電機資訊學院 電子與光電學程

碩 士 論 文

整合式解交錯演算法之設計

An Integrated De-interlacing Algorithm Design

研究生：曾坤源

指導教授：杭學鳴 博士

中 華 民 國 九 十 三 年 八 月

整合式解交錯演算法之設計

An Integrated De-interlacing Algorithm Design

研 究 生：曾坤源　　Student：Kun-Yuan Tseng

指導教授：杭學鳴　　Advisor：Hsueh-Ming Hang

國 立 交 通 大 學

電機資訊學院 電子與光電學程

碩 士 論 文

A Thesis
Submitted to Degree Program of Electrical Engineering
Computer Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Electronics and Electro-Optical Engineering
August 2004
Hsinchu, Taiwan, Republic of China

中華民國九十三年八月

整合式解交錯演算法之設計

曾坤源　杭學鳴

國立交通大學　電機資訊學院

摘　要

在最近幾年，數位電視已經開始播放和循序掃描的輸出顯示裝置已普及一般家庭，因此將交錯掃描轉換成循序掃描的解交錯(De-interlacing)演算法就變得相當重要，雖然移動可適性解交錯(Motion Adaptive De-interlacing)的演算法已經廣泛應用在現今許多的商品上，但在某些移動區域的畫質仍有需要再提昇。

本篇論文提出的整合式解交錯(Integrated De-interlacing)的演算法，可以有效提昇移動區域的畫面，但是當移動估計不正確時，反而會使移動補償後的畫面變得很差，為了改善這種情況，因此結合移動可適性解交錯的優點，並將空間圖場內插(Spatial Interpolation)的方式改成 ELA(Edge Line Average)來設計，經過電腦模擬的結果發現，不僅在視覺上提高畫面的解析度，在某些影像峰值訊號雜訊比(Peak Signal Noise Ratio , PSNR)也比線平均解交 錯(Line Average De-interlacing)多出好幾分貝的畫質增益。

此外，在整合式解交錯演算法中也增加影片偵測(Film Detection)和影像加強(Image Enhancement)的演算法設計，在這樣演算法的架構下，透過影片偵測的演算法，我們可真實地還原 3:2 Pull Down 的影片格式，而不會有鋸齒狀(Saw-Toothed)的畫面出現，而影像加強的演算法，則可以在解交錯後，經過影像的調整，使輸出畫面呈現不同的效果，達到消費者的需求。

An Integrated De-interlacing Algorithm Design

Student: Kun Yuan Tseng Advisor: Prof. Hsueh Ming Hang

Institute of Electronics

National Chiao Tung University

Abstract

In recent years, progressive scanning output devices have becoming very popular. However, many video sources are still using the conventional interlaced TV format. Therefore, the de-interlacing technique which converts interlaced picture to progressive picture is essential. It has been used widely in various kinds of commercial products. But the conventional simple de-interlacing method does not provide good picture quality particularly in the moving areas.

The main theme of this thesis is an integrated de-interlacing system, which incorporates several known and improved techniques in a nice manner to produce good de-interlaced image quality. We first develop an accurate motion detector that classifies image regions into stationary, low-motion, and high-motion categories. The simple field merging method is applied to the stationary regions. The edge line average interpolation method is applied to the slow-motion regions. Finally, the motion-compensated interpolation is applied to the high-motion regions. In addition, hierarchical motion estimation and motion vector smoothing techniques are employed to enhance the quality of estimated motion vectors. Our computer simulation shows that the subjective image quality is improved by using the proposed scheme. Also, its PSNR measures are better than the conventional spatial or temporal interpolation schemes.

In addition, in real applications, interlaced TV pictures are often derived from the films at different frame rate. To achieve a good picture quality, a film detector and an associated image enhancement scheme are added to our de-interlacing system. In this proposed system architecture, we can recover the 3:2 pull down format. With image enhancement algorithm after de-interlacing, the subjective image quality is generally improved.

誌　　謝

走過木棉，走過杜鵑，陽光燦爛裡，蟬嘶啼亮，鳳凰花紅，又一季驪歌輕揚。

三年的時間，不長也不短，若自比為一株小樹苗，經過漫長的培育、灌溉、施肥。如今，成長茁壯，最最感謝的，是那位不辭辛勞的園丁──論文指導教授──杭學鳴老師。他，嚴謹的治學態度，讓我看得更遠、更廣；積極的研究精神，令我嘆為觀止；親切的為人處事，更值得我學習、效法。

論文研究的這段期間，非常感謝杭學鳴教授給我很大的幫忙，不僅像大海中的燈塔，正確地導引我的研究方向，而且在我遇到困難或瓶頸時，都能適時給我很好的建議，並不斷的給我鼓勵，才能完成論文的研究。

另外，我要感謝我的家人，謝謝他們在背後給我的支持，尤其是我的妻子，因為家庭和小孩在她妥善的照顧下，讓我無後顧之憂，能全心全意投入論文研究，才能順利完成論文。

目　　錄

表 目 錄

圖 目 錄

第一章　緒論

早期的電視系統，因為頻寬及生產技術上的限制，電視機是以交錯式（Interlaced）的掃描方式來產生畫面，讓消費者可以獲得較為流暢的影像。但隨著科技的進步，漸進式（Progressive）的顯示裝置快速的發展，如電腦螢幕、電漿電視、液晶電視等的輸出配備，已是漸進式的掃描方式。由此看來，交錯式的掃描方式，已漸漸無法符合今日漸進式的輸出顯示裝置。

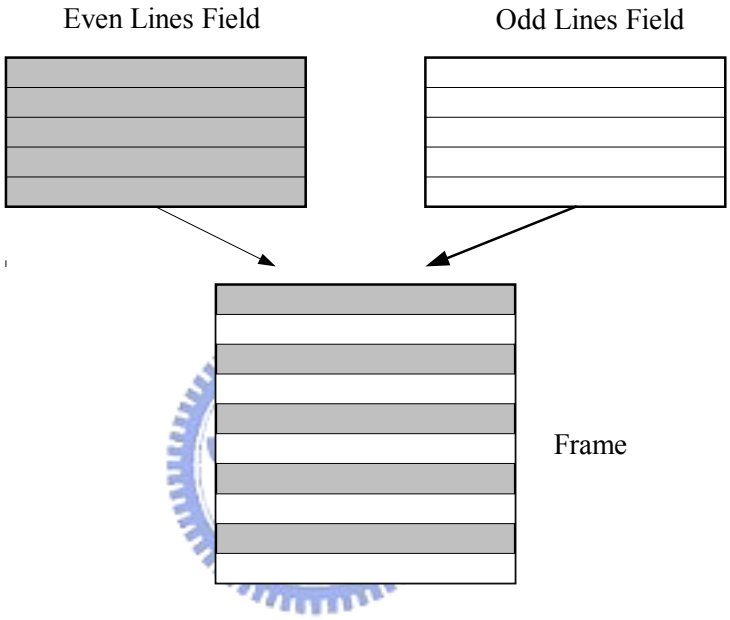


圖 1.1　圖場和圖框

交錯式掃描是在傳統電視 SDTV（Standard Definition Television）就制定的一種掃描方式。在連續交錯式的視訊廣播訊號中，影像畫面是以“圖場”（Field）為單位，在螢幕上互相顯示，每張圖場開始輸出時，電子槍會開始掃描。掃描到水平方向的最後一個像素，會跳到隔二行掃描的那些空的掃描線來開始掃描。同樣地，在水平方向的最後一點時，會跳到隔二行再繼續掃描，如此不斷的掃描動作即稱為交錯式掃描（Interlaced）。在這樣的掃描模式裡面，每張圖場的垂直解析度是整個畫面的一半，和傳統 CRT 螢幕不一樣的是，現在的液晶螢幕是從最左上角的像素，開始掃描到最右下角，這稱為漸進式的掃描。每張畫面的垂直解析度，就是整個畫面的垂直解析度，稱為圖框（Frame），圖 1.1 說明圖場和圖框的資料掃描格式。

以前使用交錯式掃描主要的原因有兩個：第一個原因是以人類能接受的流暢度而言，也就是螢幕的「更新頻率」速度，必須就像日光燈一樣，每一秒必須閃動 60 次以上，對人的眼睛來說才不會有閃爍的感覺。但由於在過去技術並不成熟，對訊號頻寬有

很大的限制，所以在有限的頻寬下，人們制定出每秒 60 張圖場的規格。因此，相較於每秒 60 張圖框，我們不但只需要一半的頻寬就足夠，而且又能給人們比較流暢的畫面享受。第二個原因是在於電子槍打到螢幕上，而螢幕上的磷粉發光時，磷粉的光芒並不會很快的消失掉，這是以前的電視特性。基於這個特性，再加上人類的視覺殘留現象，可以讓人們覺得畫面是連續的。因此，即使我們用每秒 60 張圖場，由於視覺暫留的關係，我們也看不到有缺陷的畫面。

當我們要將交錯式掃描的畫面輸出到漸進式（Progressive）掃描的顯示裝置時，漸進式掃描不能直接將兩張相鄰圖場的像素合併在一起輸出，因為這兩張畫面是在不同時間點拍攝的，所以一起輸出時，有些物件的位置改變了，會造成畫面有「交叉邊緣」（line crawling）的畫質。為了解決這個問題，使得交錯式的掃描畫面，可以顯示在漸進式掃描的顯示器上，解交錯這項技術便開始受到重視，也越來越多的人投入這方面的研究。

解交錯是要將 60 圖場/每秒，轉換成 60 圖框/每秒，這 60 圖場/每秒的資料，是將一秒鐘切成六十等份，每一份取一次資料；而 30 圖框/每秒則只切成 30 份，圖 1.2 即是解交錯示意圖[1]。在時間軸來看，交錯式的取樣遠比漸進式較多，交錯式所取得的資料卻只有一半的畫面，而漸近式則是完整的畫面。在一般要將 60 圖場/每秒轉成圖框/每秒的方法中，最直接的方法便是將兩個圖場交錯合併成一個圖框。但是這樣子會造成交叉邊緣的現象，因為相併的兩張畫面在時軸上，並不是屬於同一個時間點的位。如此，將使得整個畫面上的物體，會有一半是在前一個時間點，另一半在這個時間點，畫面就會變模糊了。

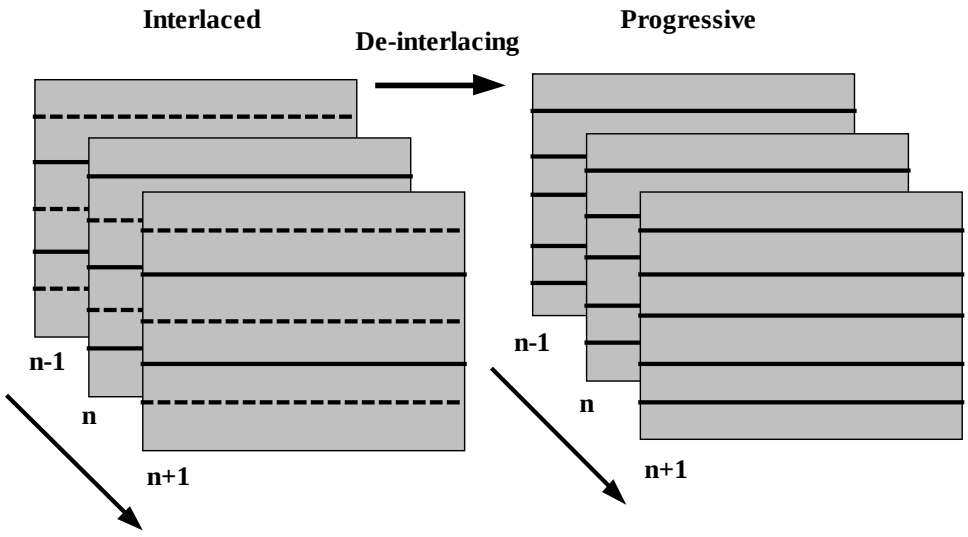


圖 1.2 解交錯示意圖

解交錯的演算法，從最早的掃描線重覆（Line Repetition） 解交錯，到目前應用在現今大多數商業使用移動可適性解交錯（Motion Adaptive De-interlacing）的方式，已經漸漸無法滿足消費者的需求，於是移動補償解交錯（Motion Compensated De-interlacing）的方式，將會是未來發展的趨勢。它可以提高畫面的解析度，並運用移動估計（Motion Estimation）尋找移動的軌跡，對插點作適當的移動補償。但相對地，這種方式也有它的缺點，除了所需的硬體成本高之外，最大的困難是——如何偵測及修正錯誤的移動向量，這在本文中將是一個研究的重點。

在下面的章節中，我們將從理論的推導，進而作各種方式解交錯演算法模擬，整合它們的優點，而成為新的解交錯架構。在第二章我們將介紹非移動補償解交錯，和移動補償解交錯兩大類的理論推導。第三章我們整合新的解交錯的架構，並針對每一方塊除了做詳細的說明外，並針對不同的方法作模擬分析比較。第四章則增加影片模式偵測、影片還原及影像加強等設計，可以讓解交錯系統的模妳更完整。第五章是模擬結果的分析，比較不同解交錯演算法的差異，第六章是結論與未來的工作。

第二章　各類解交錯的原理概述

2.1 解交錯的起源

「數位電視」在世界各國掀起一片熱潮，一般來說，所謂的數位電視必需可以接收、顯示 18 種數位電視訊號。美國消費電子協會（CEA , Consumer Electronics Association）將數位電視訊號分成三類：高畫質數位電視（HDTV）、加強畫質數位電視（EDTV）與標準畫質數位電視（SDTV），各種的數位電視訊號如表 1 所列。隨著數位電視時代來臨，電視機的接收也都將變為數位接收，所以電視在未來不僅可以收看電視，還可以同時利用現有的電視頻道上網、進行資料傳輸。「數位電視」能提供收視戶超越傳統「類比電視」的畫質、品質、影音與個人化服務的電視廣播技術。此外，美國在一九九六年初通過電信修正法案、該年年底訂出高畫質數位電視標準，接著在一九九七年四月宣佈數位電視的實施時程，顯示出美國即將邁入西元兩千年數位化、寬頻化的魄力與決心。

垂直掃描線	水平像素	畫面比例	畫面掃描頻率
480	720	16:9 or 4:3	60p, 60i, 30p, 24p
720	1280	16:9	60i, 30p, 24p
1080	1920	16:9	60i, 30p, 24p

表 1：　數位電視訊號格式

數位電視有許多優點，它能提供高畫質的影像與聲音品質。現行使用的傳統類比電視只有 525 條掃描線，影像解析度較低、畫面易閃動、易產生「鬼影」；數位電視以「010101」的數位方式傳送，可避免外界干擾，不會有畫面模糊及「鬼影」現象，數位電視將 525 條掃描線提高至 1080 條，亦提供了清晰、鮮艷、穩定且生動的高品質電視畫面。由於液晶電視是今年當紅的產品，液晶電視的優點是——可對應電影、電視，甚至是高畫質數位電視的播放速率，使螢幕播放動畫影像時流暢無殘影，精彩節目不拖泥帶水。全方位支援高畫質數位電視的 720p、1080i 訊號規格，同時具有 16：9 顯示比例，可以輕鬆迎接即將全面開播的數位電視系統，或者接收國外高畫質衛星節目，迎頭趕上全球高畫質數位電視潮流。

數位電視和高畫質數位電視究竟是採用「循序式」抑是「交錯式」掃描，一直是爭論的話題。「循序式掃描」被視為個人電腦業者，突破家電業者長久把持數位電視規格制訂的利器，然而其最具代表性的業者微軟（Microsoft）和新力（Sony）結盟的目的，

卻是一起發展 1080 條交錯掃描式，和 480 條循序式掃描兩種規格之技術，打破了兩業界技術藩籬的嚴重歧見。此一結果証明這兩種掃描型式各有其優缺點，適用之場合迥異。「交錯式掃描」的最大優點是所需傳輸的資料量較小，動態影像的色彩鮮明度和對比也比較好；但缺點是無法顯示細膩的文字，不利於互動式資訊在電視上的應用，而此功能已被視為數位電視所必備的，足以為廣播業者帶來巨大的商業利益，突破類比廣播營收惡化之窘境。而循序式掃描則沒有這個問題，這是當年個人電腦業者群起攻之的關鍵性理由。然而循序式掃描也有致命的缺點，即其資料傳送量要比交錯式大，硬體成本偏高，且在影像的品質上也不如交錯式。在講究高解析度的場合下，特別是 1080 條掃描線的情況下，只有大畫面電視才用得著，畫質之對比和鮮明度尤其重要，所以交錯式比較適合，且其資訊傳輸量也能控制得較少。

從各方面所得到的資訊，我們可以得到一個結論——不管是現有的 NTSC 或 PAL 的電視系統，以及將來的數位電視和高畫質數位電視的電視系統，傳送交錯式掃描訊號格式仍是不會改變的趨勢。所以如何將交錯式掃描訊號顯示在液晶電視（LCD）或電漿電視，只有透過解交錯的處理，才能讓畫面的穩定度與精細感都會大幅提升，消費者在看電視時因畫面不容易閃爍，即使長時間觀看眼睛也不容易疲勞。

2.1.1 交錯掃描的格式分析

在傳統 CRT 電視中，它的影像掃描方式是利用映像管內電子束的射出，其方向是靠著一個「偏向裝置」，分為垂直偏向線圈和水平偏向線線圈，裝置利用磁場的改變，影響電子束上下左右射出方向。映像管中電子槍的方式可分為：交錯式掃描和非交錯掃描。交錯式掃描的方式是電子束第一次掃描時方式先掃描各奇數列，第二次掃描時在掃描偶數列，兩次掃描後完成影像的更新；非交錯的掃描的方式，是先掃描奇數列再掃描偶數列，而且畫面看起來比較不會閃爍，對眼睛比較不會造成傷害。

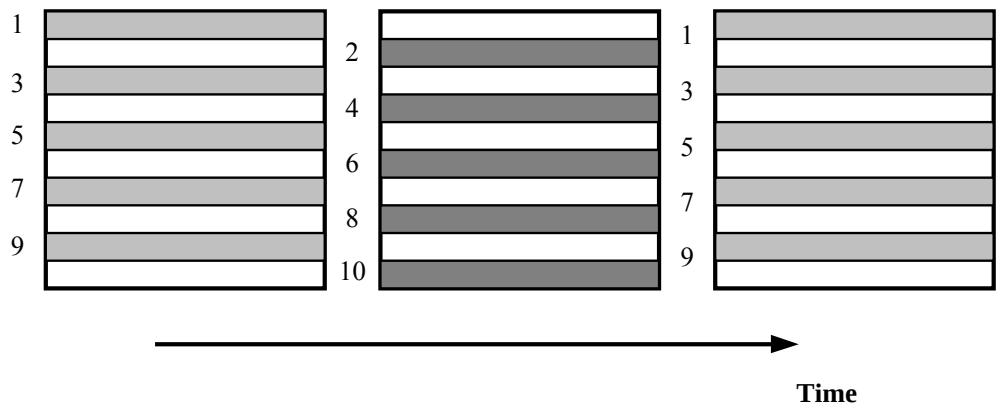


圖 2.1 交錯掃瞄格式輸出

交錯掃描可以顯示更平順的動態影像，但是在顯示靜態影像時容易閃爍。由於電腦螢幕不是使用交錯掃描顯示，所以在電腦螢幕上顯示電視畫面時，經常會產生梳子狀的效應，利用解交錯可以減少或消除這種效應。交錯模式與非交錯模式差別在交錯模式時，電子束的掃描方式並非由上到下逐一掃描的，而是先由一、三、五條的往下掃描，然後再二、四、六條往下掃描，圖 2.1 是交錯掃描輸出的資料格式。解析度越高時，因為頻寬的要求也愈高，此時再同時要達到高垂直更新率就比較難，因此透過交錯掃描的方式來達成，而與交錯掃描方式不同的逐一掃描方式，就稱為非交錯模式。

2.1.2 非交錯掃描的格式分析

所謂的「循序掃描 Progressive scan」就是以 1、2、3、4、5 至 525 線順序描繪出所有的掃描線圖，圖 2.2 是循序掃描的輸出方式。循序掃描方式，可消除交錯掃描形成的雜訊，提供穩定、不閃爍和不傷眼的畫面。

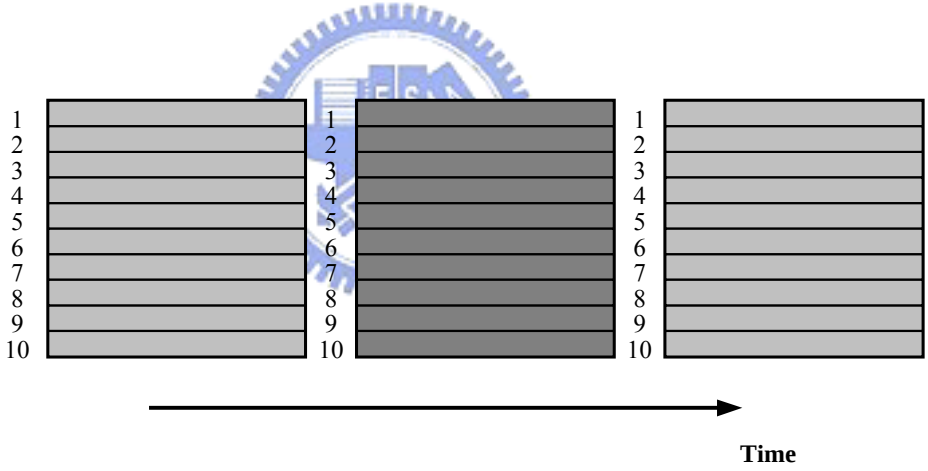


圖 2.2 循序掃瞄格式輸出

倍頻掃描是把水平掃描頻率提升為二倍，也就是把原來 15.7KHz 的水平掃描頻率提升到 31.5KHz，而水平掃描線的密度提升為二倍。這樣一來，畫面的細緻程度就會增加很多，以一個圖場掃 240 條線來說，倍頻就是 480 條線，這樣一來它與一個圖場一次〔60 分之一秒〕就掃 480 條線的循序掃描有什麼不同？單就掃描線的結果來看是一樣的，但是實際上倍頻的做法與循序掃描的過程有很大的不同。以電視的視訊來說，原本就是每秒 30 個圖場的掃描方式來形成圖像，然後再進行快速儲存記憶與播放的動作。而循序掃描卻是基礎上將一個圖場的掃描速度提升到二倍。若是從電影訊號來看，那就更複雜，因為電影訊號每秒 24 格，電視訊號每秒 30 格，所以這中間還要經過 3:2 Pull down 的程序，來把電影訊號轉換成能夠與電視訊號對應的每秒 30 格畫面。也因為這

樣，所以會產生畫面的人工失真（Artifact），在有些地方二個圖場所合起來的一個圖框會有失真的畫面。從電影轉電視訊號的角度來看，循序掃描不會有這種人工失真，而從交錯掃描倍頻之後的畫面，仍然會有這種人工失真，除非倍頻器的內部其實是以循序掃描方式在動作。

「倍頻掃描」這個名詞，基本上正式的說法應該稱之為循序漸進式掃描，在早期的電腦玩家則是曾稱之為「非交錯式掃描」，指的都是倍頻掃描的意思，事實上倍頻掃描就是將掃描頻率倍增，將交錯式掃描藉著畫面增加的方式來呈現畫面。

