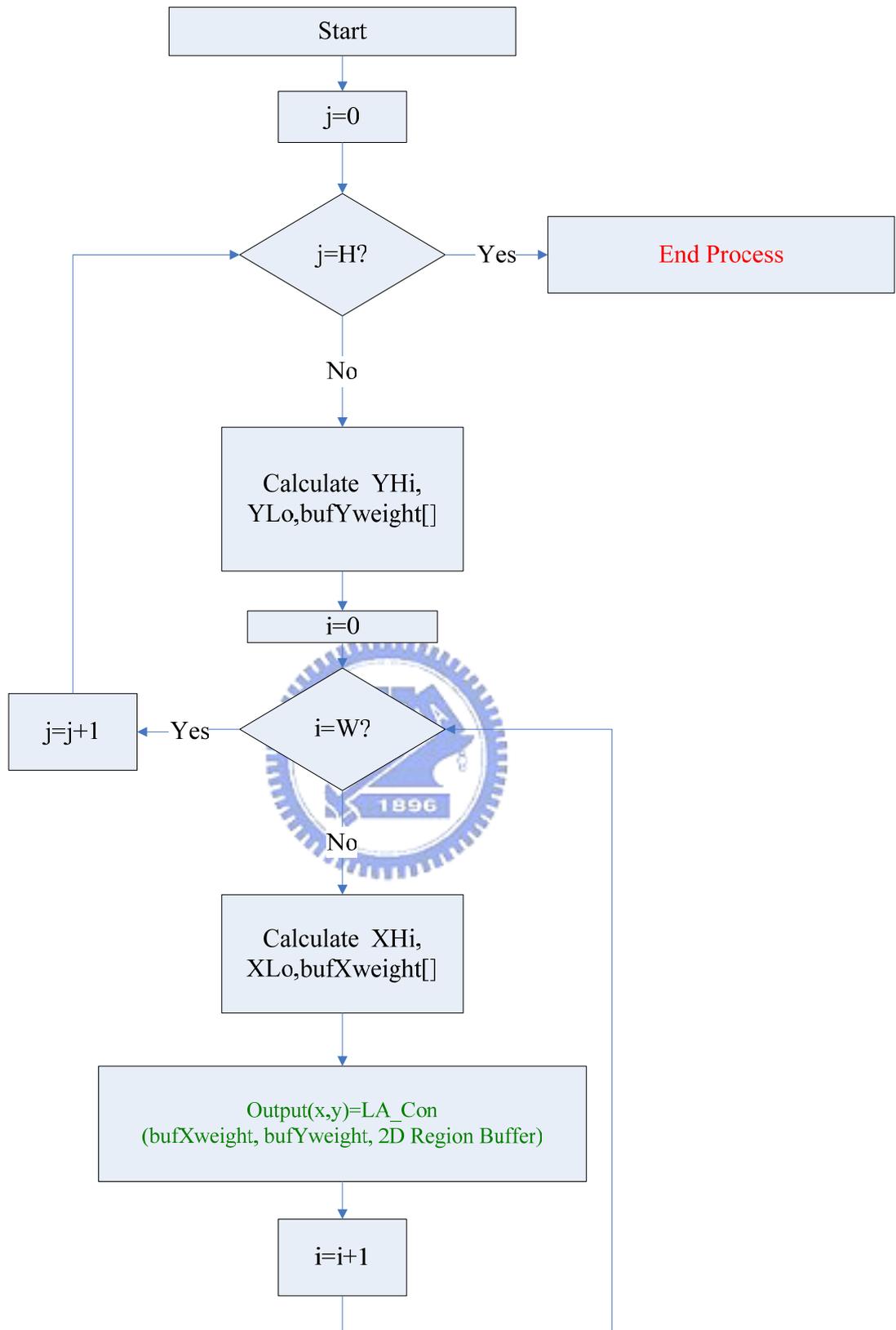


圖 4-9 二維度Local Average演算法處理流程圖。

## 4.4 實驗結果

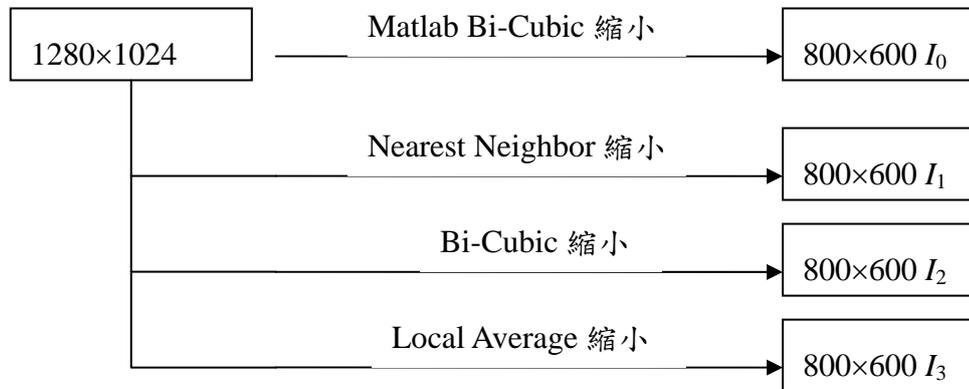


圖 4-10 縮小實驗測試比較結果流程圖。

如圖 4-10 所示，在影像縮小實驗數據比較中，我們測試了四種不同的縮小演算法，其中我們以第一個作法 Matlab Bi-Cubic（其 Cubic Kernel Function 的寬度隨著縮小比例而改變），當作影像縮小的標準，此乃是因為 Matlab Bi-Cubic 的影像縮小方法，可以透過原始程式碼來驗證影像對齊（Image Registration）是否正確，同時又是一個廣為大眾所使用的軟體方法，因此我們將其影像縮小結果當作標準。其他三個影像縮小的方法則是和此一作法相比較，包含了 Nearest Neighbor、固定寬度的 Bi-Cubic 以及 Local Average 三種。透過影像相減的差值分析，計算三種比較方法與 Matlab 標準作法之間的影像最大差值、平均差值與差值變化量，我們可以定量的得出哪一種影像縮小的作法，與一般標準作法最為接近。