循序式掃描，每秒則傳送 60 張完整的畫面，因此 NTSC 訊號經過倍頻掃描的畫面會更為「緊密、不閃爍與穩定」。但絕對不是解析度倍增這麼神的功能，反而經過晶片倍頻處理後的畫面，經過測試還會發現解析度減少的情況，值得注意的是，並不是交錯式掃描就是有這樣的缺點，只要交錯式掃描的掃描線夠高，眼睛也是看到緊密的高畫質。

2.1.3 解交錯的定義

解交錯的目地，主要是輸入交錯掃描格式的圖場時，補足在同一個時間點，輸入沒有傳送進來的偶數線或奇數線資料。偶數圖場缺的是奇數線的資料，奇數圖場缺的是偶數線的資料，我們可從圖 2-3 解交錯示意圖中，輸入偶數圖場（Even Field）時，有資料的掃描線是 1、3、5、7、9，輸入奇數圖場時，有資料的掃描線是 2、4、6、8、10。在輸出影像時，都必須轉換為循序格式的圖框，掃描線是 1、2、3、4、5、6、7、8、9、10 皆有資料。由此我們可以定義這輸出圖框的數學式[2]為（式 2.1）：

$$f\text{out}(\vec{x},n) = \begin{cases} f_o(\vec{x},n), & (\text{y mod 2} = \text{n mod 2}) \\ f_i(\vec{x},n), & (\text{otherwise}) \end{cases} \qquad (式\ 2.1)$$

$\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ 表示為離散空間的位置，n 為圖場數目。

$f_o(\vec{x},n)$ 為原始圖場，當 y 除以 2 等於 n 除以 2 時。

$f_i(\vec{x},n)$ 為需插入的像素（Pixel），當 y 除以 2 不等於 n 除以 2 時。

目前，已有很多解交錯的演算法發表於過去的文獻中，及應用在現今商業產品上。

這些不同的演算法，在品質的表現上也有很大的差別，隨著科技的進步，需要大量運算的移動補償解交錯的演算法，已成為未來趨勢。

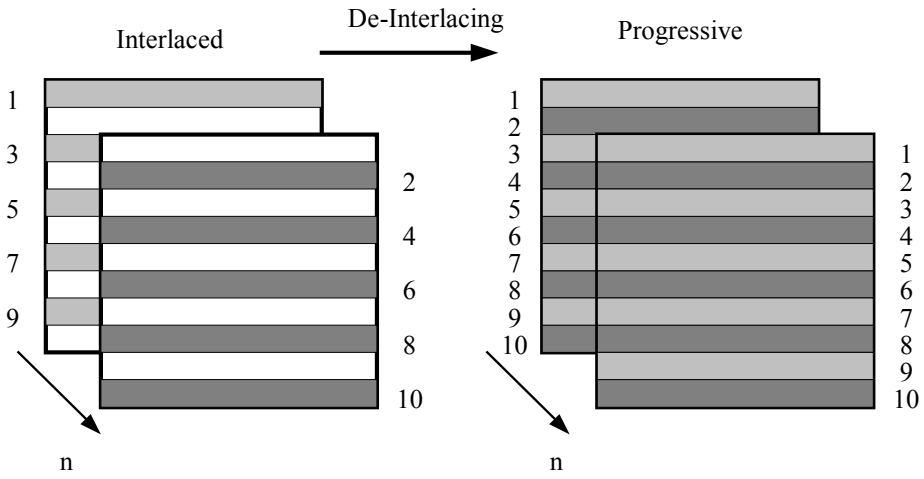


圖 2.3 解交錯轉換處理

2.2 非移動補償的解交錯

非移動補償的解交錯技術，能在靜態影像的部份，獲得較佳的品質。但動態的影像，就無法得到較佳的結果。因為非移動補償解交錯的缺點是忽略物件移動的路徑，而利用時間與空間上的技術來作圖場的插點，這些解交錯技術的運算比較簡單，所以硬體成本相對地就比較少，這是它最大的優點。我們可以將非移動補償解交錯的方式分成兩大類，分別為線性技術與非線性技術兩種解交錯的方式，在下面章節中，將會做詳細的討論。

2.2.1 空間解交錯

空間解交錯（Spatial De-interlacing）是利用同一圖場中，找垂直鄰近的像素來做中間插補的像素。這種作法適合物體移動較少的的畫面，所需的硬體成本少。一般來說比較常用的方法有兩種，掃描線複製（Line Repetition）是其中最簡單的方式，也是最早被運用在商業產品上，若以數學式來表示，其表示式如(式 2.2)：

$$f\text{out}(\vec{x}, n) = \begin{cases} f_o(\vec{x}, n), y \bmod 2 = n \bmod 2 \\ f_i(\vec{x} - \vec{u_y}, n), \text{otherwise} \end{cases} \qquad (式\ 2.2)$$

$\vec{u_y} = (0,1)^T$　　T 為轉換函數

另一種方式為掃描線垂直平均解交錯(Line Averaging De-interlacing)，比前一種方式更受歡迎，因為它可以減少受雜訊的干擾，圖 2.4 是掃描線垂直平均解交錯的示意圖。假設目前處理圖場為 n-1，則 C 像素的值等於 A 加上 B 再除 2，處理圖場為 n，則 F 像素的值等於 D 加上 E 再除 2，數學式表示如(式 2.3)：

$$f\text{out}(\vec{x}, n) = \begin{cases} f_o(\vec{x}, n), \quad y \bmod 2 = n \bmod 2 \\ [f_i(\vec{x} - \vec{u_y}, n) + f_i(\vec{x} + \vec{u_y}, n)] / 2, \text{otherwise} \end{cases} \qquad (式\ 2.3)$$

$\vec{u_y} = (0,1)^T$　　T 為轉換函數

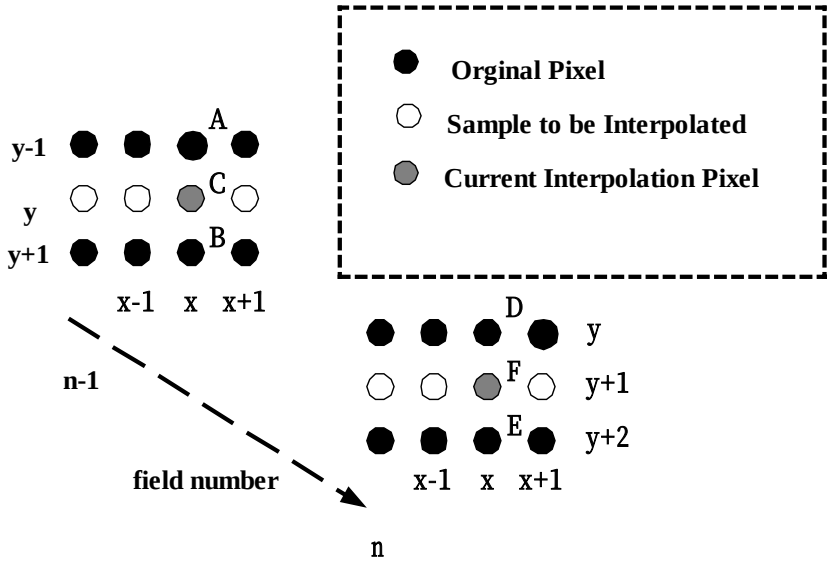


圖 2.4 空間垂直平均解交錯

2.2.2 時間解交錯

時間解交錯的方式是利用不同時間領域的圖場像素，經由一些的數學運算，來作插點的像素，這種方式的演算法最適合使用在完全靜態的影像，它是一種很簡單的解交錯方式，稱為圖場複製解交錯（Field Repetition De-interlacing），數學式為(式 2.4)：

$$f\text{out}(\vec{x}, n) = \begin{cases} f_o(\vec{x}, n), y \bmod 2 = n \bmod 2 \\ f_i(\vec{x}, n) = f(\vec{x}, n-1), \text{otherwise} \end{cases} \quad \text{（式 2.4）}$$

因為圖場複製解交錯的方法，在硬體設計上需要有 2 張圖場容量的記憶體，但它容易受雜訊的干擾，所以就有另一種時間平均解交錯（Temporal Averaging De-interlacing）演算法被提出。這種解交錯的硬體成本就比較高一些，它需要 3 張圖場的記憶體，但它可以避免一些外界雜訊的干擾，我們可參考圖 2.5 的示意圖，假如目前處理的圖場為 n，其中 C 點像素的值，是由 n-1 圖場中的 A 點和 n+1 圖場中的 B 點相加之後，再除以 2，它的輸出公式如(式 2.5)：

$$f\,\text{out}(x,\text{n}) = \begin{cases} f_o(x,n),\ y \bmod 2 = n \bmod 2 \\ f_i(x,n) = f[(\vec{x},n-1) + f(\vec{x},n+1)]/2 \end{cases} \qquad (式\ 2.5)$$

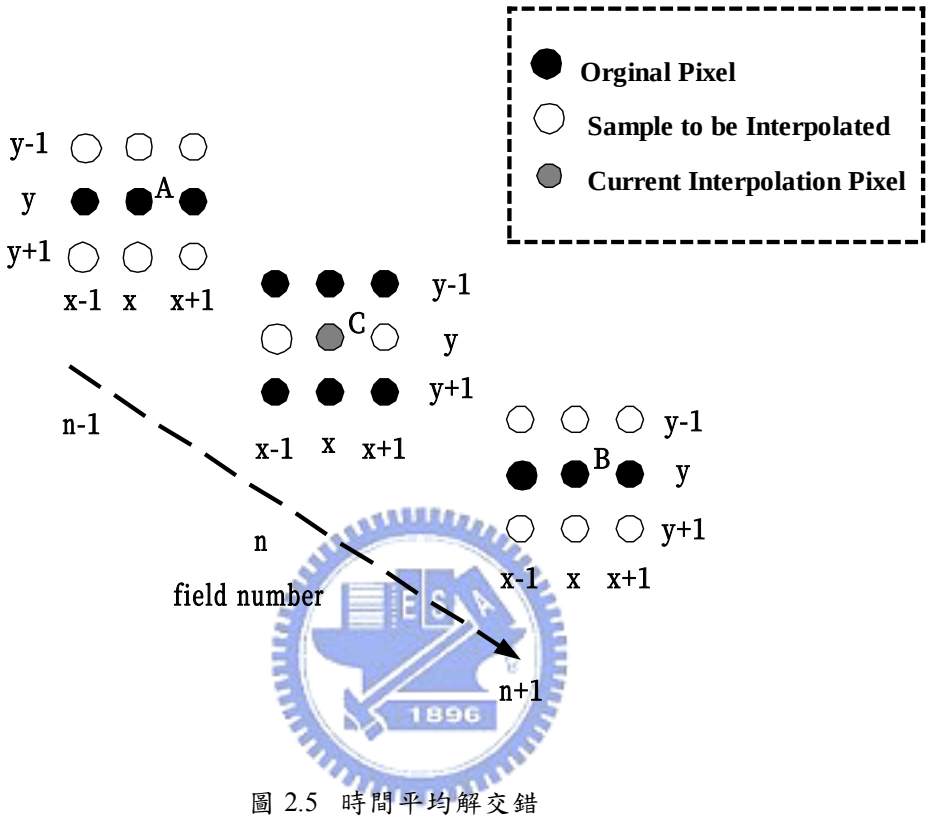


圖 2.5 時間平均解交錯

2.2.3 可適性移動補償解交錯

移動可適性解交錯（Motion Adaptive De-interlacing）是利用移動偵測來決定解交錯演算法，是使用空間掃描線垂直平均或是時間平均解交錯，所以移動偵測的正確，就變成相當重要，尤其是在交錯式輸入影像，連續的圖場像素，不能直接去比較同一位置的像素差異值，來判別是否有物件在畫面中移動。因此我們必須用圖場中或圖場間的很多資訊作計算，來判別每一點是否移動，找到適合解交錯演算法。在過去文獻中，已經有很多移動偵測的演算法被提出。在圖 2.6 是一種較常被使用作移動偵測的演算法，首先計算(A 與 C)、(B 與 C)、(F 與 G)、(H 與 G)的絕對差異值，然後取其四個值的平均值等於 X，再計算(D 與 E) 的絕對差異值等於 Y，比較 X 與 Y 的值，數值比較大的，即為移動的數值 M，經由實驗統計分析，可找到 M 與 β 之間的關係式，其中 β 是移動偵測的因素（Factor），因此移動可適性解交錯的數學式如（式 2.6）所示：

$$f\text{out}(\vec{x},n)=\begin{cases} f_o(\vec{x},n),\ y \bmod 2 = n \bmod 2 \\ f_i(\vec{x},n)=\beta f\text{st}(\vec{x},n)+(1-\beta)f\text{mot}(\vec{x},n),\ \text{otherwise} \end{cases} \qquad \text{（式 2.6）}$$

fst 是在靜態影像插補演算法

$f$mot 是在動態影像插補演算法

β 是移動偵測的圖素（Factor），範圍介於 0～1 之間

當 β＝0 表示偵測的結果是動態影像

當 β＝1 表示偵測的結果是靜態影像

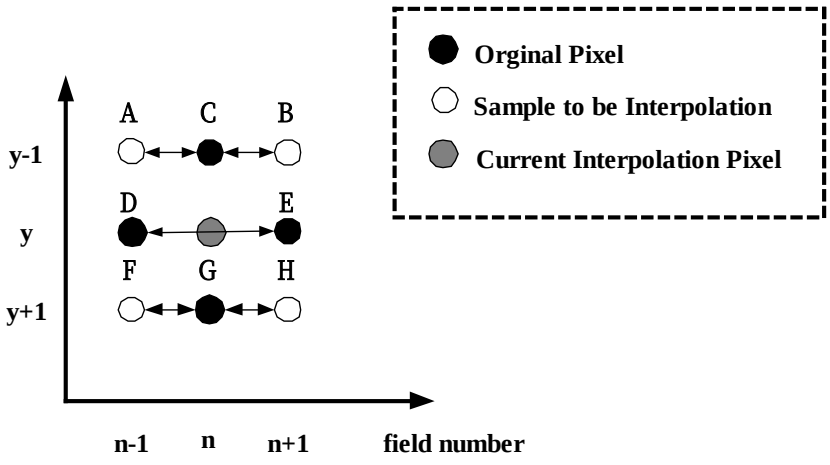


圖 2.6 移動偵測 示意圖

移動偵測在判別移動或靜止影像時，很容易受到雜訊的干擾，所以它必須在硬體上設計一些低通濾波器或是高通濾波器等，來避免外界雜訊的影響。除此之外連續輸入的圖場，在相同的位置上會相差一條掃描線，所以如何有效且正確辨別圖場的移動區域，也是另外一項研究的重點。

2.2.4 混合式解交錯

移動可適解交錯是目前在商業產品上，使用最多的演算法，但是它在物體做低角度移動時，所得到的效果，就不會令人滿意。針對這個問題，就有角度偵測（Edge Detection）的演算法[3]被提出，圖 2.7 是它的示意圖。從圖中我們可以知道，要決定目前插點 X 的值，要先尋找(A 與 R)、(B 與 Q)、(C 與 P)、(D 與 O)、(E 與 N)、(F 與 M)、(G 與 L)、(H 與 K)、(I 與 J) 等每一對點對點之間最小的差異值對。假設找到的最小的差異值對是(A 與 R) ，那 X 的值即為(A 和 R)相加的平均值。我們可以推測此一影像的紋路方向為(A,X,R) 。

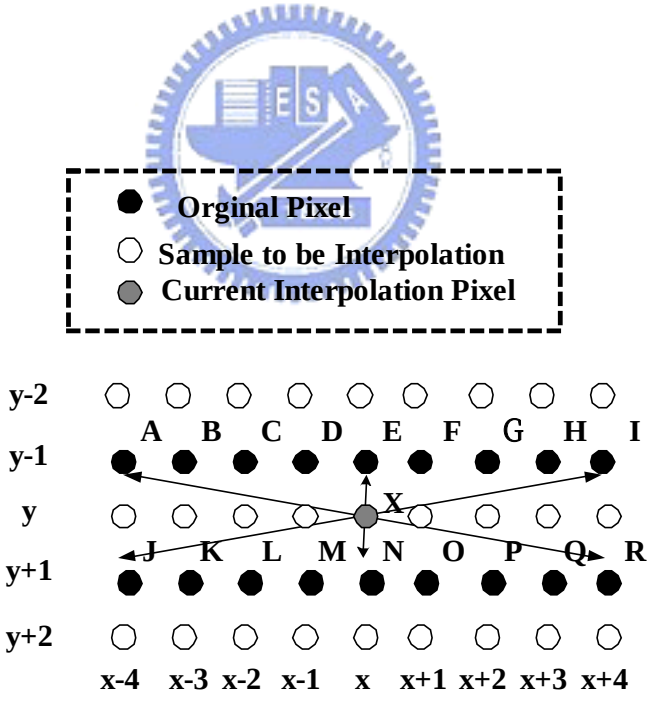


圖 2.7 角度偵測掃描線平均解交錯

在(式 2.7)中定義了 A、B、C、D、E、F、G、H、I、J、K、L、M、N、O、P、Q、R 等每一像素相對應的位置，而(式 2.8)則定義 XAR、XBQ、XCP、XDQ、XEN、XFM、XGL、XHK、X I J 等可能插點到 X 的值。經由這些式子和之前所提到移動可適性解交

錯方式，我們可以得到如同(式 2.9)的數學式，它結合移動可適性解交錯和角度偵測的演算法，而合成混合式解交錯的演算法。它的優點是物體在各種角度時，都有很好的表現。

$$
\begin{aligned}
&A = fo(\vec{x} - 4\vec{u}_x - \vec{u}_y, n) && J = fo(\vec{x} - 4\vec{u}_x + \vec{u}_y, n)\\
&B = fo(\vec{x} - 3\vec{u}_x - \vec{u}_y, n) && K = fo(\vec{x} - 3\vec{u}_x + \vec{u}_y, n)\\
&C = fo(\vec{x} - 2\vec{u}_x - \vec{u}_y, n) && L = fo(\vec{x} - 2\vec{u}_x + \vec{u}_y, n)\\
&D = fo(\vec{x} - \vec{u}_x - \vec{u}_y, n) && M = fo(\vec{x} - \vec{u}_x + \vec{u}_y, n)\\
&E = fo(\vec{x} - \vec{u}_y, n) && N = fo(\vec{x} + \vec{u}_y, n)\\
&F = fo(\vec{x} + \vec{u}_x - \vec{u}_y, n) && O = fo(\vec{x} + \vec{u}_x + \vec{u}_y, n)\\
&G = fo(\vec{x} + 2\vec{u}_x - \vec{u}_y, n) && P = fo(\vec{x} + 2\vec{u}_x + \vec{u}_y, n)\\
&H = fo(\vec{x} + 3\vec{u}_x - \vec{u}_y, n) && Q = fo(\vec{x} + 3\vec{u}_x + \vec{u}_y, n)\\
&I = fo(\vec{x} + 4\vec{u}_x - \vec{u}_y, n) && R = fo(\vec{x} + 4\vec{u}_x + \vec{u}_y, n)
\end{aligned}
\qquad \text{（式 2.7）}
$$

$$
\begin{aligned}
&XAR = (A + R)/2 && DEAR = |A - R|\\
&XBQ = (B + Q)/2 && DEBQ = |B - Q|\\
&XCP = (C + P)/2 && DECP = |C - P|\\
&XDQ = (D + O)/2 && DEDQ = |D - Q|\\
&XEN = (E + N)/2 && DEEN = |E - N|\\
&XFM = (F + M)/2 && DEFM = |F - M|\\
&XGL = (G + L)/2 && DEGL = |G - L|\\
&XHK = (H + K)/2 && DEHK = |H - K|\\
&XIJ = (I + J)/2 && DEIJ = |I - J|
\end{aligned}
\qquad \text{（式 2.8）}
$$

$\vec{u}_y = (0,1)^T$　　T 為轉換函數

$$
f\text{out}(\vec{x}, n) = \begin{cases} f\text{o}(\vec{x}, n)\text{, y mod 2} = \text{n mod 2} \\ f_i = [f\text{o}(\vec{x}, n-1) + f\text{o}(\vec{x}, n-1)]/2, \quad \text{物體靜止} \\ f_i = \min(\text{DEAR, DEBQ, DECP, DEDQ, DEEN, DEPM, DEGL,} \\ \qquad \text{DEHK, DEIJ}), \quad \text{物體紋路方向} \end{cases}
\qquad \text{（式 2.9）}
$$

2.3 移動補償解交錯介紹

大多數先進的解交錯演算法，都是利用移動補償[4]的方式，來作為產品的主要技術。因為越來越多的消費者能接受比較昂貴的價格，在一些高級電視機中，都應用到這樣的技術。

在介紹移動補償解交錯的定理前，我們將先定義一些參數。我們使用 $d(\vec{x}, n) = (dx(\vec{x}, n), dy(\vec{x}, n))$ 來描述移動向量，其中 dx 代表水平的方向，dy 代表垂直的方向。在交錯掃描影像輸入時，並不是所有時間的資訊變化，都能被適當的描述而得到正確地移動向量。像場景改變時，或者背景變微弱時及在矇朧背景情況下等，都是比較困難的部份。不過，這物體的移動模式有很強烈自然定律，如物體的移動有物理慣性，在時間改變中不會突然消失，或突然改變物體時幾何形狀，所以我們可以運用這個自然定律，找到正確的移動向量，適當補償到要插入的點。在這個章節中，我們將討論各種移動補償演算法。

2.3.1 基本移動補償解交錯

基本移動補償解交錯的方式是利用移動估計的技術，來找到每一個插點的移動向量，進而從相對應移動向量的像素中，插入要插補的像素。移動估計主要是運用區塊與區塊之間的比對方式，所以可以很準確的找到移動向量，經由圖 2.8 移動補償解交錯的示意圖，我們可得知圖中 D 的值，經移動向量後得到 C 的值，從數學的觀點來看，可得到(式 2.10)。因為基本移動補償解交錯的方式是在理想的情況下，才會有很好的結果。但實際上會有各種不同的情形發生，例如畫面中有很多雜訊之類，所以下面將有其它改善的演算法被提出。

$$fout(\vec{x}, n) = \begin{cases} fo(\vec{x}, n) & , y \bmod 2 = n \bmod 2 \\ fi(\vec{x} - \vec{d}(\vec{x}, n), n-1) & , \text{otherwise} \end{cases} \qquad (式\ 2.10)$$

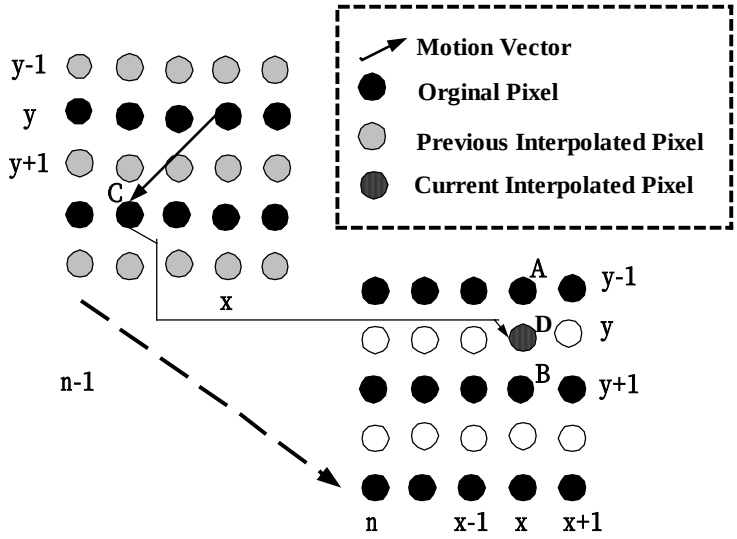


圖 2.8 移動補償解交錯

2.3.2 時間遞迴解交錯

時間遞迴解交錯（Time Recursive De-interlacing）是被 Wang et al. [5]提出的。它是利用前一張已作過解交錯的圖場來作移動補償，這種的想法是很簡單概念，但它可用強而有力來形容。

不過，要讓時間遞迴解交錯達到最大的效果，我們必需保證已經得到正確的移動向量，然後將前一張的圖場運用廣義取樣定理[6][7][8]的方式，取得原始像素與插點之後的像素，作為目前圖場將要插點的像素。我們可從圖 2.9 中，可看出目前要插入的像素，通常取自上一張圖場的原始像素，與插點後的中間值，但如此的作法會因為兩張圖場基準點不同，就如某一個時間點圖場它是由偶數線組成，那它前一張或下一張的圖場則由奇數線組成，所以當前一張畫面由圖場變為圖框之後，假設它移動補償的像素是錯誤的，那之後圖場的移動補償插點也都不會正確，這是它最主要的缺點。

為了彌補這個的缺點，可利用中值濾波器（Median Filtering）作一個保護機制，讓時間遞迴解交錯的演算法，變成非常地有效益，它的輸出數學式可定義成(式 2.11) :

$$f_{out}(\vec{x}, n) = \begin{cases} f_o(\vec{x}, n) & y \bmod 2 = n \bmod 2 \\ \text{median}\begin{cases} f_i(\vec{x} - \vec{d}(\vec{x}, n), n-1) \\ f_o(\vec{x} - \vec{u}_y, n) \\ f_i(\vec{x} + \vec{u}_y, n) \end{cases} & \text{otherwise} \end{cases} \quad \text{（式 2.11）}$$

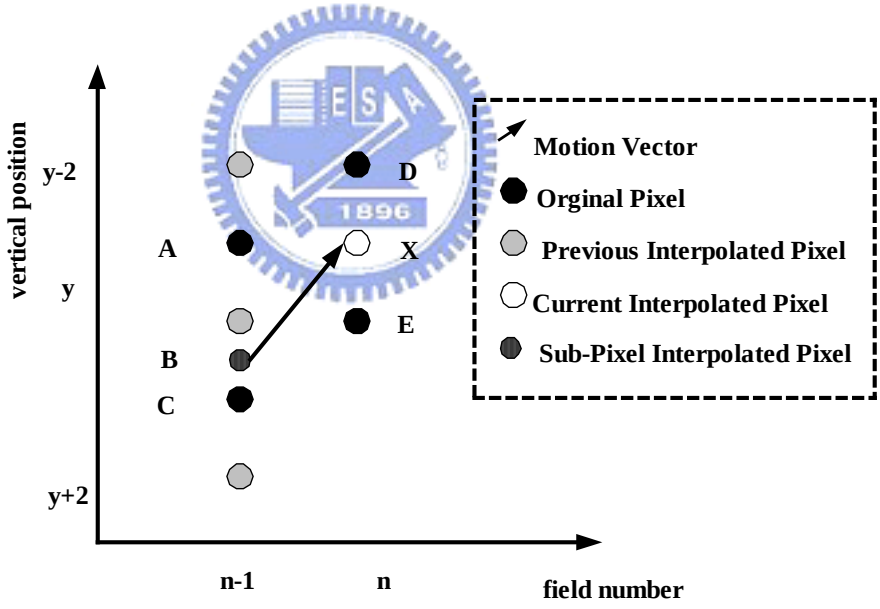


圖 2.9 時間遞迴解交錯

2.3.3 可適性遞迴解交錯

可適性遞迴解交錯（Adaptive Recursive De-interlacing）是被 De Haan et al. [9] 提出的，它是運用一階遞迴時間濾波器（First-Order Recursive Filtering）來作插點的移動補償，式 2.13 是它的數學表示式。從這數學式子，需要在作每一插點時，都要重新計算 K1、K2 的參數值，所以這種解交錯的演算法，假設每一次都能找到一個最適當 K1、K2 的值，則可避免在移動估計時，因為找到不正確的移動向量，或是移動向量需整數處理而捨去小數點誤差，而造成了移動軌跡不平滑，進而產生的鋸齒狀（Aliasing）的畫面。從圖 2.11 示意圖中，我們可清楚的了解它在作移動補償解交錯時，僅需要兩張圖場的記憶體，這些都是它的優點。但也因為可適性遞迴解交錯的演算法，在每一插點都需要大量且複雜的計算，而成為它最大的缺點。

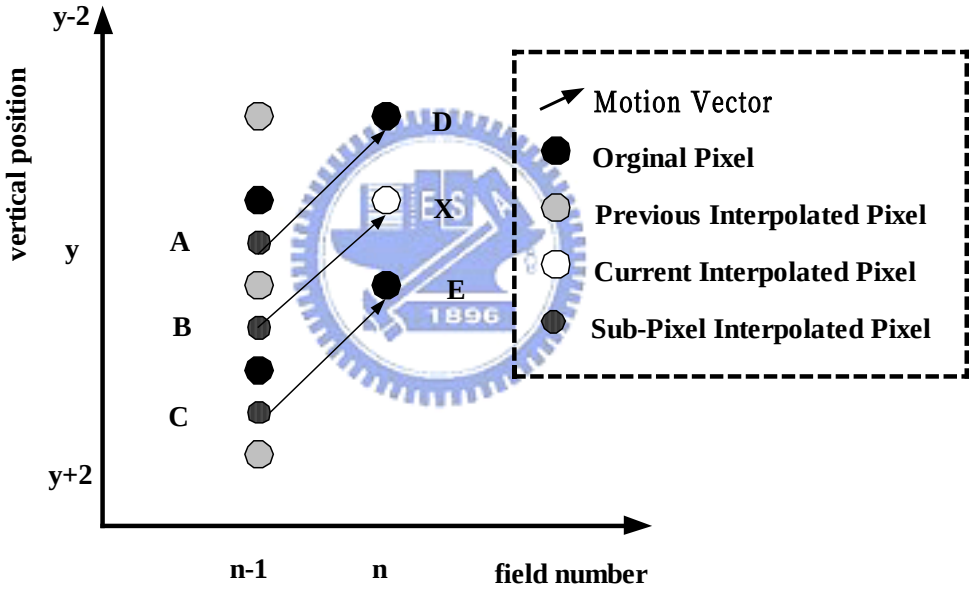


圖 2.10 可適性遞迴解交錯

$$f_{out}(\vec{x}, n) = \begin{cases} K_1(\vec{x}, n) f_i(\vec{x}, n) + (1 - K_1(\vec{x}, n)) f_o(\vec{x} - \vec{d}(x, n), n-1), \\ \quad (y \bmod 2 = n \bmod 2) \\ K_2(\vec{x}, n) f_i(\vec{x}, n) + (1 - K_2(\vec{x}, n)) f_o(\vec{x} - \vec{d}(x, n), n-1), \\ \quad (\text{otherwise}) \end{cases} \quad (\text{式 } 2.13)$$

$$K_2(\vec{x}, n) = \frac{|A+B| + \delta}{2|f_i(\vec{x}, n) - f_o(\vec{x}-\vec{d}(\vec{x}, n), n-1)| + \delta} \quad (式\ 2.14)$$

$$A = f_o(\vec{x}-\vec{u_y}, n) - f_o(\vec{x}-\vec{d}(\vec{x}, n) - \vec{u_y}, n-1)$$
$$B = f_o(\vec{x}+\vec{u_y}, n) - f_o(\vec{x}-\vec{d}(\vec{x}, n) + \vec{u_y}, n-1) \quad (式\ 2.15)$$

δ 是一個常數

$u_y = (0, 1)^T$　　T 是轉換函數

$$K_1(x, n) = clip(0, 1, C1\sqrt{Diff}) \quad (式\ 2.16)$$

$$clip(0, 1, a) = \begin{cases} 0, (a<o) \\ 1, (a>1) \\ a, (otherwise) \end{cases} \quad (式\ 2.17)$$

$$Diff(\vec{x}, n) = |fo(\vec{x}-\vec{d}(\vec{x}, n), n-1) - fi(\vec{x}, n)| \quad (式\ 2.18)$$

在上面的(式 2.14)、(式 2.15)是計算 K2 的數學運算式，其中 δ 是一個常數，它的用意主要是保護(式 2.14)的分母不為 0，並使得(式 2.14)為一個合適的數值，若分母為 0，(式 2.14)就無法計算數值了。另外(式 2.16)、(式 2.17) 、(式 2.18) 是計算 K1 的數學運算式，其中最重要的參數是 C1，它必需透過很多的實驗數據，去歸納統計找到最適當的值，所以可適性遞迴解交錯的演算法是相當複雜數學式，如果要得到很好的影像品質，就必須不斷修正 K1、K2 的參數值。

2.4 解交錯系統架構

在本文的第三章和第四章，將提出一個整合式解交錯的架構，但在此之前，我們需要分析在過去文獻中，已經被提出的解交錯架構，在圖 2.11 的解交錯架構[10]，它至少需要四張圖場的記憶體，主要的方塊圖包括雙向移動估計 (Bi-directional Block-base Motion Estimation) 方塊、移動向量修正偵測(Motion Vector Correctness Detection)方塊、權重產生器(Weighting Generation)方塊、圖場內內插(Intra-Field Interpolation)方塊、圖場間內插(Inter-Field Interpolation) 方塊等組成。它的優點是針對錯誤的移動向量有很多偵測補償的機制，但它並未考慮影片偵測及雜訊干擾之類的問題，而且對於何時使用圖場內解交錯、圖場間解交錯及移動補償解交錯的機制，並沒有建立出來，這些都是需要改進的地方。

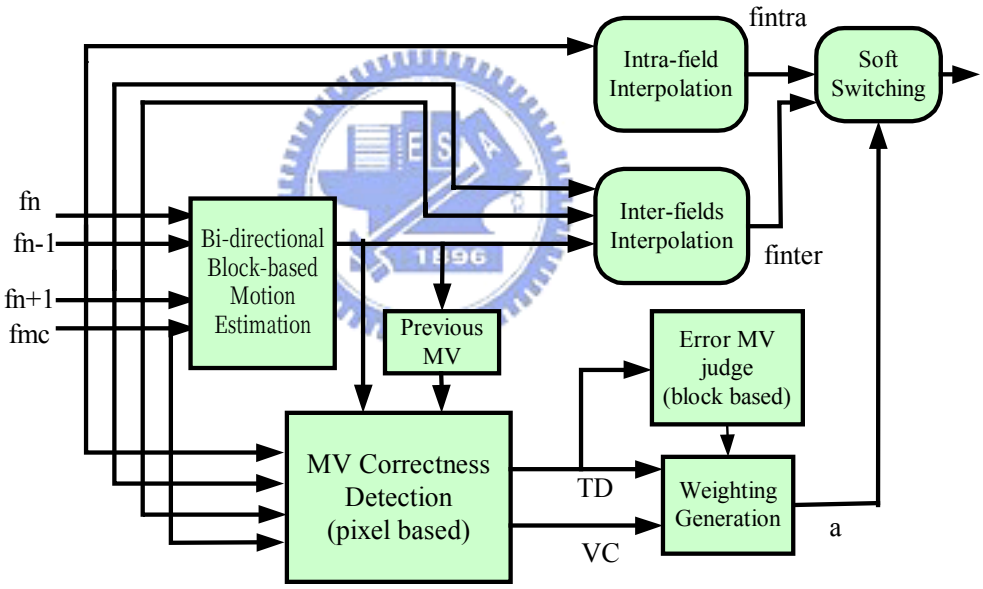


圖 2.11 移動補償解交錯架構參考(一)

從圖 2.12 中，我們可以了解另一個解交錯的架構[11]，它主要的方塊圖包括移動補償圖場插點(Motion Compensated Field Interpolation，簡稱 MCFI)方塊、圖場間內插(Field Merging)方塊、移動向量修正偵測(Rearrangement)方塊等。其中比較特殊的為移動補償圖場插點方塊，它含有移動估計、移動向量平滑處理及圖場插點等功能。這個架構也需

要有三張圖場的記憶體，在移動估計時，使用雙向相同圖場來找移動向量，由於點對點比對的絕對差異總合（SAD），同為偶數圖場或奇數圖場，所以很容易找到正確的移動向量，這是它的優點。但是當畫面中的物體在快速移動時，因為超出移動估計的搜尋範圍，可能無法找到最好的移動向量，而且在移動估計不正確時，應設計另一種解交錯的演算法來代替移動補償解交錯，形成保護機制，不然會有很差的畫面出現，這些都是需要加強的部份。

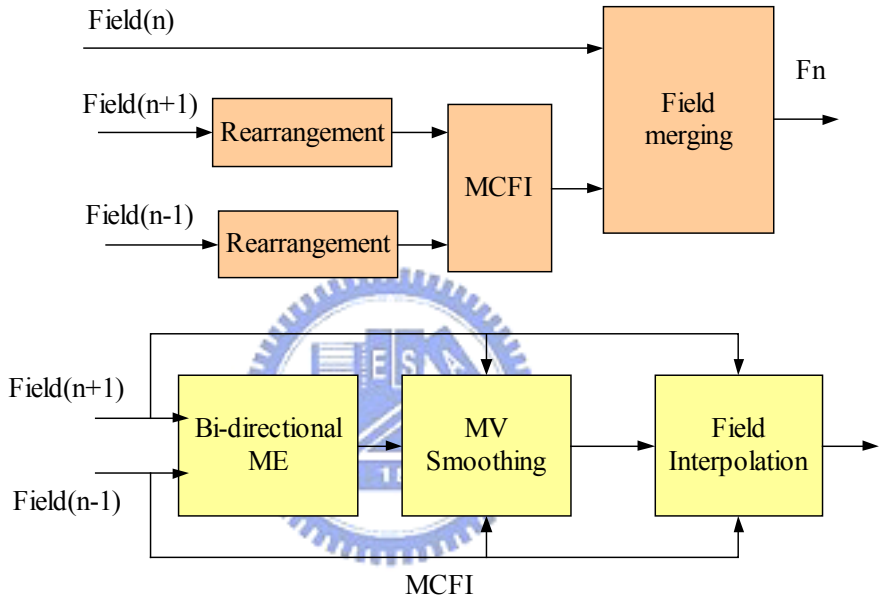


圖2.12 移動補償解交錯架構參考(二)

第三章 整合式解交錯演算法設計

3.1 整合式解交錯架構

本文所提出的整合式移動補償解交錯的架構，是以三張圖框為基礎的系統架構。其中的方塊圖主要包括移動估計方塊、移動偵測方塊、移動補償方塊、影像前置處理方塊、影像後置處理方塊、移動向量平滑方塊、影片偵測方塊，及記憶體方塊等。圖 3.1 是移動補償解交錯架構方塊圖，其中實線表示資料路徑，虛線表示控制信號，在這章節（第三章）及下一個章節（第四章）我們將詳細說明每一個方塊的設計理論。

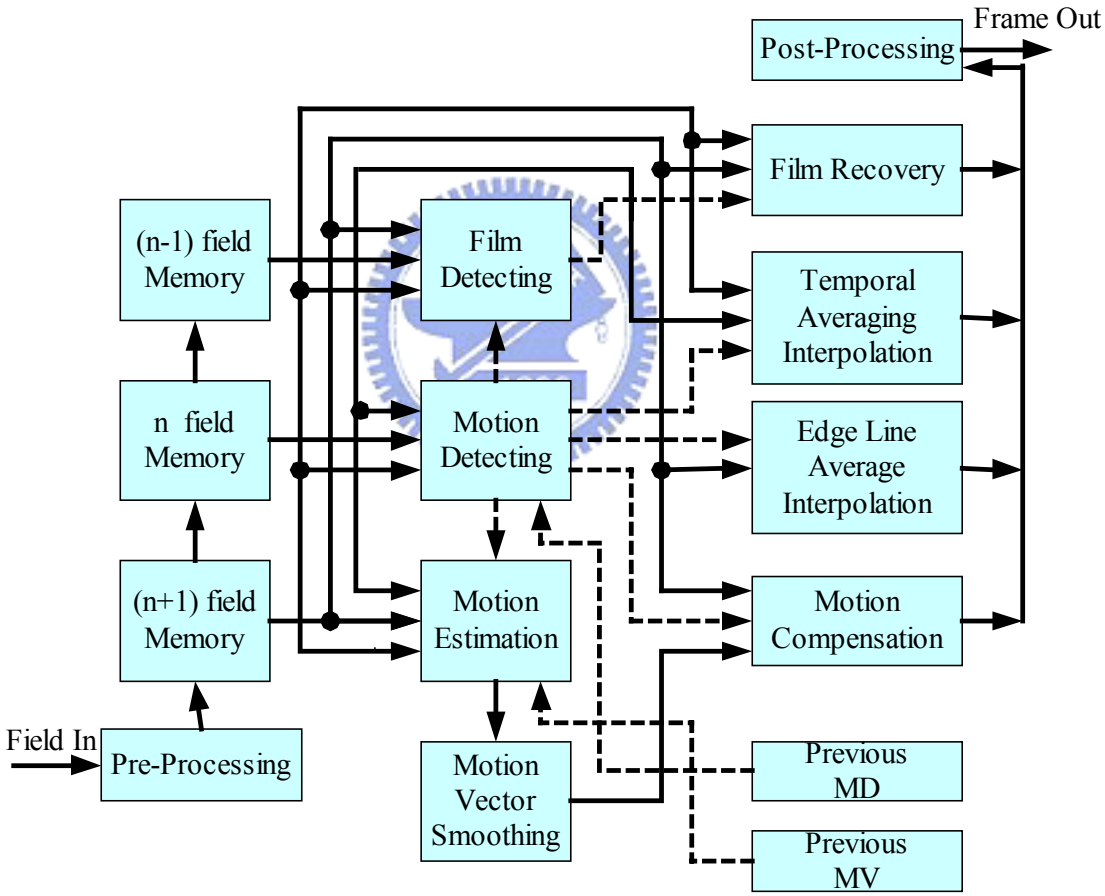


圖3.1 整合式移動補償解交錯的架構

在下圖 3.2 中，將說明移動補償解交錯的資料處理流程。透過這個資料處理的流程，讓我們更容易了解整個移動補償解交錯的架構，及每一個方塊所扮演的角色。

在移動補償解交錯的架構中，記憶體的大小會影響硬體的成本，所以本文所提出的架構，是以最小記憶體的概念，來設計每一個區塊的演算法。這樣的想法，是期望未來有機會能在硬體上實現，為了驗證每一個演算法的正確性及可行性，我設計一套移動補償解交錯模擬工具，藉由這套模擬的工具，可清楚地了解每一個演算法的效果。

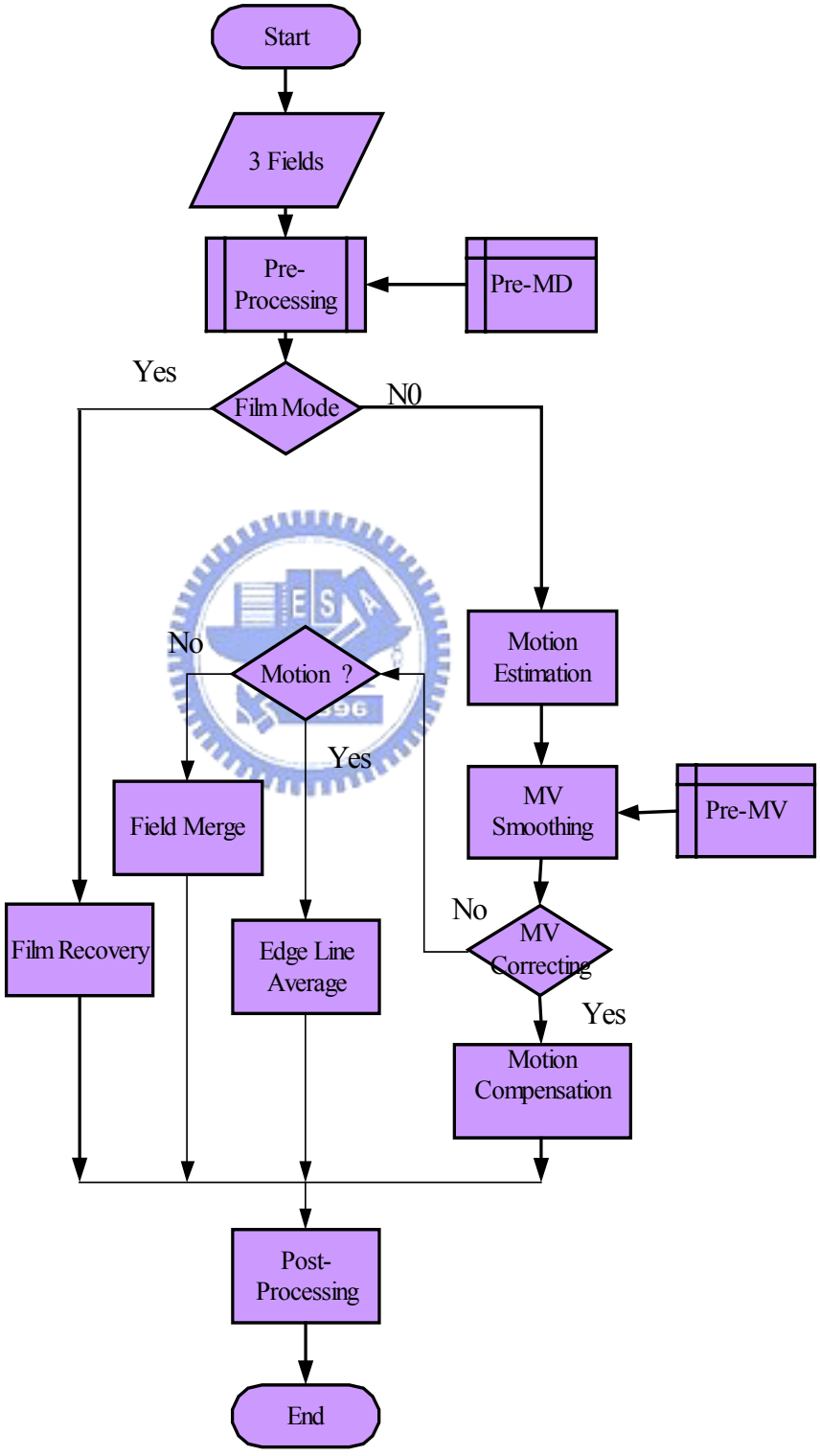


圖3.2 整合式移動補償解交錯的資料流程圖

3.2 移動估計的設計

移動估計（Motion Estimation）的演算法，是設計在移動補償解交錯的架構中。這是相當重要的系統方塊，因為當移動估計錯誤時，錯誤的移動向量很難作正確的移動補償而還原真實影像，因此移動估計的目地主要是找到畫面間的移動向量，如圖 3.3 所示。輸入影像的資料格式是交錯格式，我們可以假設目前的畫面是偶數圖場，前一個畫面則是奇數圖場，同理若是目前的畫面是奇數圖場，前一個畫面則是偶數圖場。這表示我們若要補償目前的畫面偶數圖場成一個圖框，就必需從目前的偶數圖場，和利用前一個奇數圖場作移動估計，而得到目前的偶數圖場奇數線，而合成一個圖框，這一連串的過程即可稱為解交錯，如下圖 3.4 所示。