實驗結果如圖 4-11 至圖 4-14 以及表 4-1 至表 4-4 所示，我們可以發現，無論是對於自然影像或是對於文字模式的測試影像，Local Average 的縮小方法，皆有較小的誤差，這樣的實驗數據結果，說明了 Local Average 此一簡單的影像縮小方法，可以得出與一般較複雜的影像縮小方法相類似的結果，亦為一種符合人類視覺觀感的設計選擇。



圖 4-11 自然風景影像：原圖由 1280×1024 縮小至 800×600



表 4-1 自然風景影像之縮小誤差統計表

$ \Delta $	NN	Bi-Cubic	Local Average
Max	90.666667	57.666667	25.666667
Ave.	1.334505	0.692206	0.449797
Std.	2.927855	1.751077	1.383008

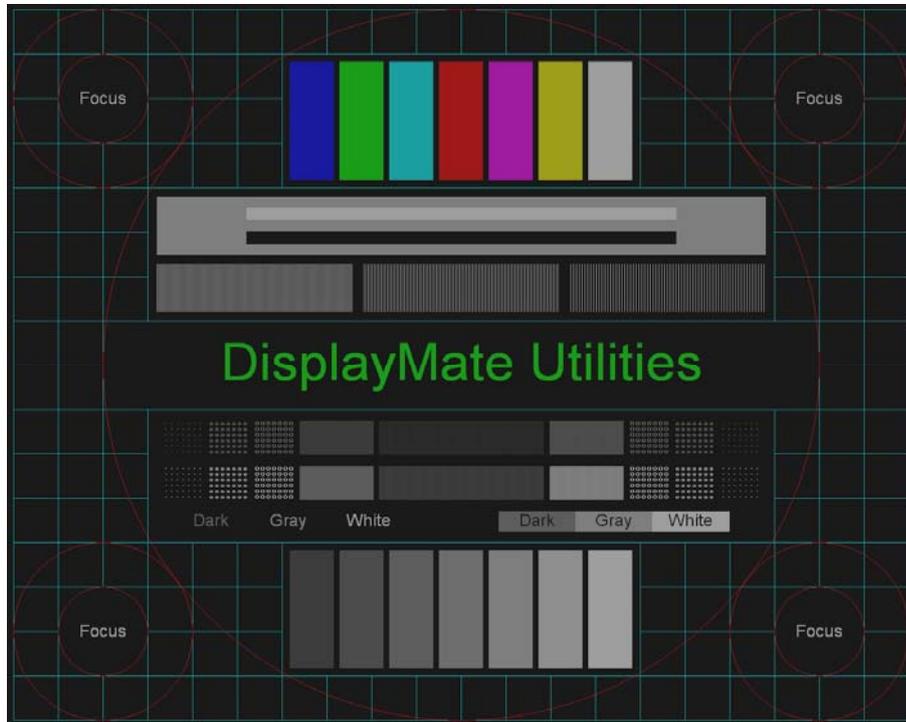


圖 4-12 主要測試影像：原圖由 1280×1024 縮小到 800×600

表 4-2 主要測試影像之縮小誤差統計表

$ \Delta $	NN	Bi-Cubic	Local Average
Max	205.000000	188.000000	64.333333
Ave.	6.425226	3.597701	1.461898
Std.	4.686032	4.203241	3.426446

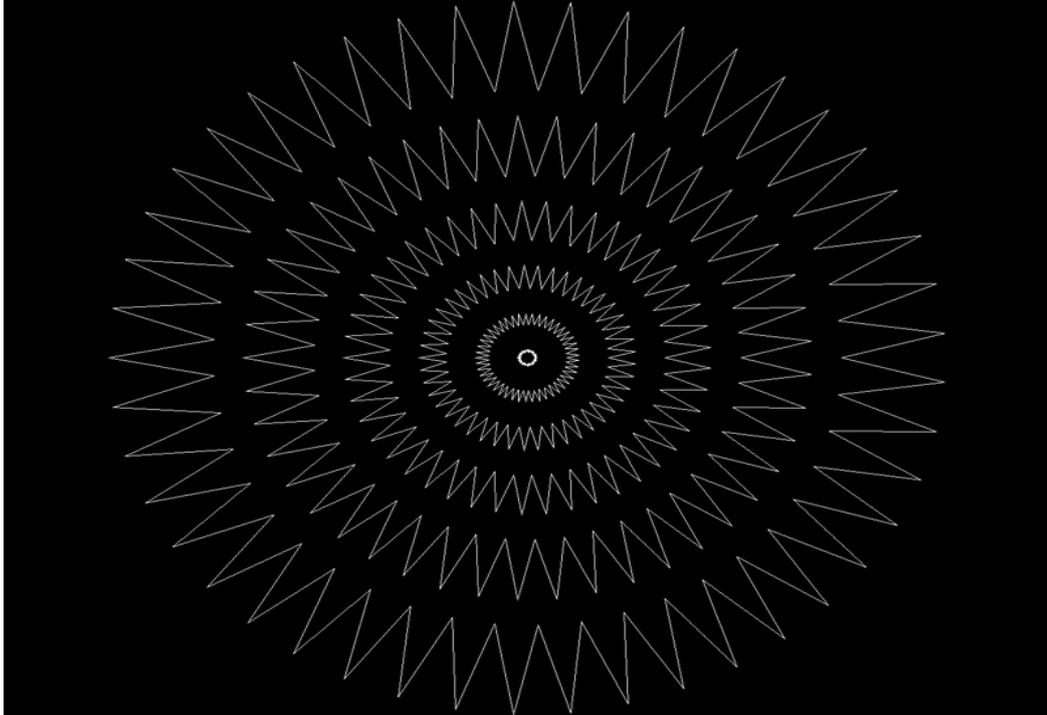


圖 4-13 環形條紋影像：原圖由 1280×1024 縮小至 800×600

表 4-3 環形條紋影像之縮小誤差統計表

$ \Delta $	NN	Bi-Cubic	Local Average
Max	203.000000	174.000000	59.000000
Ave.	3.063358	1.661148	0.813340
Std.	3.320005	3.047122	2.613338



圖 4-14 動物影像：原圖由 1280×1024 縮小至 640×480

表 4-4 動物影像之縮小誤差統計表

$ \Delta I $	NN	Bi-Cubic	Local Average
Max	156.333333	134.000000	26.000000
Ave.	2.014083	1.398590	0.446765
Std.	2.933172	2.336573	1.075022

## 五、 討論

本章節中，我們首先將我們所研究並採用的內插核心函數作一總結的討論，其次我們也提出了本研究論文之未來的研究方向及展望。

### 5.1 所採用方法的總結

在影像放大的部分，我們嘗試了許多種方法，如 Sinc、Nearest Neighbor、Bi-Linear、Bi-Cubic。經過複雜度、效能以及硬體上的可實現性之間的比較，最後我們選擇了 Bi-Cubic 當作我們影像放大部分的主要演算法，而且決定使用 Look-up Table 的方式實作出 Bi-Cubic 演算法，在我們的實作中，只要增加或改變 Look-up Table 的內容，便可以直接進行不同解析度的影像放大的運算，而無須再更動相關的設計方式。