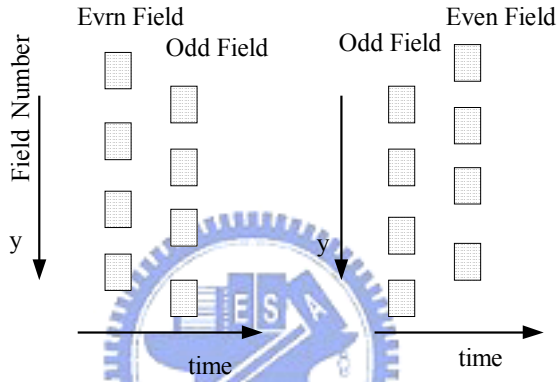


圖3.3 交錯式的資料格式

在這些過程中，我們知道移動估測運用在影像壓縮，和運用在交錯掃描格式移動估計是有很大的不同。不同的地方是影像壓縮在作移動估測，主要減少平均預測的錯誤值（Average Predictive Error），並針對整個畫面，找尋移動向量。而運用在交錯掃描格式時，可以比較的資料只有一半的畫面，所以如何找到真實的移動向量，就顯得相當重要。

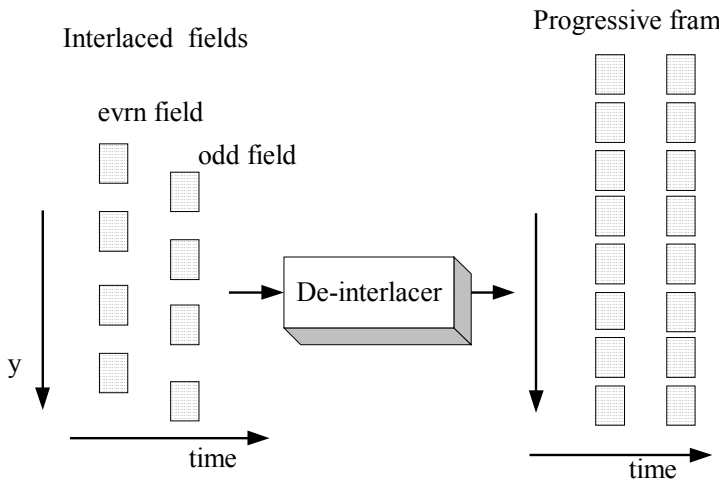


圖3.4 資料格式的轉換

3.2.1 交錯掃描格式的移動估測設計

交錯掃描格式的電視訊號是由連續的圖場所組成的，而交錯掃描格式移動估計是要從相鄰的兩個圖場，找到每一個像素的移動向量，而移動向量的定義，即由圖 3.5 所示。由現在的影像往前一張找到最相似的區塊，我們可將(X1, Y1)－(X2, Y2)作為移動向量。

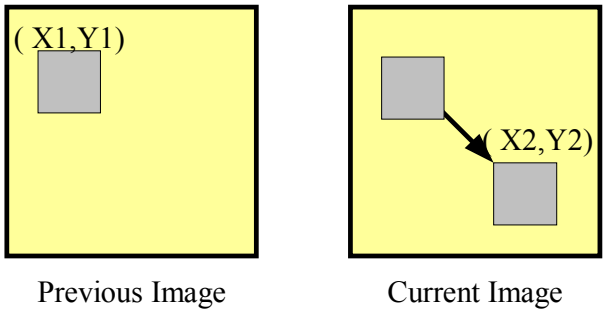


圖 3.5　移動向量定義

解交錯掃描格式轉換過程中，移動向量的估測，若是用連續的兩個圖場作為比對的標準，我們可從圖 3.6a 得知，虛線表示不存在的掃描線，實際的影像應如圖 3.6b 所示。偶數圖場只有偶數線，奇數圖場只有奇數線，若是用連續兩個圖場作為移動估計，垂直的方向就有一條線的差異。但如果考慮的是奇數場對奇數場，偶數場對偶數場，就如圖 3.7a 和圖 3.7b 所示。也就是對相同的圖場畫面作移動估計，那就可以避免上述的問題產生，但卻又衍生另一個問題，假設有一個物件移動速度相當的快，它將超出在兩場的移動搜尋範圍。因此可能將找不到正確的移動向量，但是如果只隔一個圖場的時間，通常就不會有此問題，所以在此我們將探討下列幾種方法。

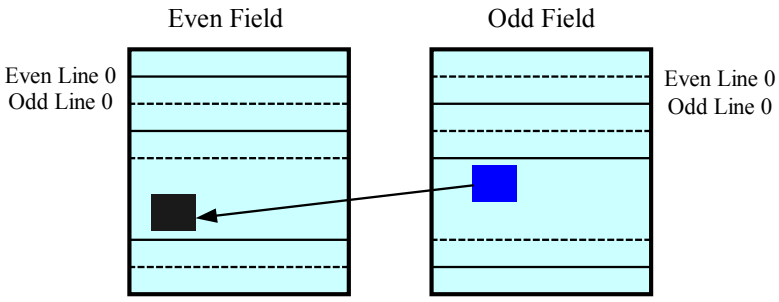


圖3.6a 理論上偶數圖場對奇數圖場的移動估計

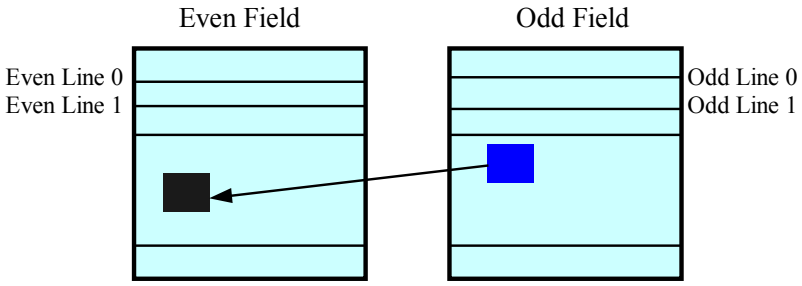


圖3.6b 實際上偶數圖場對奇數圖場的移動估計

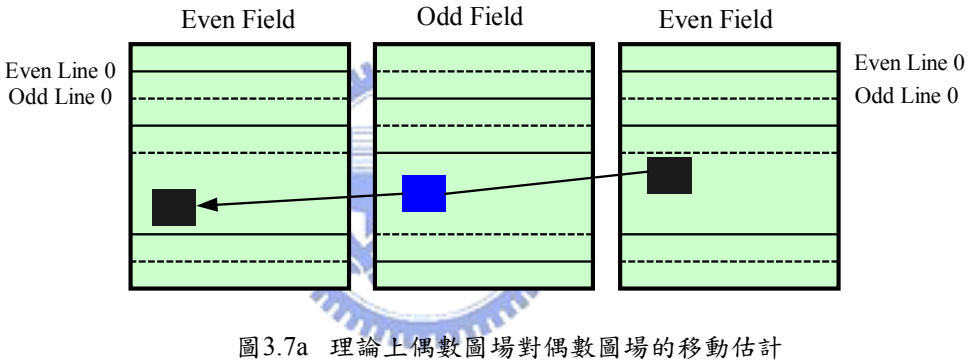


圖3.7a 理論上偶數圖場對偶數圖場的移動估計

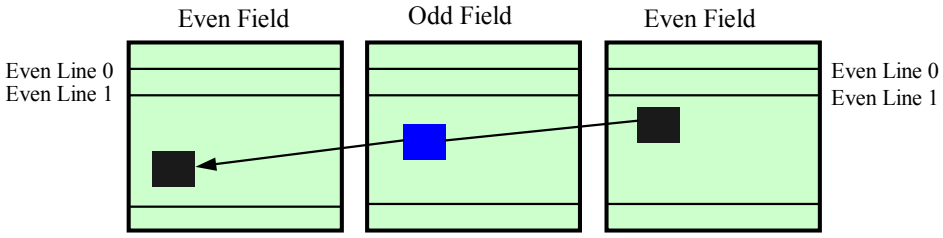


圖3.7b 實際上偶數圖場對偶數圖場的移動估計

在一般影像處理系統中，以方塊比對的移動估計使用最廣泛，主要是硬體計算非常簡單，搜尋的方式也有很多已知方法可以運用。目前最常被拿來作移動向量估測的比對

準則（match criterion），是使用的誤差方程式（Error function）中絕對差異的總和（Sum of Absolute Difference, SAD）。在 MPEG 中，通常是在連續的全場畫面中，計算絕對差異的總和。不同於解交錯掃描格式轉換，移動向量估測所用來計算絕對差異總和的方法，在解交錯掃描格式轉換中，連續的圖場畫面有奇數圖場和偶數圖場的差異。如果直接拿相鄰的兩個圖場，來計算絕對差異的總和是不適當的，會發生一些問題。尤其是在畫面垂直亮度訊號的高頻成分很大時，影響更為嚴重。

依據交錯掃描格式圖場的不同，我們將移動估計作法，大約分成下列五種方式：

方法一： 偶數圖場對奇數圖場模式 1 的移動估計，如圖 3.8a 所示，若目前的圖場是奇數圖場，而它前一張是偶數圖場。其中黑色的點，表示有資料；白色的點，表示無資料，所以用兩張圖場的資料，來作移動估計的區塊比對，則會有一條線的差異。

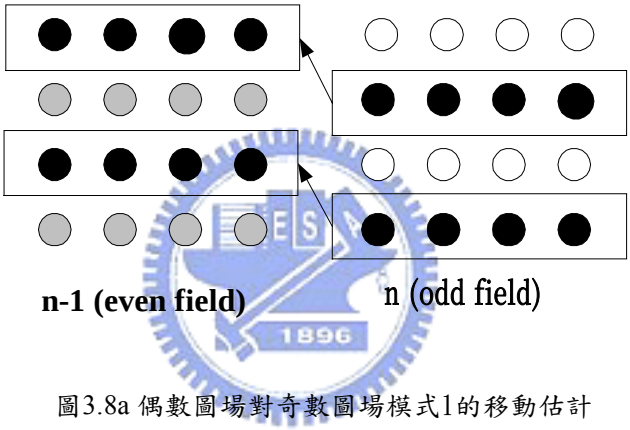


圖3.8a 偶數圖場對奇數圖場模式1的移動估計

方法二： 偶數圖場對奇數圖場模式 2 的移動估計，如圖 3.8b 所示。若目前的圖場是奇數圖場，而它前一張是偶數圖場，其中黑色的點表示有資料，白色的點表示無資料。因為用兩張圖場的資料來作移動估計的區塊比對，則會有一條線的差異，所以可利用奇數圖場奇數線，作垂直兩點的平均，而得到偶數線，再與前一張偶數圖場作移動估計。

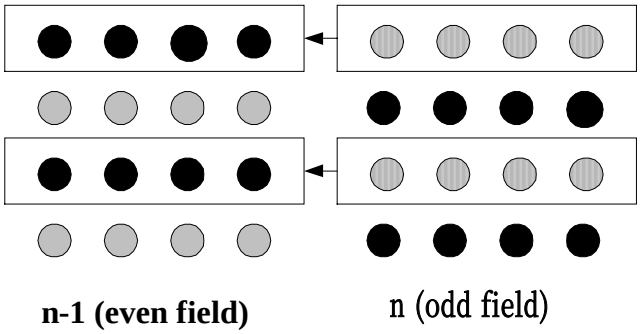


圖3.8b 偶數圖場對奇數圖場模式2的移動估計

方法三： 偶數圖場對奇數圖場模式 3 的移動估計，如圖 3.8c 所示。若目前的圖場是奇數圖場，而它前一張是偶數圖場，其中黑色的點表示有資料，白色的點表示無資料，灰色的點表示已插點過的資料。可利用這些點和目前的圖場點作移動估計，而不會有位置的差異。不過這種方式必需保證已插過點是正確的，否則解交錯的畫質會隨時間越來越差。

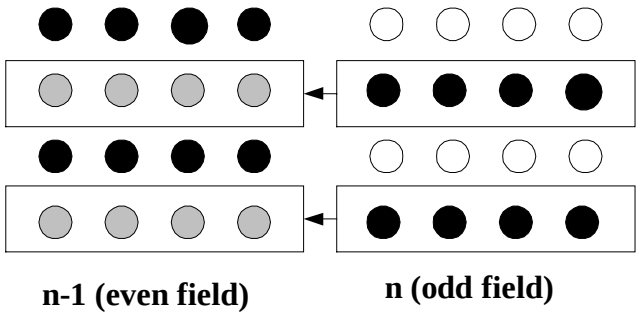


圖3.8c 偶數圖場對奇數圖場模式3的移動估計

方法四： 偶數圖場對偶數圖場的移動估計，如圖 3.8d 所示。若目前的圖場是奇數圖場，而它前一張是偶數圖場，下一張也是偶數圖場，其中黑色的點表示有資料，利用前一張偶數圖場，和下一張偶數圖場作移動估計，而不會有位置的差異。但當物體移動速度很快時，移動向量可能已經超過搜尋範圍。

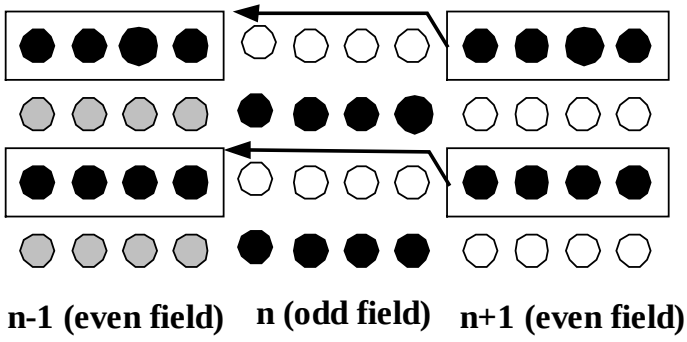


圖3. 8d 偶數圖場對偶數圖場的移動估計

方法五： 雙向偶數圖場對奇數圖場的移動估計[12]，如圖 3.8e 所示。若目前的圖場是奇數圖場，而它前一張是偶數圖場，下一張也是偶數圖場，由目前的奇數圖場分別向前一張，及下一張的偶數圖場作移動估計，然後將兩個移動估計的值作平均的處理。

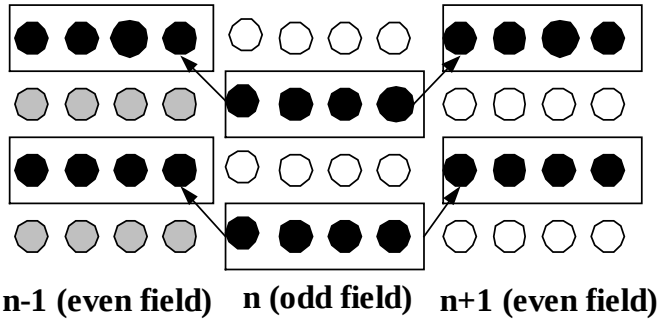


圖3.8e 雙向偶數圖場對奇數圖場的移動估計

各種移動估計的方式，是先取得循序掃描方式拍攝的自然圖片，由於它的寬和高的像素為 1920*1080，拿來觀察與分析並不方便。所以選取某一區域的大小為 720*480 的像素，連續取 20 張，將它作成交錯掃描的格式，依照偶數圖場、奇數圖場、偶數圖場、奇數圖場的排列順序，由循序掃描格式(720*480)轉換成偶數圖場時，需將奇數線的資料移除，因此偶數圖場變成 720*240。同理循序掃描格式(720*480)，轉換成奇數圖場時，需將偶數線的資料移除，因此奇數圖場變成 720*240，由這樣的作法及即可得到交錯掃描的格式。將這 20 張交錯掃描的格式，輸入到上述五種不同移動估計解交錯系統，可得到 720*480 循序掃描圖片(720*480)輸出，將這 20 張的輸出和原始的循序掃描圖片(720*480)作 MSE 的計算，可得到表 2 的結果。MSE 的數學式如（式 3.1）所式:

$$MSE = \frac{1}{N.\ X.\ Y} \sum_{n=0}^{N-1} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \left| \hat{F}(x, y, n) - F(x, y, n) \right|^2$$

（式 3.1）

N 表示圖場數，N = 20 張

X 表示圖場寬度 X = 720 像素

Y 表示圖場高度 Y = 480 像素

$F\left|x, y, n\right|$ 表示原始的循序掃描圖片

$\hat{F}\left|x, y, n\right|$ 表示解交錯後循序掃描圖片

經過實際模擬的結果，我們可得到表 2 中的結果。其中方法二和方法五有差不多的表現（Performance），而方法五是使用雙向偶數圖場對奇數圖場的移動估計。所以這種方式所需要的運算量、速度及硬體成本等都要雙倍的增加，因此最理想移動估計是採用方法二，偶數圖場對奇數圖場模式 2 的移動估計。

	fish (慢速移動)	tomato (鏡頭縮放)	traffic (快速移動)
方法一	0.7351	1.1415	1.2269
方法二	0.6665	1.1044	1.0394
方法三	0.7819	1.5515	2.0916
方法四	0.8629	1.3815	3.1943
方法五	0.6706	0.9949	1.1085

表 2：各種方式移動估計的比較表

3.2.2 移動估測搜尋範圍與方塊大小的設計

在影像壓縮標準中，有很多的移動估計的搜尋法已的文獻被提出。在本文中，將使用兩種搜尋法來作模擬：第一種是全區域搜尋法（full search）。全區域搜尋法是對於每一個區塊，其相對的搜尋範圍內，所有的位置都做 SAD 值的計算與比對，最後選取擁有最小 SAD 值的區塊位置，而相對的位移就是此區塊的移動向量。因為所有搜尋位置的區塊位置都要計算 SAD 值，所以保證可以找到最小的 SAD 值，也就是最相似的候選區塊。換言之，就可以找到最佳的移動向量，但是相對地，需要很高的運算量。

由於全區域搜尋需要大量的運算量，因此有許多運算量較小，且達到相似結果的演算法已被提出，統稱為快速搜尋演算法。這些快速演算法可以區分為兩類，第一類是只計算在搜尋區域當中，最有可能擁有最小 SAD 值的位置，如三步數搜尋法（Three-Step Search）、鑽石搜尋法（Diamond Search） 等。而另一類是減少計算 SAD 值時所需要的運算量，也就是將比對的兩個區塊作次取樣而不需要計算每一點的絕對差值，如階層式的搜尋法（Hierarchic Search）。

所以本文使用第二種搜尋法是階層式搜尋法，圖 3.9 是階層式的搜尋法的示意圖，

它是一種將搜尋的範圍由粗而細，從方塊 16*16 變 8*8 最後是 4*4。其中在方塊 16*16 大小時，是四個像素只取一個像素，方塊 8*8 大小時，是兩個像素只取一個像素，方塊 4*4 大小時則是每一個像素都比對。由於這樣的作法會減少計算 SAD 值時所需要的運算量，另外也對搜尋範圍做改變，方塊 8*8 大小搜尋的範圍減少為 4*4，方塊 4*4 大小時則減少為 2*2，從圖 3.10 我們就可以很清楚的了解它的搜尋過程。

以方塊為基準的移動估計，有兩個重要參數需要決定。其中一個參數是方塊的大小，當方塊選擇太大時，計算量就相地增加、硬體成本高。當方塊選擇太小時，在作方塊比對時，容易有誤判的情況發生，而且容易受到雜訊的干擾。

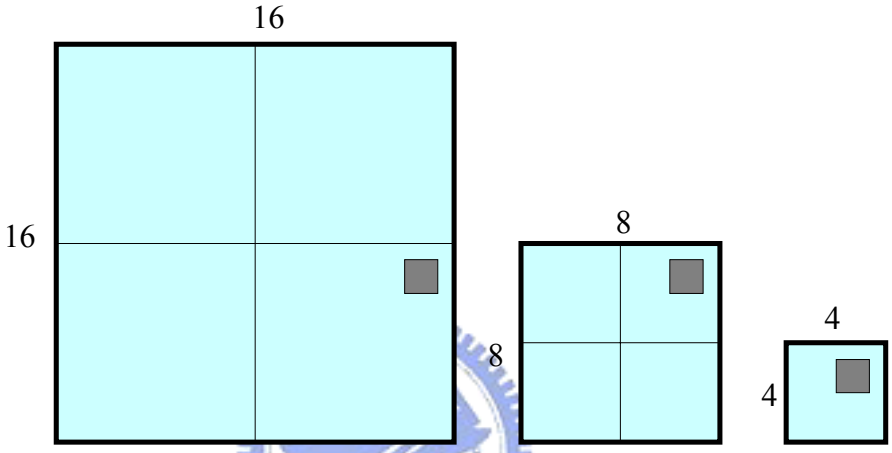


圖3.9 階層式搜尋法方塊

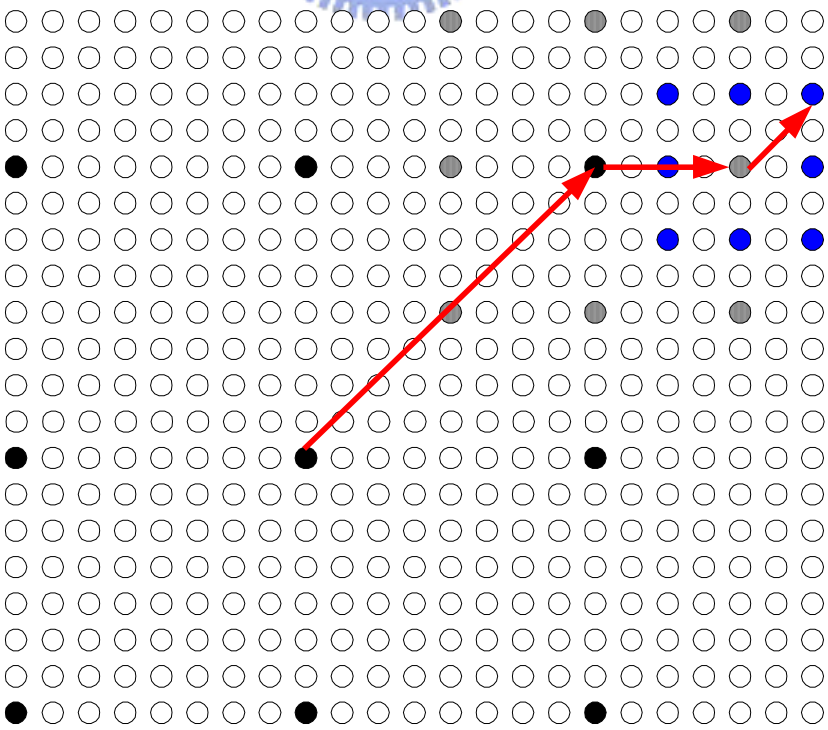
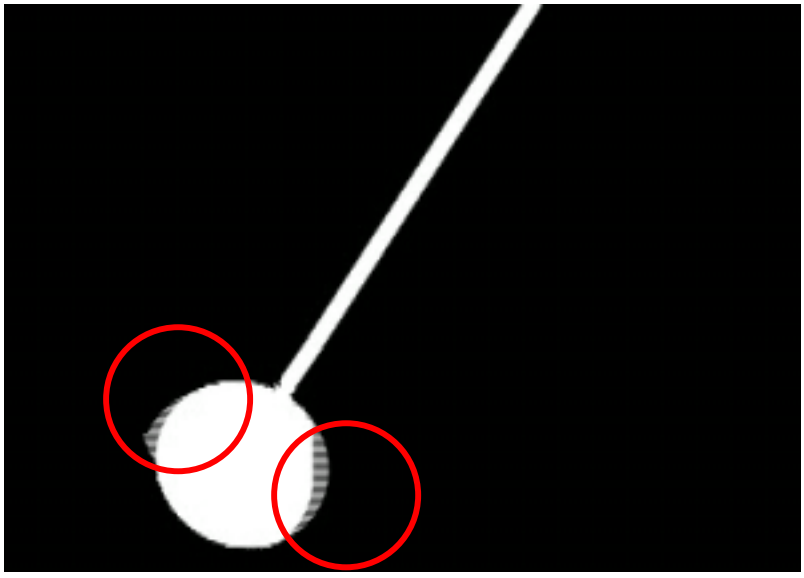


圖3.10 階層式搜尋法步驟

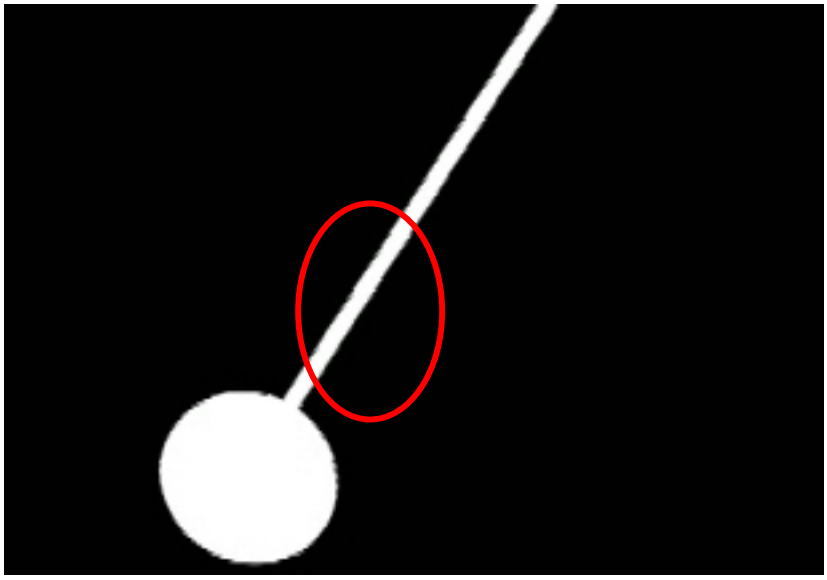
圖 3.11 是針對不同的區塊大小所模擬的結果，其中圖 3.11a 是區塊大小 4*4 移動估計，因為比對區塊小，就容易找到很多相似的區塊而造成誤判，所以在鐘擺球的邊緣就有很多鋸齒狀。若區塊設計太大時，除了增加運算量，也會在一些特別形狀的物體上，無法找到最好的區塊。圖 3.11b 是區塊大小 16*16 移動估計，鐘擺的地方就呈現不規則線。因此為了解決上述兩種的問題，可採用階層式的搜尋法，如圖 3.11 是階層式的搜尋法模擬的結果。此外，在表 3 中，我們可以得到全區域搜尋法，和階層式搜尋法的三種不同畫面 MSE 值。歸納這些數據，可以得到階層式搜尋法和全區域搜尋法有相同的效能，而且階層式搜尋法的速度快，是它的另一項優點。

	fish (慢速移動)	**tomato (縮放)**	**traffic (快速移動)**
全區域搜尋法	0.7351	1.1415	1.2269
階層式搜尋法	0.7167	1.1079	1.1127

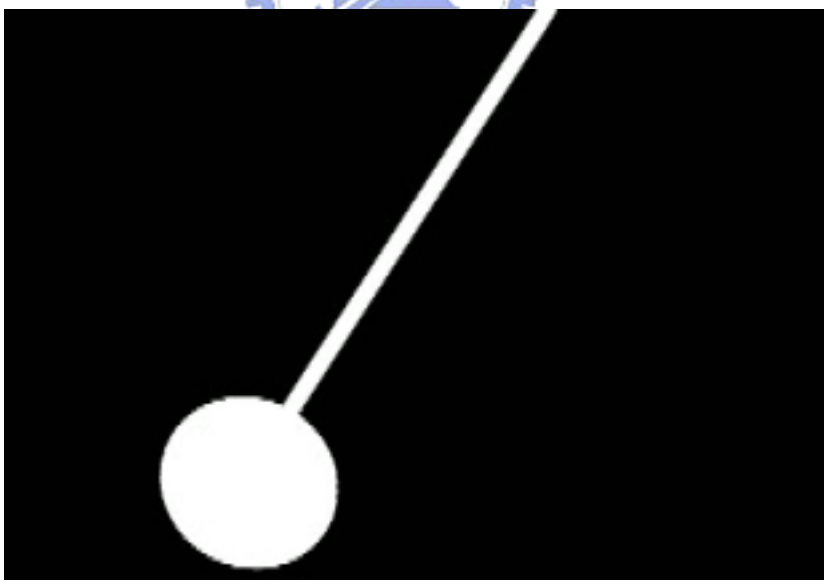
表 3：移動搜尋法的比較表



(a)



(b)



(c)

圖 3.11　不同移動估計搜尋法的模擬結果：(a) 4*4 區塊全區域搜尋法 (b) 16*16 區塊全區域搜尋法 (c) 階層式搜尋法 。

3.3 移動向量平滑處理的設計

影響移動向量測量的最主要因素包括影像本身的雜訊，和在作方塊比對時，方塊定義太小，或是搜尋範圍不夠大，都會影響移動向量的估計。我們假設一個移動物件，至少會大於數個方塊的大小，因此在相同的移動物體內若出現不同的移動向量，很可能出現了錯誤判斷，如圖 3.12a 所示。由此特性我們定義，包含要被處理的移動四週的資訊，總共 3×3 區塊的區域，用文字來表示，可以如 3.12b 所示，表示方式為 MV1、MV2、MV3、MV4、MV5、MV6、MV7、MV8、MV9 等九個移動向量。移動向量若以兩度空間作平滑處理，將會變得很複雜，對整個解交錯系統來說，比較重要的是在移動估計時，找到正確的移動向量，而不是在移動估計找到錯誤的移動向量時，透過移動向量平滑處理來改善。因此，為了減少移動向量平滑處理的複雜度，可以使用簡單一維空間對這九個移動向量，作水平方向平滑處理及與垂直方向的平滑處理，其符號如圖 3.12c 及圖 3.12d 所示。

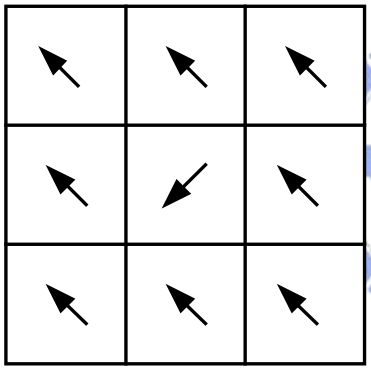


圖 3.12a 移動向量

MV1	MV2	MV3
MV4	MV5	MV6
MV7	MV8	MV9

圖 3.12b 移動向量符號

Xa	Xb	Xc
Xd	Xe	Xf
Xg	Xh	Xi

圖 3.12c X座標移動向量

Ya	Yb	Yc
Yd	Ye	Yf
Yg	Yh	Yi

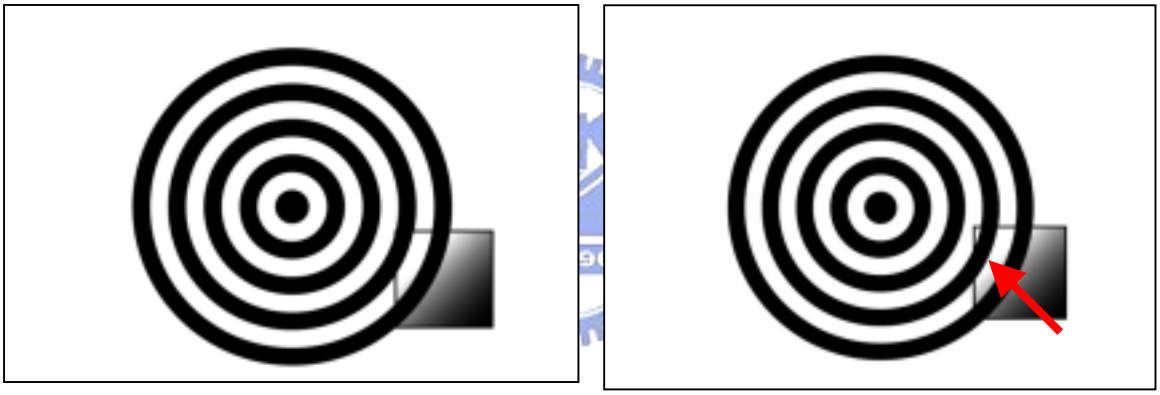
圖 3.12d Y座標移動向量

在這裡設計了三種判斷移動向量錯誤及修正的方式，詳細的說明如下：

方式一：辨別移動向量的方向，將移動向量分成水平方向與垂直方向各別處理，辨別的方式為假設物件移動的範圍超過 3×3 區塊，則中心點(Xe 或 Ye)的移動向量是向右，其它八個點的移動向量是向左。表示中心點(Xe 或 Ye)的移動向量是錯誤的，我們可依據上一張同一點的移動向量，比較得到其它八點最接近移動向量，來代替中心點(Xe 或 Ye)的移動向量。

方式二：和方式一相同的方式去辨別移動向量的方向是否正確，當辨別中心點(Xe 或 Ye)的移動向量是錯誤時，我們可將其它八點的移動向量做平均後，所計算的值，來代替中心點(Xe 或 Ye)的移動向量。

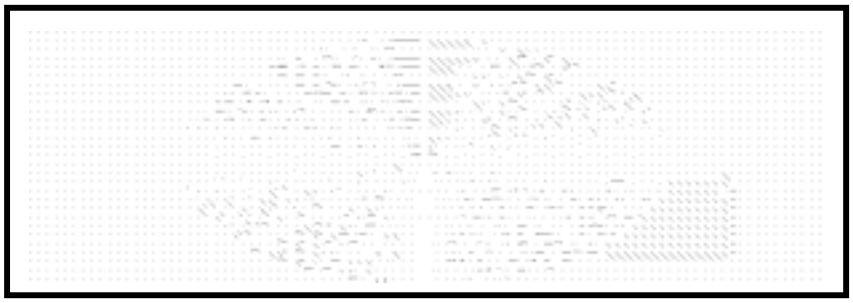
方式三：將 3×3 區塊的每一個移動向量，依水平方向和垂直方向，分別作大小的排序，取得九個移動向量中，排序第五的移動向量，來代替中心點(Xe 或 Ye)的移動向量。



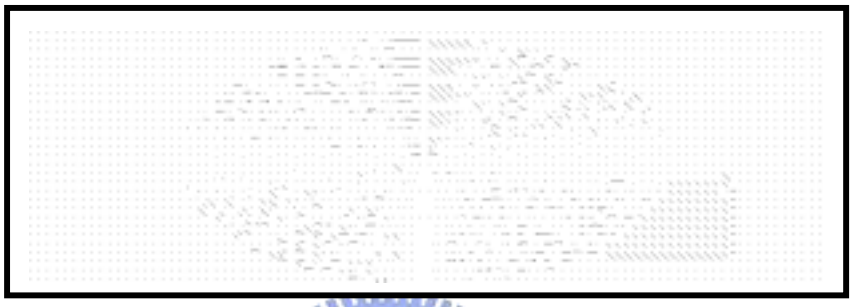
(a)　　(b)

圖 3.13 移動向量平滑處理測試圖 (a). 第一張圖 (b). 第二張圖

圖 3.13 是移動向量的測試圖，由第一張圖和第二張圖的前後關係，可以知道移動向量的方向是由右下角往左上角的方向移動，所以圖 3.14 是不同移動向量平滑處理的結果。圖 3.14(a)是無移動向量平滑處理的結果，圖 3.14(b)是方式一的移動向量平滑處理的結果，這是最簡單的一種方式，硬體成本低是它最大的優點。圖 3.14(c)是方式二的移動向量平滑處理的結果，這種平均處理的方式，比第一種的方式稍為複雜一些，但它的優點是可以避免一些雜訊的干擾。圖 3.14(d)是方式三的移動向量平滑處理的結果，這種經由排序的方式，而取中間值的移動向量，從圖來看是移動向量是最平滑，但相對地成本高是它的缺點，而且要減少作移動估計時，一些雜訊的干擾，否則會有更差的效果。基於成本的考量，我們將使用方式一的移動向量平滑處理，在解交錯的系統架構中。



(a)



(b)

(c)

(d)

圖 3.14　移動向量平滑處理的模擬結果 :(a) 無平滑處理 (b) 方式一的平滑處理 (c) 方式二的平滑處理(d) 方式三的平滑處理。

3.4 移動補償的設計

在一連串的圖場當中，其實只要物體並沒有消失或是轉動，大部份的圖像，即使在目前的圖場沒有移動的物體，也可以在相鄰的圖場找到相似可契合的部份來貼上，這是移動補償解交錯的優點。而要在附近的圖場找出契合的部份來貼上，最常被使用到的是移動補償，移動補償解交錯最早也和移動可適性解交錯一樣，是使用前一張圖場和現在圖場這兩張圖場來做移動估計，再將估計好的移動向量經過移動向量平滑處理後，再作移動補償的處理，讓它重新組合出一張，可以和現在圖場契合的另一張圖場。由圖 3.15 可了解整個移動補償的架構，由於使用了移動補償的技術，使得新增的圖場大部份都是真實的像素，如此也增加了相當多的解析度。

然而，因為移動估計使用不同的圖場，若以 SAD 為基礎的移動估計在這時候也會找不準，因此錯誤的移動向量會造成移動補償錯誤。所以本文中，會增加一個角度偵測線平均(ELA)解交錯演算法，當某一個區域無法以移動補償來成功完成解交錯的時候，便會使用角度偵測線平均解交錯演算法來完成。

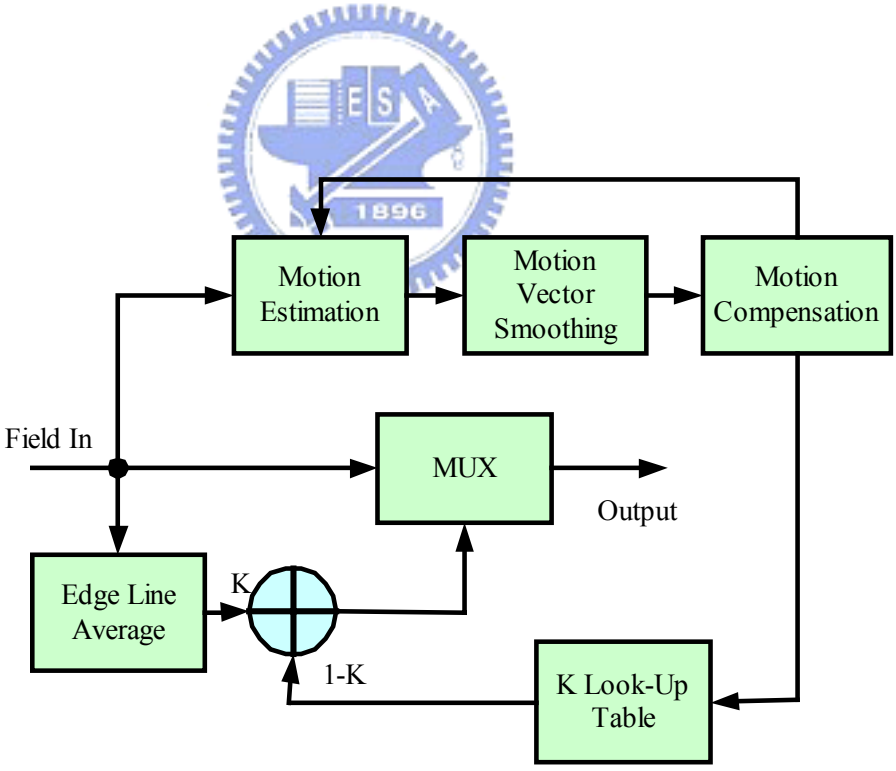


圖 3.15　移動補償的架構

在(式 3.2)中，輸出式$f_{out}(\vec{x}, n)$是循序掃描的資料格式，$f_o(\vec{x}, n)$是目前圖場的原始資料，$f_i(\vec{x}, n)$是目前圖場需要插點的資料。在式(3.2) 中有三種圖場插點的方式，我們先用移動向量是否作錯誤第一次的判斷，若移動向量是正確時，選擇使用(式 3.2)中(C 式) 移動補償解交錯的方式來插點，否則使用非移動補償解交錯的方式來作插點。若選擇非移動補償時，需要再作第二次的判斷，透過移動偵測的機制，若判斷為靜態時，則用(式 3.2)中(A 式)時間平均解交錯的方式來插點，否則為動態，則用(式 3.2)中(B 式)角度偵測線平均(ELA)解交錯的方式來插點。

$$f_{out}(\vec{x}, n) = \begin{cases} f_o(\vec{x}, n), & y \bmod 2 = n \bmod 2 \\ f_i(\vec{x}, n) = [f_o(\vec{x}, n-1) + f_o(\vec{x}, n+1)]/2, & \text{(Static)（A 式）} \\ f_i(\vec{x}, n) = \text{Min（DEAR, DEBQ, DECP, DEDQ, DEEN, DEPM,} & \\ \qquad \text{DEGL, DEHK, DEIJ）, (MV Correct)} & \text{（B 式）} \\ f_i(\vec{x}, n) = K(\vec{x}, n) f_i(\vec{x}, n) + (1-K)(\vec{x}, n) f_o(\vec{x} - d(\vec{x}, \vec{n}), n-1), \text{(MV Incorrect)} & \text{（C 式）} \end{cases}$$

（式 3.2）

$$A = f_o(\vec{x} - \vec{uy}, n) - f_o(\vec{x} - \vec{d}(\vec{x}, n) - \vec{uy}, n-1) \quad \text{(式 3.3)}$$

$$B = f_o(\vec{x} + \vec{uy}, n) - f_o(\vec{x} - \vec{d}(\vec{x}, n) + \vec{uy}, n-1) \quad \text{(式 3.4)}$$

$$C = f_i(\vec{x}, n) - f_o(\vec{x} - \vec{d}(\vec{x}, n), n-1) \quad \text{(式 3.5)}$$

在(式 3.2)中，K 值的參數將影響整個移動補償的效能，所以如何計算 K 值就變得很重要，K 值的範圍在 0~1 之間。首先，將(式 3.3)的 A 和將(式 3.4)的 B 相加等於 D，然後利用 C 與 D 則得到下列的演算法：

演算法如下:

```
if  C≦D
Begin
    if (C=0)
        R=0
    else if ((D-C) ≦2)
```

```
            K=0.5
      else if ((D-C) ≦4)
            K=0.4
      else if ((D-C) ≦6)
            K=0.3
      else if ((D-C) ≦8)
            K=0.2
      else if ((D-C) ≦10)
            K=0.1
      else
            K=0
End

else //if ( D＜C)
Begin
      if (C=0)
            K=0
      else if ((C -D) ≦2)
            K=0.5
      else if ((D-C) ≦4)
            K=0.6
      else if ((D-C) ≦6)
            K=0.7
      else if ((D-C) ≦8)
            K=0.8
      else if ((D-C) ≦10)
            K=0.9
      else
            K=1
End
```

3.5 移動偵測的設計

由於移動補償仍有部份不能解決的問題，像是超越搜尋範圍的移動、轉動或是縮放。所以當某一個區域，無法以移動補償來成功地完成解交錯的時候，便會使用非移動補償解交錯來完成，針對無法用移動補償的區域，將利用移動偵測（Motion Detection）的機制，來辨別每一像素。若移動偵測的結果是不動時，便使用時間平均解交錯，否則就採用角度偵測線平均(ELA)解交錯的演算法，因此移動偵測的準確性將變得很重要。

一般用來移動偵測的方法，是將前一張圖場，及目前的圖場的內容儲存在外部的記憶體裡。在讀到某一像素的資料時，就直接輸入兩個像素的資料，然後要經過一個加法器對他們做相減，就可以得到該位置的差異。再透過位準（Threshold）的比對，對於差異較大的值就判定為在移動的區域，並將辨別符號顯示為“1”；差異不大的值，就是在判定為靜止的區域，辨別符號顯示為“0”。

但這樣的做法，有一個缺陷：前一張圖場和目前的圖場在拍攝時，並非具有相同的真實空間位置。直接拿來作處理的話，會有空間上的差異，也就是他們對應到的真實空間會相差一行。另外，多半都有經過人工處理的部分，畫面上往往都會有字幕或是旁白的標示出現。如果我們忽略他們，直接全部相減，將會把字幕當成也在運動的部份，而把整個字幕變模糊。

因此在這裡，我們所提出來的演算法，是將目前的圖場和前一張圖場都存起來，然後做後一張圖場和前一張圖場的差異。這樣的好處是，字幕以及對應到真實空間的位置都會是恆定的，就不會造成每張畫面都有不少誤判的部份了。但是缺點是前一張圖場，及後一張圖場在時間軸上會有較大的落差，我們要用這兩個圖場的資訊，來判定目前的圖場，在移動的地方會不妥。因此我們在前端做了小處理，增加低通濾波器及高通濾波器，可以讓畫面中的雜點都不見了、消除了許多外圍的雜點。圖 3.16 是移動偵測的方塊，圖 3.17 是移動偵測的流程圖，由這兩個圖可以很清楚地了解移動偵測的作法及設計的概念。