在影像縮小的部分，為了避免取樣不均造成雜訊出現的問題，故需先對影像做了去高頻的動作，再做縮小。因此，最後我們選擇了 Local Average 做為我們主要的演算法。

### 5.2 未來展望

影像放大中，除了基本的 Bi-Cubic 演算法架構之外，我們還實作了為了硬體實現而提出的 Look-up Table 的作法，並討論定點(Fixed Point)運算中，補償數值計算誤差的相關問題。而在於第二章，我們使用了  $n \times n$  的 Window Size 來當做評估整張影像中的個別區塊適用於何種內插核心函數，我們得到的結果是：在整張影像中，並非所有的區塊影像皆適用於同一種內插核心函數。在這個部分我們希望未來可以更深入研究探討，是否可以根據輸入影像之影像特性，作最佳化的複合式內插演算法。

此外，以多個核心函數組成的內插演算法之硬體化實現亦是一個相關的研究發展方向，亦期望以發展一調適性的內插硬體架構，以改良既有的 Bi-Cubic 內插效果與發展不

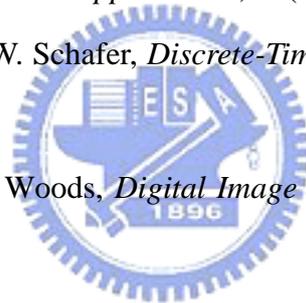
同的硬體實現方式。



## 參考文獻

- [1] A.J. Eglit, "Method and Apparatus for Upscaling an Image in Both Horizontal and Vertical Directions," US005739867, Feb. 1997.
- [2] Harasimiuk and Michal, "Image Scaling," US006825857b2, Jan. 2002.
- [3] Parks, T.W., and C.S. Burrus, "*Digital Filter Design. New York*" John Wiley & Sons, 1987. Pages. 209-213.
- [4] R.G. Keys, "Cubic Convolution Interpolation for Digital Image Processing", *IEEE Trans. Acoustic, Speech and Signal Processing* , vol. 29, pp. 1153-1160, 1981.
- [5] K. Turkowski, "Filters for Common Resampling Tasks," *Graphics Gems I*, Academic Press, pp. 147-165, 1991.
- [6] D.P. Mitchell and A.N. Netravali, "Reconstruction Filters in Computer Graphics," *Proc. SIGGRAPH*, volume 22, pages 221--228, August 1988.
- [7] <http://www.cg.tuwien.ac.at/~theussl/DA/node11.html>
- [8] C.H. Kim, S.M. Seong, J.A. Lee, and L.S. Kim, "Winscale: An Image-Scaling Algorithm Using an Area Pixel Model," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 13, No. 6, pp. 549-553, June 2003.
- [9] AVS Forum, "Lanczos vs Bicubic Comparison," <http://www.avsforum.com/avs-vb/showthread.php?t=460922&page=1&pp=30>
- [10] X. Wu and X. Zhang, "Image Interpolation Using Texture Orientation Map and Kernel Discriminant," *International Conference on Image Processing*, 2005.
- [11] A.J. Storkey, "Dynamic Structure Super-Resolution," *Proc. NIPS*, 2002.
- [12] M. Bertalmio, A.L. Bertozzi, and G. Sapiro, "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting," *Proc. CVPR*, 2001.
- [13] B.S. Morse and D. Schwartzwald, "Image Magnification Using Level-Set Reconstruction," *Proc. CVPR*, 2001.

- [14] M. Elad and A. Feuer, "Restoration of a Single Superresolution Image from Several Blurred, Noisy, and Undersampled Measured Images," *IEEE Trans. Image Processing*, vol. 6, no. 12, pp. 1646–1658, December 1997.
- [15] M. Irani and S. Peleg, "Improving Resolution by Image Registration," *CVGIP*, 1991.
- [16] S. Baker and T. Kanade, "Limits on Super-Resolution and How to Break Them," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, September 2002.
- [17] H. Chang, D.Y. Yeung, and Y. Xiong, "Super-Resolution through Neighbor Embedding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. I: 275–282, 2004.
- [18] W.T. Freeman, T.R. Jones, and E.C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications*, 22(2):56–65, March/April 2002.
- [19] A.V. Oppenheim and R.W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, 2<sup>nd</sup> Edition, 1999.
- [20] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, 2<sup>nd</sup> Edition, 2002.