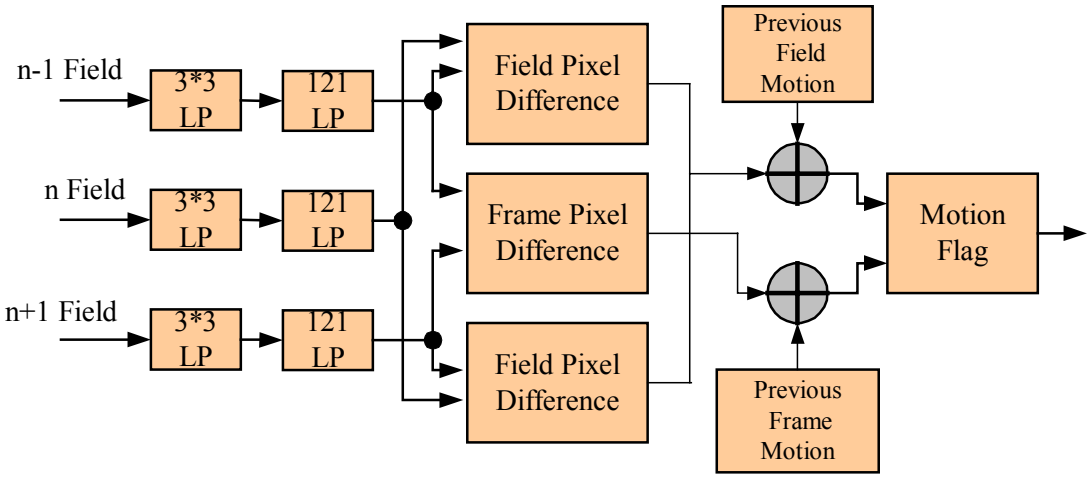


圖3.16 移動偵測區塊圖

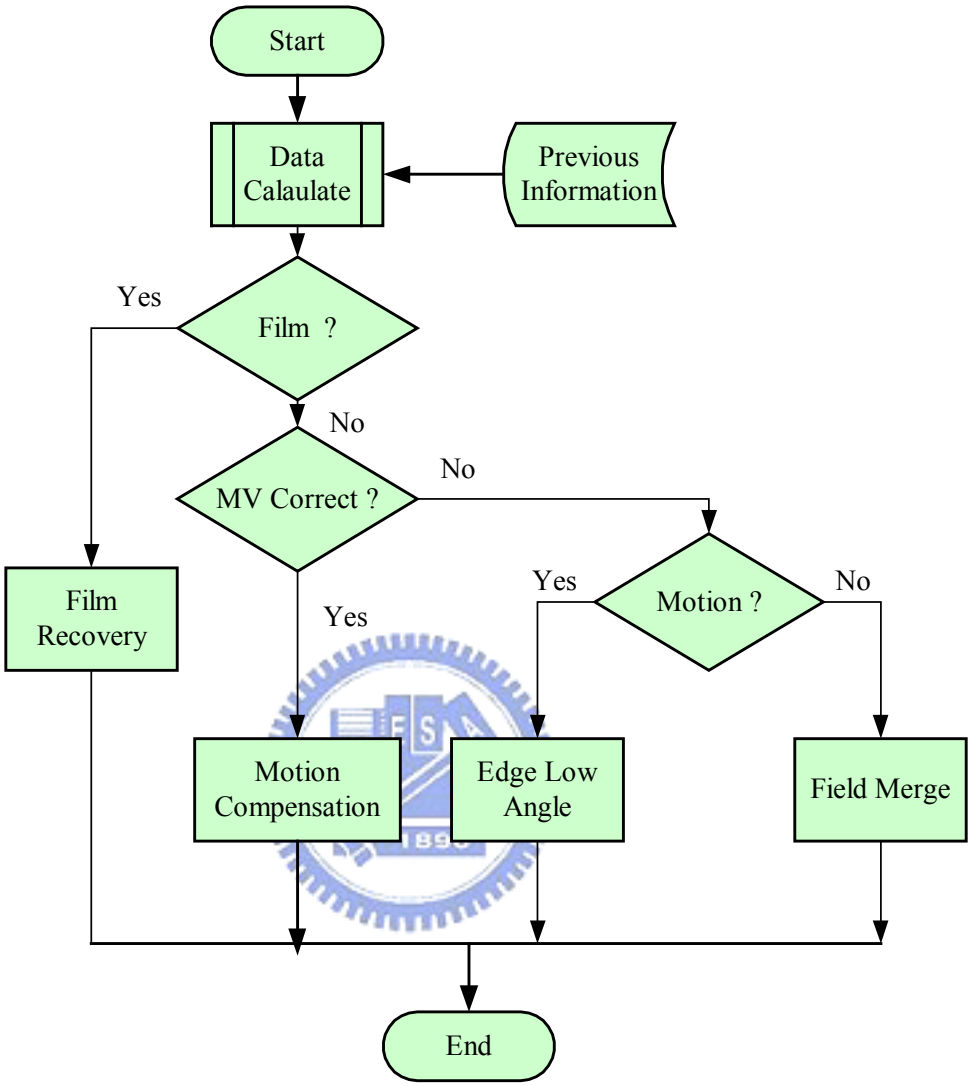


圖3.17 移動偵測流程圖

第四章 電影膠卷偵測與影像加強的演算法設計

4.1 電影膠卷模式的介紹

在一般的倍頻掃描，只是將掃描線加倍。因此，若將 DVD 影片或高化質數位電視的影片，只將兩個圖框做循序處理，就會有許多的圖框完全一樣，卻又要重疊起來，看起來就有跳格及色彩影像錯誤。為了消弭這種情形，就需影片還原（Film Recovery）的功能。此功能，能依據影片來源格式自動判別，還原成原來 2-3-2-3 格式，再做循序成每秒 24 個畫面。如此一來，就可以看到與電影院同為平順流暢的影像畫面。當然這功能對電影拍攝方式的影片才有作用，而且處理晶片也要即時的判斷才行，因為有些影片在製作時就會先行一併處理，沒判斷錯的話，可是會有反效果出現。

原始的電影膠卷，為每秒拍攝 24 個畫面。影片每秒記錄了 24 個循序掃描的畫面，為了轉換成一般電視能觀賞的每秒 60 個圖框的交錯掃描，就必須將影片做 2-3-2-3 的轉換。也就是說第一格的膠卷轉換成 2 個圖框，第二格膠卷轉換成 3 個圖框，第三格膠卷轉換成 2 個圖框，此 2-3-2-3 類推。如圖 4.1 所示，轉成一般的電視都可以接受的交錯掃描畫面。

電影和電視在播放時每秒有 12 格的差距，不管是錄影或播放，都必須進行轉換。此時轉換的方式是第 1 格轉換 2 次，第 2 格轉換 3 次，第 3 格轉換 2 次，第 4 格轉換 3 次，以確保每秒鐘 60 掃描場的影像，這是傳統「2：3 Pull down」技術。3 : 2 Pull down 電影格式補差技術可完全還原電影原始訊號，避免交錯掃描所產生的閃爍、鋸齒現象，讓電影畫面更清晰穩定。

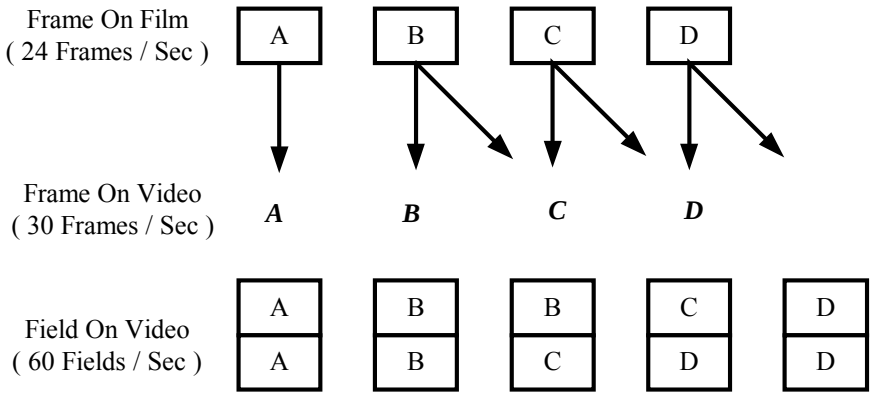


圖4.1 3:2 Pull Down 資料格式

4.2 電影膠卷偵測的架構設計

在電視系統中我們可以看到有兩種規格的影像，一種是每秒鐘播放 60 張圖場的交錯式影像；另一種是每秒鐘播放 24 張圖框的漸進式影像。一般的電影都是以 24 圖框/秒來拍攝的，但是要存放到 DVD 或在電視上要播出時，勢必要先轉換成 60 張圖場。因此一般的處理方法，就是把一張圖框拆成三張圖場，下一張圖框拆成兩張圖場。這樣不斷地拆下去，就可以生成每秒鐘播放 60 張圖場的影像了。

這種 3:2 Pull down 的回復並不困難，只要我們能找到到正確的圖場位置，把正確的幾張圖場抓出來，拼成原來的圖框就可以回復了。真正主要麻煩的地方在於我們要怎麼去「偵測」這種屬於 3:2 Pull down 的畫面。在此我們也提出一個方法，就是運用了我們前面敘述解交錯架構的移動偵測，圖 4.2 是影片偵測的方塊圖。如果我們對連續的十張圖場一張張來作偵測，透過比對相鄰的圖框，累積每一像素的差異值，若大於所設定的參考位準，表示移動符號設成“1”，否則設成“0”，連續累積順序是否為“01111”，若比對相同，表示判定這一段是 3：2 Pull down 的影像，因此也就可以進行接續 3：2 Pull down 的回復動作了。

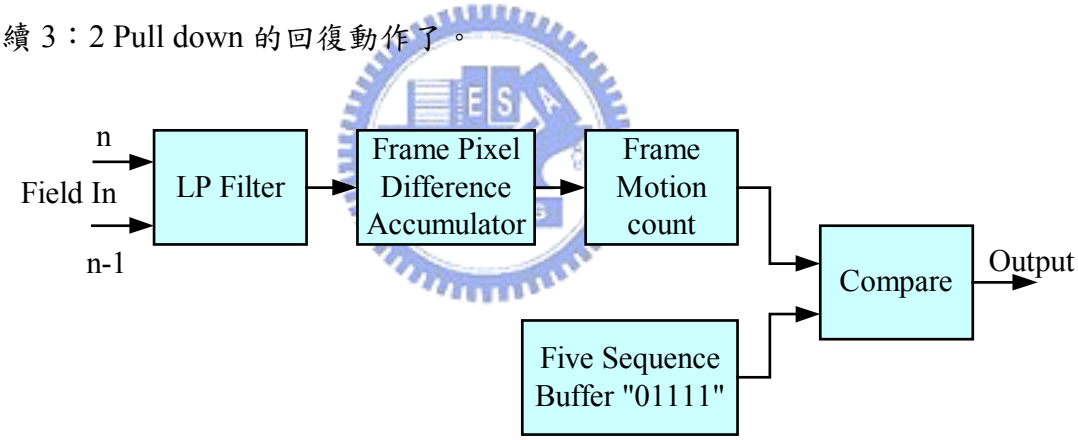


圖4.2 電影膠卷偵測方塊圖

在圖 4.3 和圖 4.4 是 3 : 2 Pull Down 影片的測試結果，我們比較解交錯的架構中有設計影片偵測及還原機制，與沒有影片偵測及還原機制。結果發現在圖 4.3(a) 空間掃描線垂直平均解交錯的結果會使得看台區域的畫面有嚴重的鋸齒狀，圖 4.3 (b) 時間平均解交錯的方式，會有些畫面清晰，有些畫面有更嚴重的鋸齒狀。在圖 4.4(a) 空間掃描線垂直平均解交錯的結果，會使得的六條水平線有時變粗，有時完全不存在，在圖 4.4(b) 時間平均解交錯的結果，會使得的六條水平線有時完全不存在，若能像圖 4.3(c)和圖 4.4(c)正確偵測到現在的畫面是 3 : 2 Pull Down 的資料格式，則在作影片還原後，畫面非常清析，且不會在畫面出現鋸齒狀。反之，則無論是使用非移動解交錯或移動解交錯的演算法，都會讓畫面看起來很模糊。

第一種 電影膠卷 Speedway 圖片測試

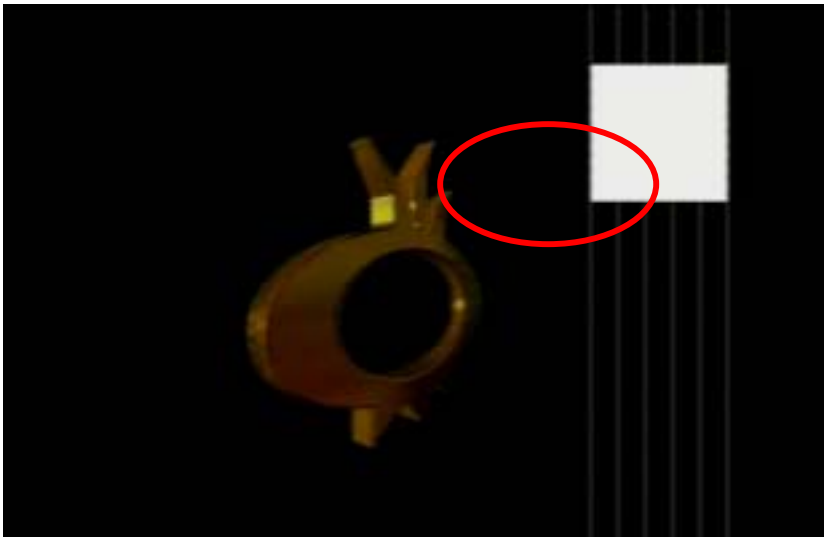
(a) 空間掃描線垂直平均解交錯

(b) 時間平均解交錯

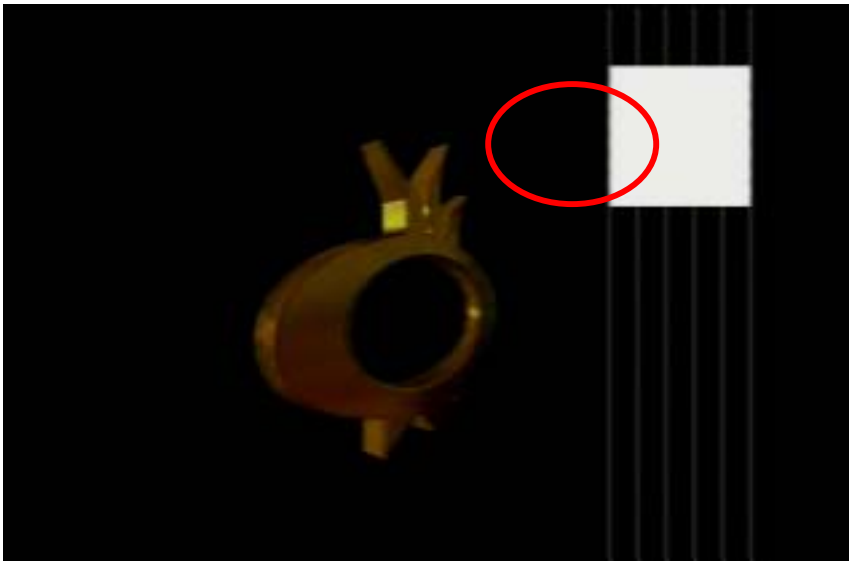
(c) 電影膠卷復原

圖 4.3　3 : 2 Pull Down Speedway 測試結果 :
(a) 空間掃描線垂直平均解交錯　(b) 時間平均解交錯　(c) 電影膠卷復原

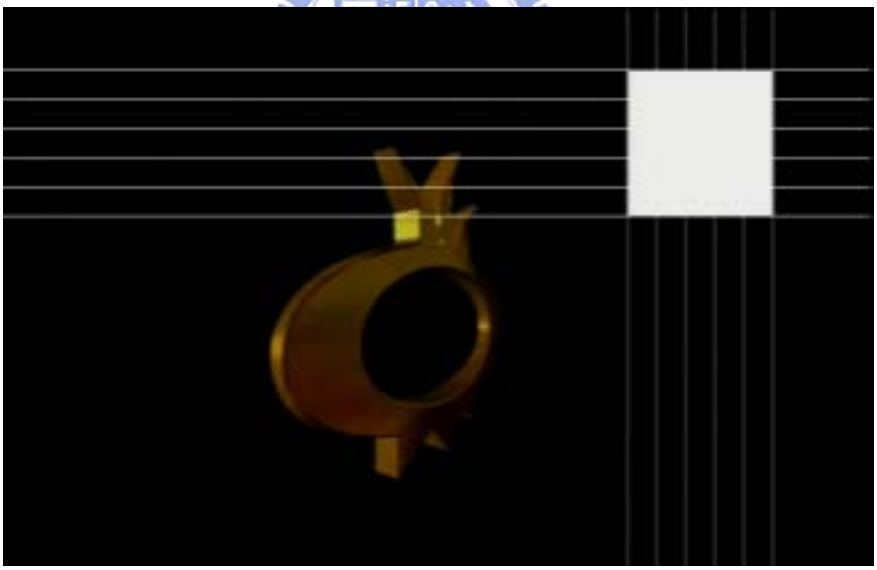
第二種 Music 電影膠卷圖片測試



(a) 空間掃描線垂直平均解交錯



(b) 時間平均解交錯



(c) 電影膠卷復原

圖 4.4　3 : 2 Pull Down Music 測試結果 :
(a) 空間掃描線垂直平均解交錯　(b) 時間平均解交錯　(c) 電影膠卷復原

4.3 前置與後置處理器的設計

前置處理器主要是針對移動偵測做低通濾波或高通濾波，以減少雜訊的干擾。後置處理器主要是對解交錯後的影像作補償，使視覺上產生較佳的效果。圖 4.5 是後置處理器的方塊圖。我們將提出數種提升影像畫面品質的方法，這些方法對解交錯掃描格式轉換後的影像提升了畫面品質。處理過的影像畫面並不會更接近原始畫面，但可以讓經過調整後的影像品質達到最好的視覺效果。

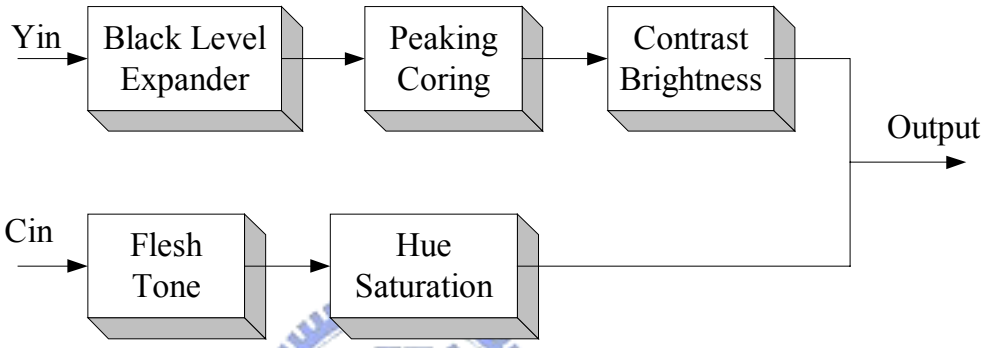


圖4.5 後置處理器方塊圖

4.3.1 黑階位準延伸設計

在電視系統中，有時會感覺畫面黑的部份不夠黑；白色的部份不夠白。可能是訊號傳送當中，或者在做色彩轉換處理時，忽略低位元的部份，造成色彩動態範圍不足。利用如圖 4.6 的設計[13]，就可以達到黑階位準延伸的效果。在圖 4.6 中，我們要先設定黑階位準（Black Level, 簡稱 BL）的值，經由偵測器（Detector）去辨別輸入的值是否要作黑階位準延伸。若輸入的值小於 BL，則做黑階位準延伸的處理；否則不做黑階位準延伸的處理，其中增益設定越大，會使曲線越傾斜。

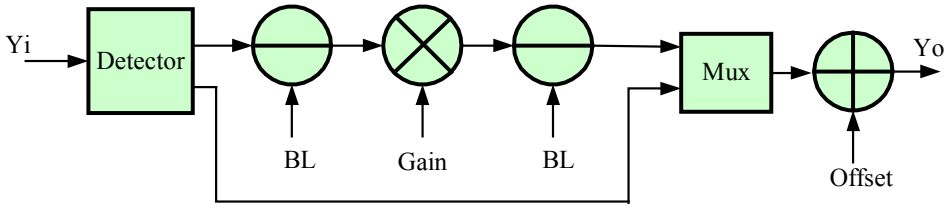


圖4.6　黑階延伸方塊圖

黑階位準延伸的演算法如下：

if Input Y[i] < BL

　　Yo[n] = (BL – Gain * (BL - Yi [n])) + Offset

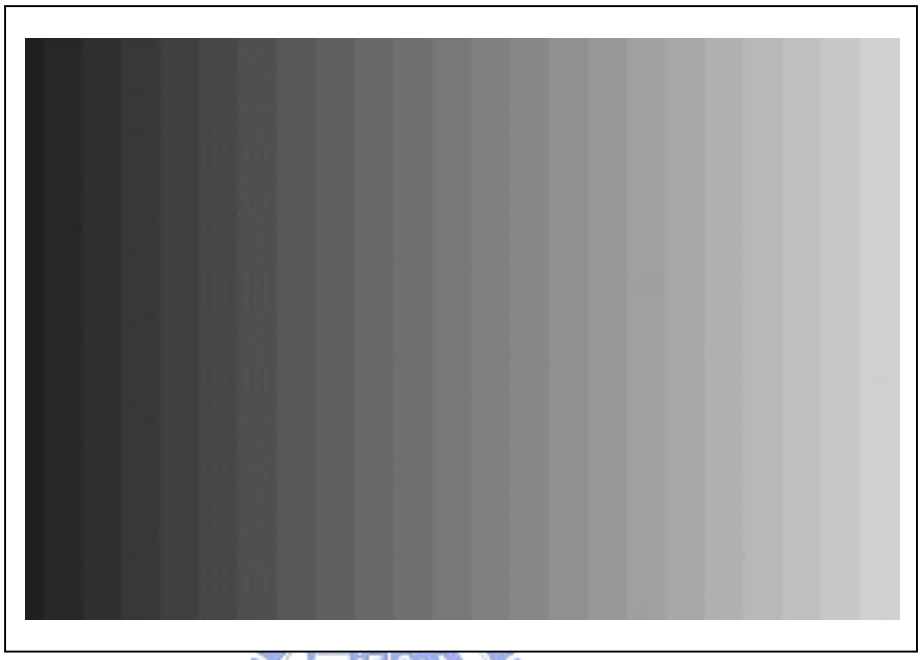
else

　　Yo[n] = Yi [n] + Offset

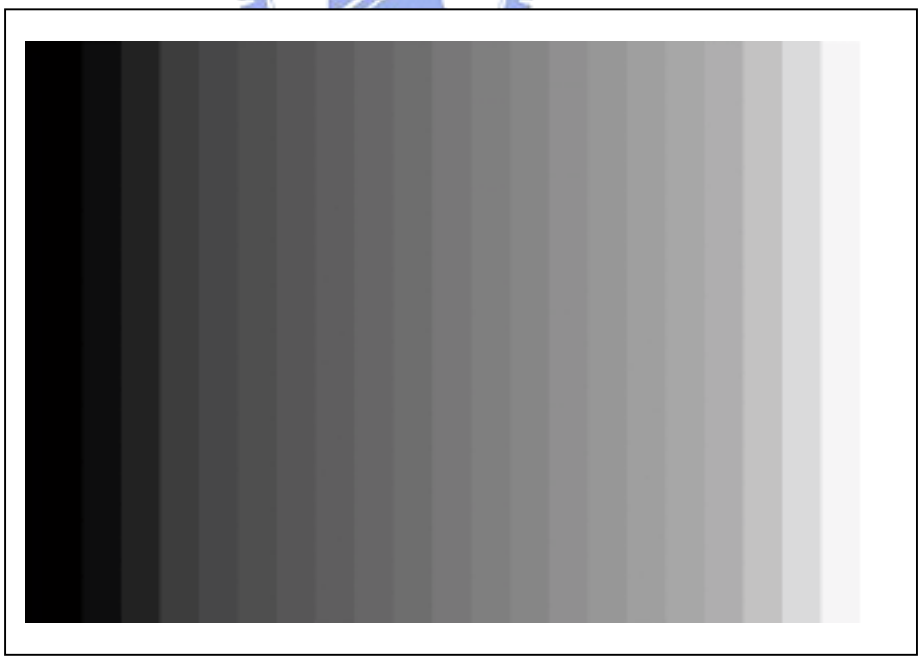
參數	參數範圍	位元數	設定值
Gain	0/64 ~ 255/64	8 位元無號數	128
BL	0 ~ 255	8 位元無號數	40
Offset	-128 ~ +128	8 位元有號數	-40

表 4：黑階延伸設計參數表

圖 4.7a 是一張灰階的原始圖片，從圖中可知道灰階的動態範圍不夠，而造成黑不夠黑，但經由黑階位準延伸調整之後，則整個灰階的動態範圍變寬，如圖 4.7b 所示。



(a) 原始圖片



(b) 黑階位準延伸調整後圖片

圖 4.7 黑階位準延伸測試結果 :(a) 原始圖片 (b) 黑階位準延伸調整後圖片

4.3.2 邊緣加強高頻雜訊濾除設計

在一些畫面中，會覺得有些物件很模糊，透過高通濾波器（High Pass Filtering）的處理，如圖 4.8 的硬體設計[13]，就能達到邊緣加強及濾除一些雜訊干擾。它是五個 Tap 高通濾波器，係數的排列順序為 C2 、C1、 C0、 C1、 C2，這些係數可以是正數，也可以是負數。但總合起來的係數應該為 0，因為要作邊緣加強，C0 最好設定是正數，方塊圖中的 Coring 是準位控制器，依照設定的位準（LV），決定輸入是否作邊緣加強。

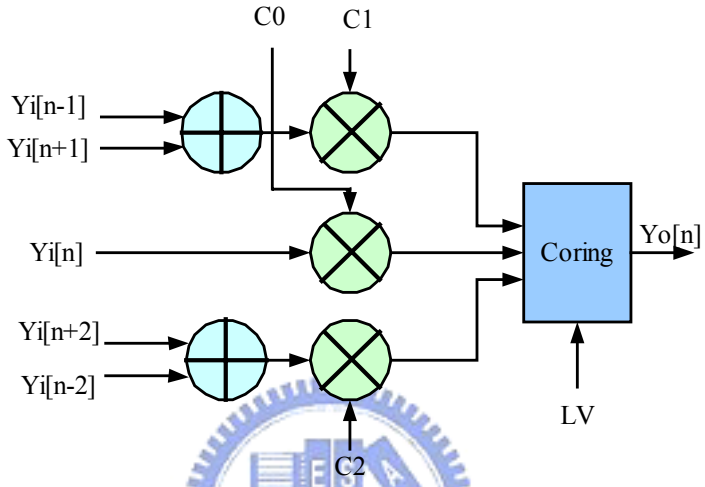


圖4.8 邊緣加強高頻雜訊濾波器方塊圖

邊緣加強高頻雜訊濾除的演算法如下：

$Y1[n] = C0*Yi[n] + C1*(Yi[n+1] + Yi[n-1]) + C2*(Yi[n+2] + Yi[n-2])$

if $|Y1[n]| \leqq LV$

$Yo[n] = Y[i]$

else // ($|Y1[n])| > LV$)

$Yo[n] = Y1[n]$

參數	參數範圍	位元數	設定值
C0	+127/128 ~ -128/128	8 位元有號數	0.5
C1	+127/128 ~ -128/128	8 位元有號數	–0.125
C2	+127/128 ~ -128/128	8 位元有號數	–0.125
LV	0 ~ 31	5 位元有號數	31

表 5：邊緣加強高頻雜訊濾波器設計參數表

圖 4.9a 是一張藍天白雲加上椰子樹的原始圖片，從圖中可知道椰子葉的地方很模糊，而使得畫面看起來很朦朧的感覺，但經由邊緣加強高頻雜訊濾除調整之後，則整個畫面就變清晰很多，如圖 4.9b 所示。

(a) 原始圖片

(b) 邊緣加強高頻雜訊濾除後圖片

圖 4.9 邊緣加強高頻雜訊濾除測試結果 :(a) 原始圖片 (b) 邊緣加強高頻雜訊濾除調整後圖片

4.3.3 對比度和亮度調整的設計

對比度和亮度調整是一般最常見的影像處理功能，透過如圖 4.10 中一個乘法器及加法器[13]，就能達到對比和亮度的控制。對比度調整相當於乘以增異，亮度調整相當於加一個位移（Offset），限制器（Limiter）主要保護輸入的值在 0～255。

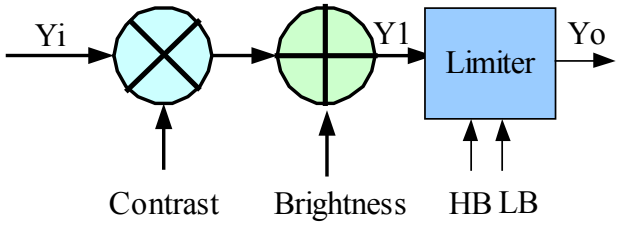


圖4.10 對比度與亮度調整方塊圖

對比度和亮度調整演算法如下：

Y1[n] = Contrast * Yi [n] + Brightness

if (Y1[n] ≧ HB)

Yo[n] = HB

else if (Y1[n] ≦ LB)

Yo[n] = LB

else (LB< Y1[n] <HB)

Yo[n] = Y1[n]

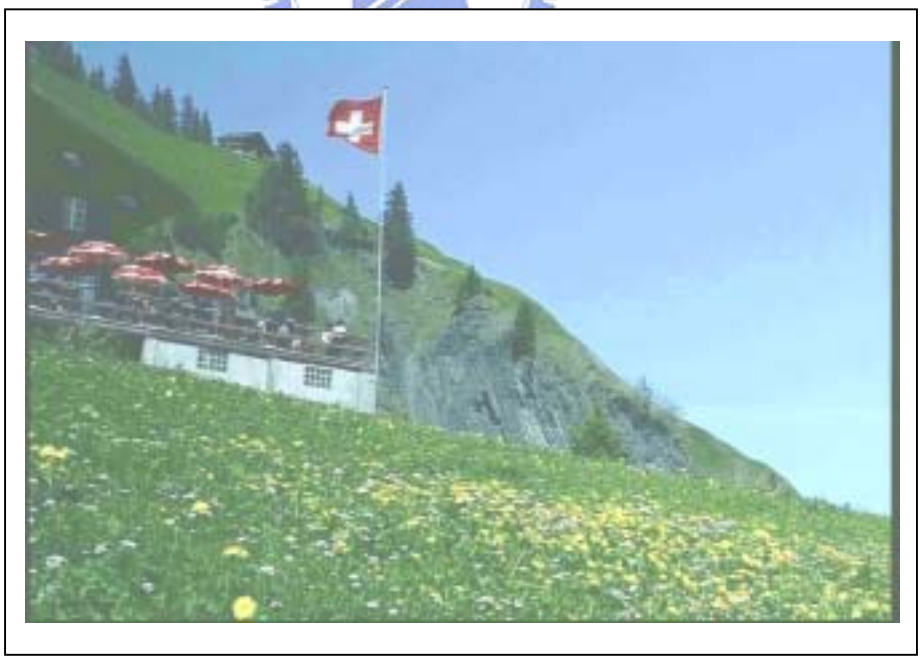
（式 4.3）

參數	參數範圍	位元數	設定值
Contrast	0 ~ 255/128	8 位元無號數	192
Brightness	-128 ~ +127	8 位元有號數	30
LB	0 ~ 255	8 位元無號數	0
HB	0 ~ 255	8 位元無號數	255

表 6 對比度和亮度設計參數表

圖 4.11 (a)是一張美麗風景的原始圖片，從圖中可知道畫面黑白的對比度不夠大，亮度也很暗，而使得畫面看起來有陰天的感覺，但經由對比度和亮度調整調整之後，則整個畫面就變得很明亮，如圖 4.11 (b)所示。

(a) 原始圖片



(b) 對比度和亮度調整後圖片

圖 4.11 對比度和亮度測試結果 :(a) 原始圖片 (b) 對比度和亮度調整後圖片

4.3.4 膚色補償的設計

在電視或是影片中，最重要的就是人物，人物中最強調的，就是臉的膚色，膚色不好會讓人覺得無生氣，所以針對膚色作加強是有必要的，架構如圖 4.13 所示[13]。由於人的眼睛對於膚色的感覺特別靈敏，當電視螢幕所呈現的膚色與個人主觀的認定不同時，例如東方人膚色過白，西方人膚色過黃等，都會讓觀看者無法接受，所以必需有膚色補償的設計，讓使用者調整自己喜歡的膚色。圖 4.12 是臉膚色的示意圖，其中黑色虛線的扇形區域，是需要增強的地方，但硬體上要話劃分扇形視窗是很困難，所以使用四個矩形的視窗來代替。這個架構有四個視窗可以設定，分別是 UHx、ULx、VHx、VLx，它可設定不同的大小，及不同的增益，視窗有優先權的分別，視窗 1 的優先權最高，依次視窗 2、視窗 3、視窗 4。四個視窗的設計，依不同的增益可表現層次感，當然越多視窗的設計會更好，但相對地需要更多的硬體成本。

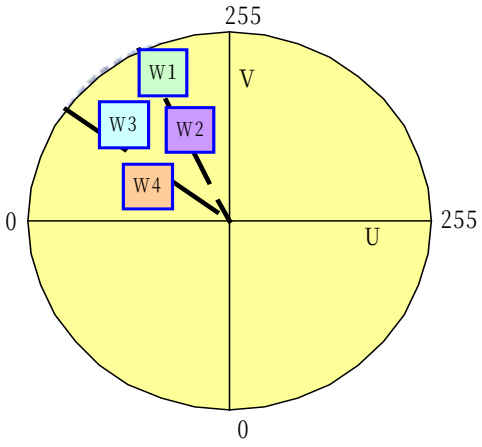


圖 4.12 臉的膚色示意圖

膚色補償的演算法如下

```
for(x =0 ; x < 4 ; x++)
Begin
    if (UL[x]≦Uin[n] ≦Uh[x])
        Uo[n] = UG[x] * Uin[n]
    If (VL[x]≦Vin[n] ≦VH[x])
        Vo[n] = VG[x] * Vin[n]
End
```

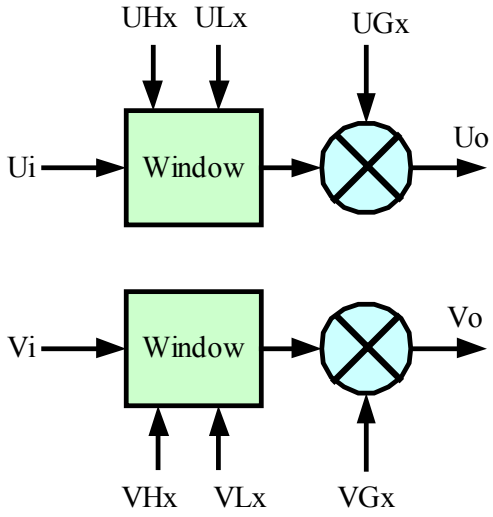


圖4.13 膚色補償方塊圖

參數	參數範圍	位元數	設定值
UH[x]	-128 ~ +127	8 位元有號數	100
UL[x]	-128 ~ +127	8 位元有號數	64
VH[x]	-128 ~ +127	8 位元有號數	-80
VL[x]	-128 ~ +127	8 位元有號數	-20
UG[x]	0 ~ 127/64	7 位元有號數	127
VG[x]	0 ~ 127/64	7 位元有號數	127

表 7：膚色補償設計參數表

圖 4.14 (a)是一張美女的原始圖片，從圖中可知道美女臉部的膚色太白，而顯得比較沒有精神，但經由膚色補償調整之後，則美女的臉色變得很紅潤，如圖 4.14 (b)所示。

(a) 原始圖片

(b) 膚色補償後圖片

圖 4.14 膚色補償測試結果 :(a) 原始圖片 (b) 膚色補償調整後圖片

4.3.5 色度與飽合度調整的設計

對色度（Hue）位於座標軸的位置[13]，以座標軸為中心旋轉，可以改變不同的顏色，也就是改變顏色的色度。同時對色度乘以一個增益，可以改變顏色飽和度（Saturation）。飽和度較大，會使顏色較為飽滿；飽和度較小，則呈現較透明的感覺。如果再對色度的準位作調整，可以改變顏色的深淺，色度與飽合度會影響影像的色彩，硬體的架構如圖 4.15 所示：

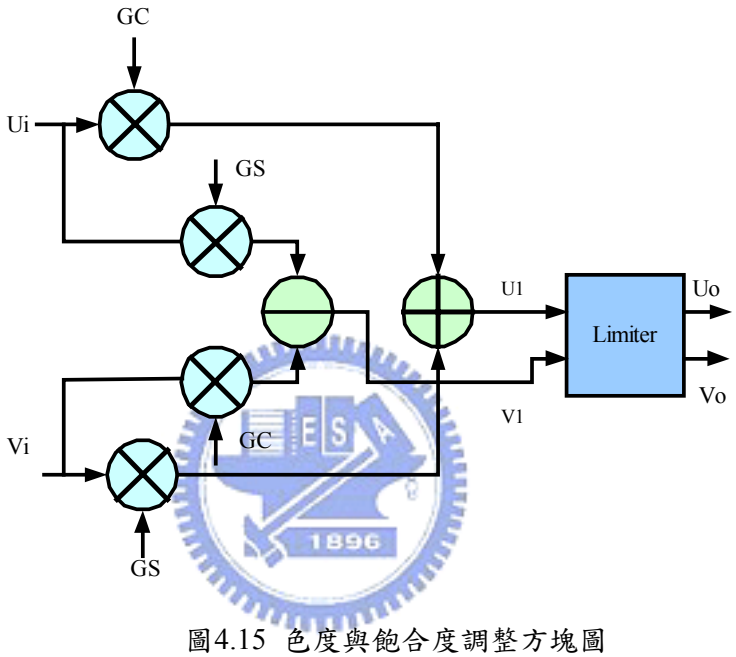


圖4.15 色度與飽合度調整方塊圖

色度與飽合度調整的演算法如下

U1 [n] = GC * Ui[n] + GS * Vi[n]

V1 [n] = GC * Vi[n] – GS * Ui[n]

if (| U1[n] | ≧BND)

    Uo[n] = Sign(U1[n]) * BND

else // (| U1[n] | <BND)

    Uo[n] = U1[n]

if (| V1[n] | ≧BND)

    Vo[n] = Sign(V1[n]) * BND

    else // (| V1[n] | <BND)

    Vo[n] = V1[n]

參數	參數範圍	位元數	設定值
GC	-2～+2	12 位元有號數	0.5
GS	-2～+2	12 位元有號數	0.86

表 8：色度和飽合度設計參數表

GC 和 GS 的參數設定方式是互相有關聯，假如我們要將色度旋轉 60 度，GS 的參數等於 sin60 度的值 0.86，而 GC 的參數等於 cos60 度的值 0.5。

圖 4.16 (a)是一張美麗風景的原始圖片，從圖中可知道紅色和黃色的鬱金香顏色不夠飽和，但經由色度與飽合度調整之後，則紅色和黃色鬱金香的花色就變得比較鮮豔亮麗，如圖 4.16 (b)所示。

(a) 原始圖片

(b) 色度和飽合度調整後圖片

圖 4.16 色度和飽合度測試結果 :(a) 原始圖片 (b) 色度和飽合度調整後圖片

第五章 模擬的結果分析

5.1 解交錯演算法的模擬方式

在這一節中，我們將整合式解交錯演算法，利用電腦設計一套模擬的軟體上，用以驗證我們所提出來的演算法，及系統架構的可行性與輸出品質。我們會先針對各個演算作模擬，並檢討各個演算法的輸出結果，最後依據第三章所提出來的整合式解交錯演算法作整個系統的模擬，並對輸出結果作比較。我們可以比較兩者間的峯值訊號雜訊比(PSNR)及錯誤平方平均值(MSE)，如(式 5.1)及(式 5.2)所示。

$$PSNR = 10 \log_{10} \left(\frac{255}{MSE} \right)^2 \quad （式 5.1）$$

$$MSE = \frac{1}{N.\ X.\ Y} \sum_{n=0}^{N-1} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \left| \hat{F}(x, y, n) - F(x, y, n) \right|^2 \quad （式 5.2）$$

在本文解交錯的演算法，採用兩種的模擬方式，圖 5.1 是第一種模擬的方式，我們先取得循序掃描方式拍攝的自然圖片，或者可以利用圖片編輯的工具，製作循序掃描格式的圖片，假如圖片的寬和高的像素為 1920*1080 或者其它大於 720*480 的尺寸，因為觀察與分析並不方便，我們可以選取特定區域，大小為 720*480 的像素，連續取 20 張，然後依照偶數圖場、奇數圖場、偶數圖場、奇數圖場的排列順序，將它作成交錯掃描的格式。當由循序掃描格式(720*480) 轉換成偶數圖場時，需將奇數線的資料移除，偶數圖場就變成 720*240，同理，循序掃描格式(720*480)轉換成奇數圖場時，將偶數線的資料移除，奇數圖場就變成 720*240，由這樣的作法，可得到交錯掃描的圖片。將這 20 張交錯掃描的圖片，當作解交錯系統的輸入，可得到循序掃描 (720*480)的圖片輸出，再將這 20 張的輸出和原始的循序掃描圖片作 MSE 的計算，當從 MSE 的數值越大時，表示解交錯的效能越差，另外也可以從視覺上來判斷。

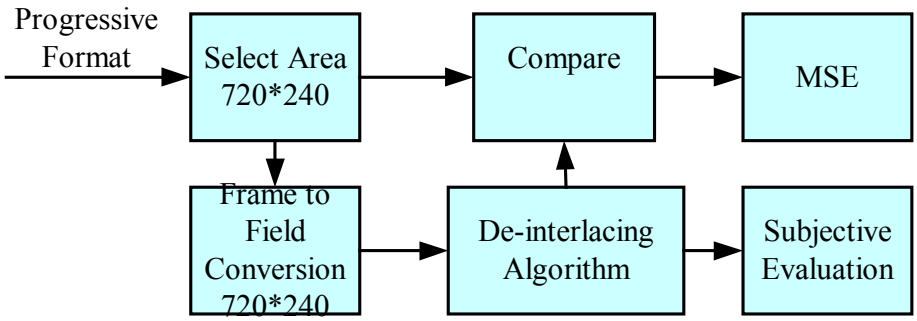


圖5.1 解交錯演算法模擬方法1

第二種解交錯演算法，將使用如圖 5.2 所示的模擬方式，和第一種模擬方式的最大不同點是模擬方法 2 的輸入來源是 DVD 影片，它是交錯掃描的資料格式，經過解交錯演算法後，就變成循序掃描 (720*480)的圖片輸出，因為 DVD 影像拍攝的方式是用交錯掃描的資料格式，所以無法計算 MSE 的值，但是可以透過視覺的感受，來判斷解交錯後的影像是否清晰，而且這種方式也是工業界最常被使用。

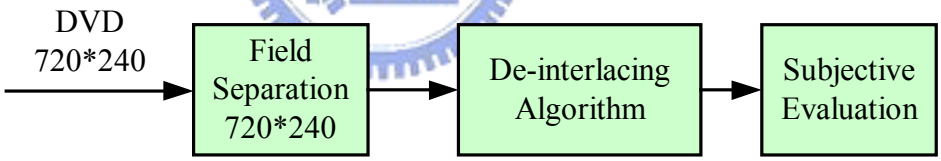


圖5.2 解交錯演算法模擬方法2

圖 5.3 是依照整合式解交錯系統的程式流程圖。圖 5.4 是電腦模擬軟體所設計的所有頁數，模擬軟體主要的功能分為六頁，茲說明如下:第一頁是移動估計的方式和移動估計的參數設定，移動估計如第三章所提的演算法共分五種方式，可配合不同的解交錯演算法，移動估計的參數設定，它可以選擇比對方塊大小及移動估計的搜尋範圍，依據不同的參數設定，有不同的結果。第二頁是移動向量平滑處理的方式和 3:2 Pull Down 的開關，移動向量平滑處理共分四種，它可以輸出移動向量到檔案中，也可以畫出移動向量到 BMP 的圖檔。第三頁是前端處理的功能設定，它包含雜訊濾波器的選擇、輸入影像的資料格式選擇、移動估計的資料格選擇及測試模式的開關等。第四頁是後端處理

的選擇，影像加強的功能可以單獨選擇，也可以混合設定。第五頁是各種解交錯演算法的選擇和搭配各種影像輸入，解交錯主要分非移動補償和移動補償兩大類。第六頁是各種影像輸入前處理設定。