## 附錄一、Cubic convolution 核心方程式之係數推導

由已定義之內插函數：

$$g(x_k) = \sum_k c_k u\left(\frac{x-x_k}{h}\right) = \sum_k c_k u(s_k) \quad (\text{A.1})$$

為出發點。其中， $g(x)$  為  $f(x_k)$  影像內插所得的方程式， $u$  為內插核心函數， $x_k$  為內插取樣點， $c_k$  為在  $x_k$  處的取樣資料影像強度，即  $f(x_k)$ ， $h$  為取樣密度（取樣區間），由於上述  $g(x_k) = f(x_k)$  的限制，故  $u(0)=1$  且  $u(1)=0$ 。在設計 Cubic Convolution 內插方程式  $g(x)$  時，其定義的取樣子區間為  $(-2,-1)$ ， $(-1,0)$ ， $(0,1)$ ， $(1,2)$ ，且內插方程式是由三次方多項式組成，用來近似 Sinc 核心函數，而在區間  $(-2,2)$  之外內插方程式定義為 0。於是可以得到下面的內插方程式：

$$u(s) = \begin{cases} A_1 |s|^3 - B_1 |s|^2 + C_1 |s| + D_1, & 0 < |s| < 1 \\ A_2 |s|^3 - B_2 |s|^2 + C_2 |s| + D_2, & 1 < |s| < 2 \\ 0, & 2 < |s| \end{cases} \quad (\text{A.2})$$

如前所述，若放大影像的內插點位置對應到原始影像的像素點位置，則它們的亮度值必須一致，因此內插方程式必須符合  $u(0)=1$  和  $u(n)=0$ ，其中  $n$  為任何非 0 的整數， $h$  為每次取樣的增加量，而內插取樣點  $x_j$  與  $x_k$  的距離為  $(j-k)/h$ ，再將  $x_j$  代回式 A.1，得到：

$$g(x_j) = \sum_k c_k u(j-k) \quad (\text{A.3})$$

若  $j$  不等於  $k$ ，則  $u(j-k)=0$ 。式 A.3 中的方程式係數在  $u(0)=1$  與  $u(1)=u(2)=0$  的條件下，可以得到以下四個等式：

$$\begin{aligned} 1 &= u(0) = D_1 \\ 0 &= u(1^-) = A_1 + B_1 + C_1 + D_1 \\ 0 &= u(1^+) = A_2 + B_2 + C_2 + D_2 \\ 0 &= u(2^-) = 8A_2 + 4B_2 + 2C_2 + D_2 \end{aligned} \quad (\text{A.4})$$

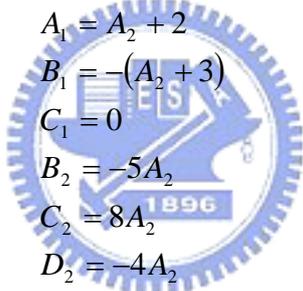
再利用方程式的連續性，將函數做一次微分，可以得以下等式：

$$u'(s) = \begin{cases} 3A_1s^2 + 2B_1s + C_1 \\ 3A_2s^2 + 2B_2s + C_2 \\ 0 \end{cases} \quad (\text{A.5})$$

再將  $x=0$ 、 $x=1$ 、 $x=2$  三個點分別代入上式即可得到：

$$\begin{aligned} -C_1 &= u'(0^-) = u'(0^+) = C_1 \\ 3A_1 + 2B_1 + C_1 &= u'(1^-) = u'(1^+) = 3A_2 + 2B_2 + C_2 \\ 12A_2 + 4B_2 + C_2 &= u'(2^-) = u'(2^+) = 0 \end{aligned} \quad (\text{A.6})$$

在這七個方程式中有八個未知係數，因此我們可以求出係數間的關係，並且以  $A_2$  表示如下：



$$\begin{aligned} A_1 &= A_2 + 2 \\ B_1 &= -(A_2 + 3) \\ C_1 &= 0 \\ B_2 &= -5A_2 \\ C_2 &= 8A_2 \\ D_2 &= -4A_2 \end{aligned} \quad (\text{A.7})$$

再令  $A_2 = a$  代回式 A.3，得到：

$$u(s) = \begin{cases} (a+2)|s|^3 - (a+3)|s|^2 + 1, & 0 < |s| < 1 \\ a|s|^3 - 5a|s|^2 + 8a|s| - 4a, & 1 < |s| < 2 \\ 0, & 2 < |s| \end{cases} \quad (\text{A.8})$$

假設  $x$  為任意需要被內插補點的位置，且  $x$  在兩個連續內插取樣點  $x_j$  與  $x_{j+1}$  之間，則令

$$s = \frac{(x - x_j)}{h},$$

因此

$$\frac{(x - x_k)}{h} = \frac{(x - x_j + x_j - x_k)}{h} = \frac{(x - x_j)}{h} + \frac{(x_j)}{h} - \frac{(x_k)}{h} = s + j - k,$$

則式 A.1 可改寫為：

$$g(x) = \sum_k c_k u(s+j+k) \quad (\text{A.9})$$

然而  $u$  在  $(-2,2)$  區間外，其值為 0，所以式 A.9 在  $0 < s < 1$  的條件下，可以改寫為：

$$g(x) = c_{j-1}u(s+1) + c_j u(s) + c_{j+1}u(s-1) + c_{j+1}u(s-2) \quad (\text{A.10})$$

在將上面的結果代入式 A.8 可得：

$$\begin{aligned} u(s+1) &= a(s+1)^3 - sa(s+1) + 8a(s+1) - 4a = as^3 - 2as^2 + as \\ u(s) &= (a+2)s^3 - (a+3)s^2 + 1 \\ u(s-1) &= -(a+2)(s-1)^3 - (a+3)(s-1)^2 + 1 = -(a+2)s^3 + (2a+3)s^2 - as \\ u(s-2) &= -a(s-2)^3 - 5a(s-2)^2 - 8a(s-2) - 4a = -as^3 + as^2 \end{aligned} \quad (\text{A.11})$$

將  $u(s+1), u(s), u(s-1), u(s-2)$  代回式 A.10，依照  $s$  的幕次整理過後，可以得到：

$$\begin{aligned} g(x) &= -[a(c_{j+2} - c_{j-1}) + (a+2)(c_{j+1} - c_j)]s^3 \\ &\quad + [2a(c_{j+1} - c_{j-1}) + 3(c_{j+1} - c_j) + a(c_{j+2} - c_j)]s^2 - a(c_{j+1} - c_{j-1})s + c \end{aligned} \quad (\text{A.12})$$

若  $f$  在  $x_j, x_{j+1}$  區間至少有三個微分値（為一個連續的三次方程式），根據泰勒定理（Taylor's Theorem）可得出： $(O(h^3))$  在此處略過不加以討論）

$$c_{j+1} = f(x_{j+1}) = f(x_j) + f'(x_j)h + \frac{f''(x_j)h^2}{2} + O(h^3) \quad (\text{A.13})$$

$$c_{j+2} = f(x_{j+2}) = f(x_j) + 2f'(x_j)h + 2f''(x_j)h^2 + O(h^3) \quad (\text{A.14})$$

$$c_{j-1} = f(x_{j-1}) = f(x_j) - f'(x_j)h + \frac{f''(x_j)h^2}{2} + O(h^3) \quad (\text{A.15})$$

由以上三式代回式 A.12 可得：

$$\begin{aligned} g(x) &= -(2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 \\ &\quad + [(6a+3)hf'(x_j) + (4a+3)h^2 f''(x_j)/2]s^2 \\ &\quad - 2ahf'(x_j)s + f(x_j) + O(h^3) \end{aligned} \quad (\text{A.16})$$

然而， $sh = x - x_j$  所以  $f(x)$  的泰勒展開式為：

$$f(x) = f(x_j) + shf'(x_j) + s^2 h^2 f''(x_j)/2 + O(h^3) \quad (\text{A.17})$$

將式 A.17 減式 A.16 可得：

$$\begin{aligned}
f(x) - g(x) &= (2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 \\
&\quad - (2a+1)[3hf'(x_j) + h^2 f''(x_j)]s^2 \\
&\quad + (2a+1)hf'(x_j)s + O(h^3)
\end{aligned} \tag{A.18}$$

若令內插方程式  $g(x)$  必須與  $f(x)$  的泰勒展開式前三項 ( $s$  乘冪項) 必須一致 ( $O(h^3)$  為重新取樣頻率方程式，對主要方程式影響不大，在此處略過不加以討論)，我們可以

得到： $a = -\frac{1}{2}$ 。

再將  $A_2 = a = -\frac{1}{2}$  代回式 A.8，則得到 Cubic Convolution 內插放大方程式：

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1, & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 - \frac{5}{2}|s|^2 + 4|s| + 2, & 1 < |s| < 2 \\ 0, & 2 < |s| \end{cases} \tag{A.19}$$

由以上函數核心的推導可知：立方內插 (Cubic Convolution Interpolation) 演算法以原始輸入影像座標系中的四個已知像素點的影像強度值 (Intensity)，乘以其權重值來計算出欲補的點之影像強度值。