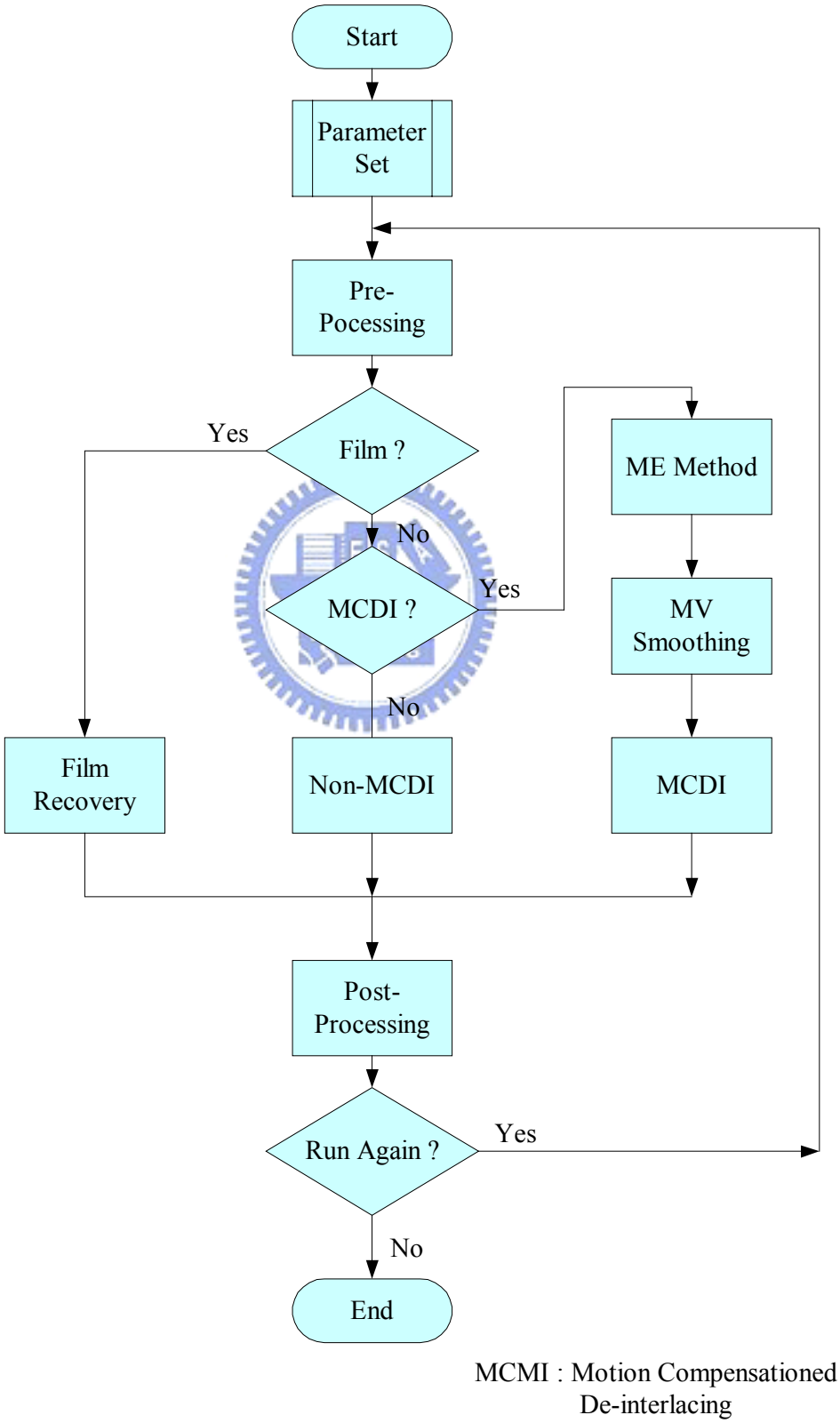


圖 5.3　模擬程式的流程圖

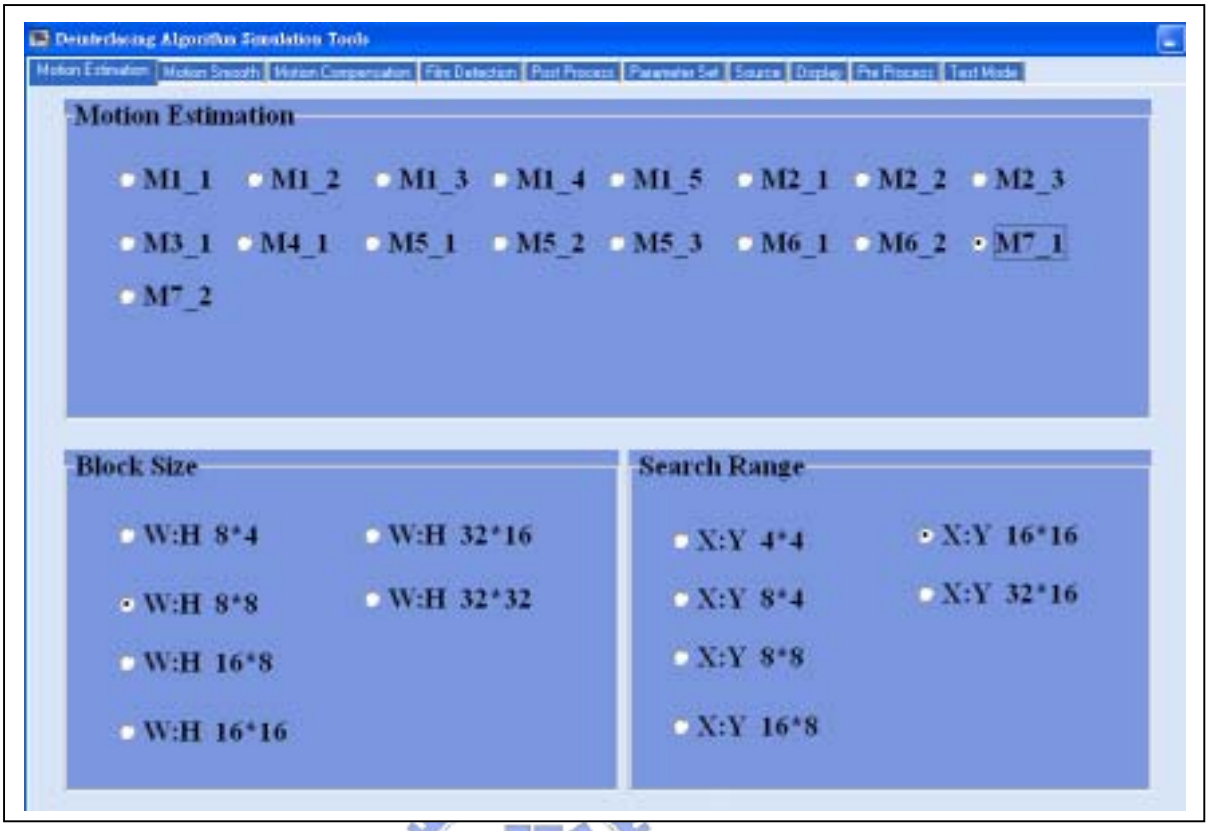


圖 5.4(a)　模擬程式第一頁

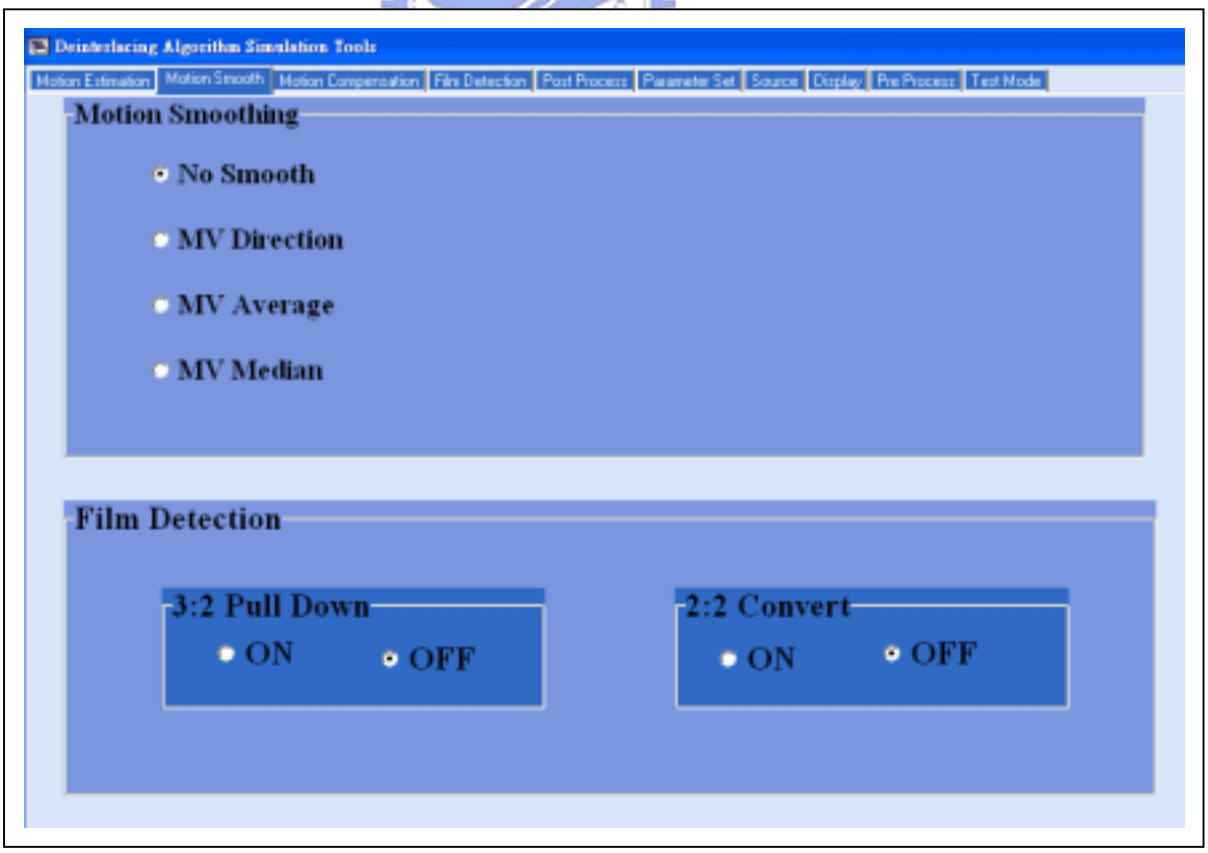


圖 5.4(b)　模擬程式第二頁

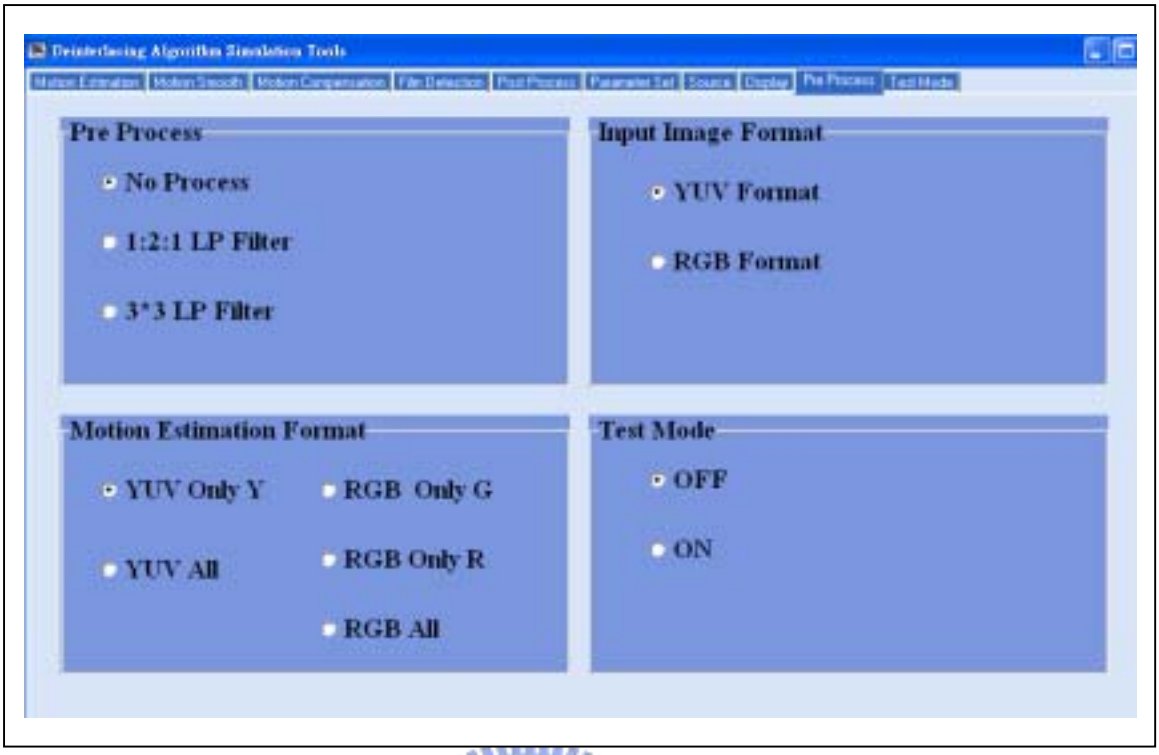


圖 5.4(c)　模擬程式第三頁

圖 5.4(d)　模擬程式第四頁

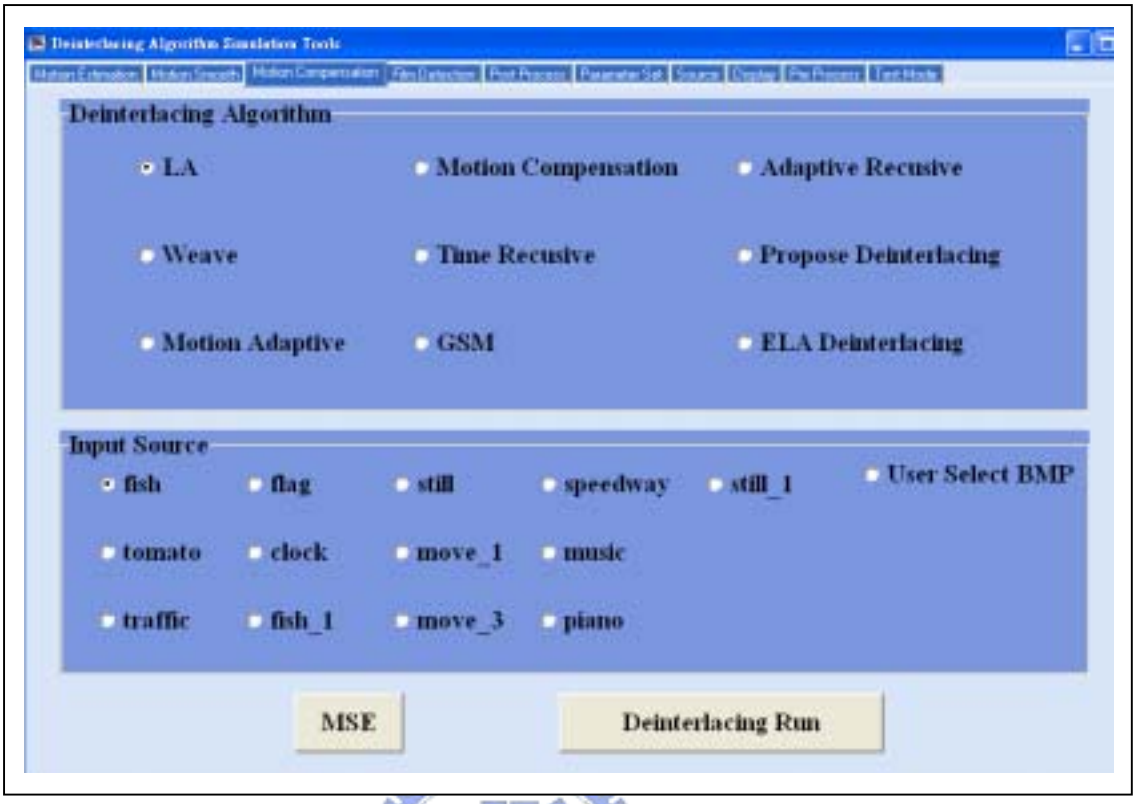


圖 5.4(e)　模擬程式第五頁

圖 5.4(f)　模擬程式第六頁

5.2 解交錯演算法的模擬結果

5.2.1 第一種解交錯模擬方式的模擬結果

解交錯演算法	Test1 (動態)	Test2 (靜態)	平均值
A	2.2863	0.42486	1.35558
B	75.8982	0	37.9491
C	1.5323	0.1226	0.82745
D	1.3131	0.2934	0.80325
E	1.3377	0.2934	0.81555
F	0.4105	0.0962	0.25335

表 9　各種解交錯演算法 MSE 的比較

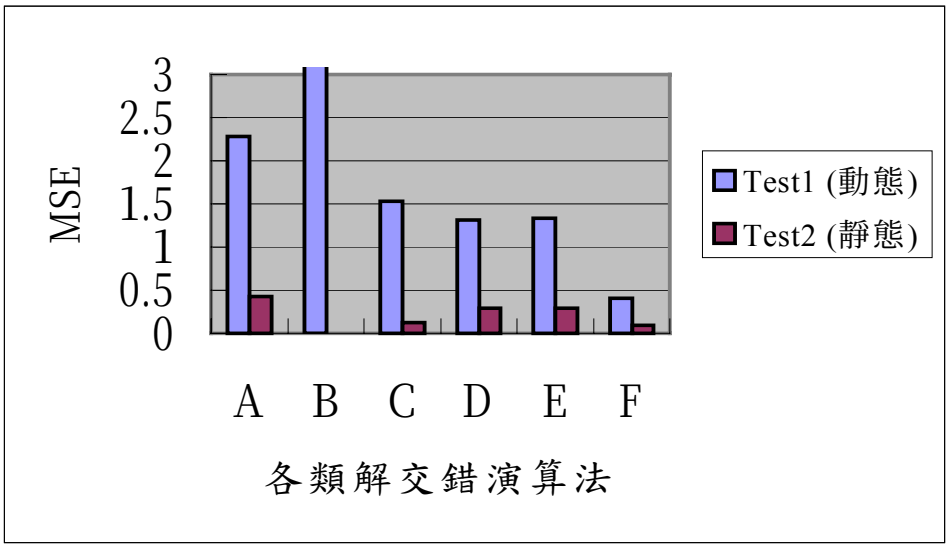


圖 5.5　各類解交錯演算法直方圖

A. 空間掃描線垂直平均解交錯(Line Averaging De-interlacing，簡稱 LA)演算法

B. 時間平均解交錯(Temporal Averaging De-interlacing，簡稱 Weave)演算法

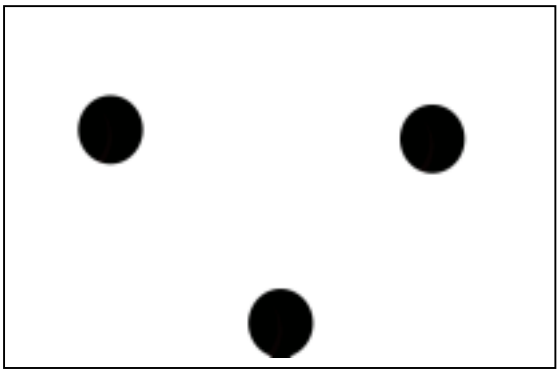
C. 移動可適性解交錯(Motion Adaptive De-interlacing，簡稱 MA)演算法

D. 時間遞迴解交錯(Time Recursive De-interlacing，簡稱 TR)演算法

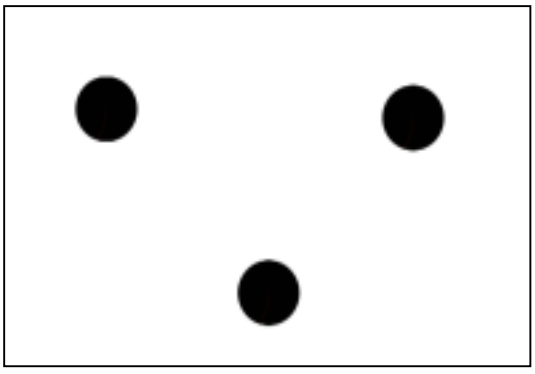
E. 可適性遞迴解交錯(Adaptive Recursive De-interlacing，簡稱 AR)演算法

F. 整合式解交錯(Integrated De-interlacing)演算法

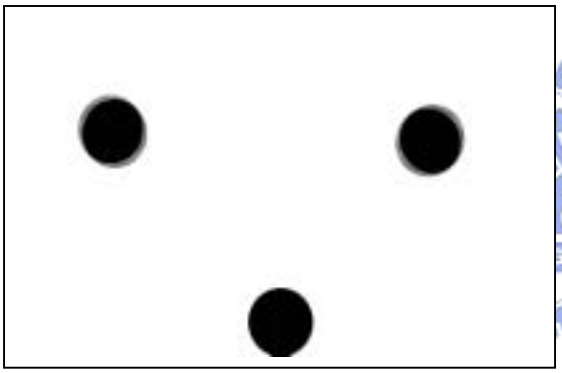
(1) 測試圖片(動態)



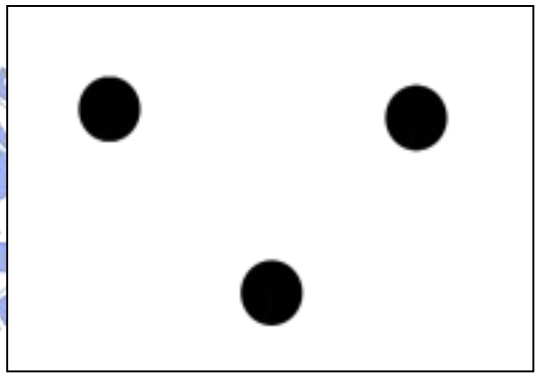
(a) 原始圖片



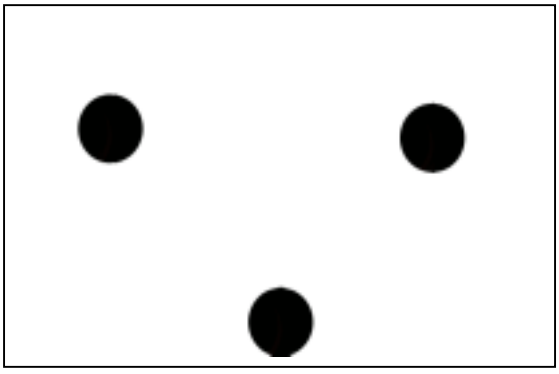
(b) 空間掃描線垂直平均解交錯



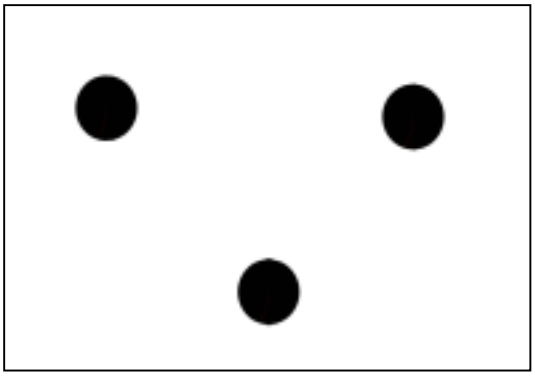
(c) 時間平均解交錯



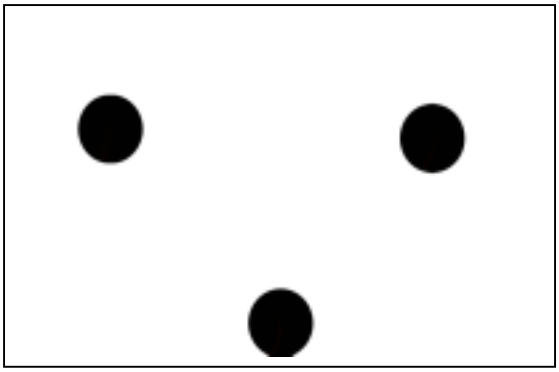
(d) 移動可適性解交錯



(e) 時間遞迴解交錯



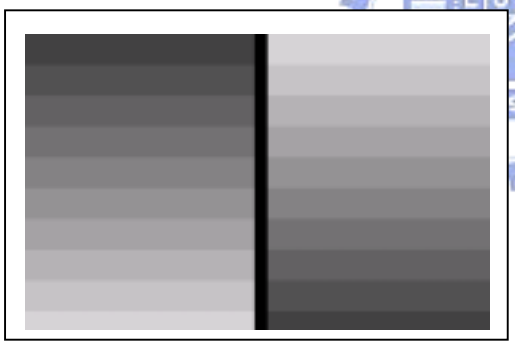
(f) 可適性遞迴解交錯



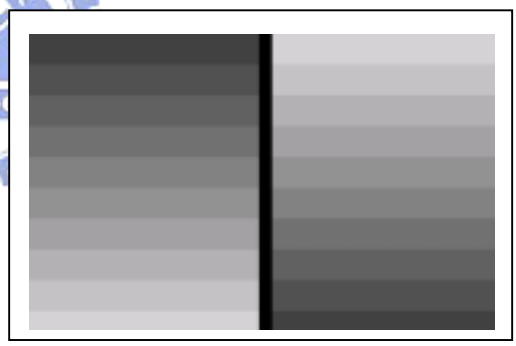
(g) 整合式解交錯

圖 5.6　第一種模擬方式的解交錯結果(一)

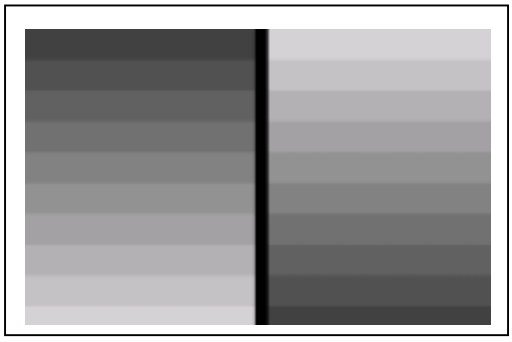
(2) 測試圖片(靜態)



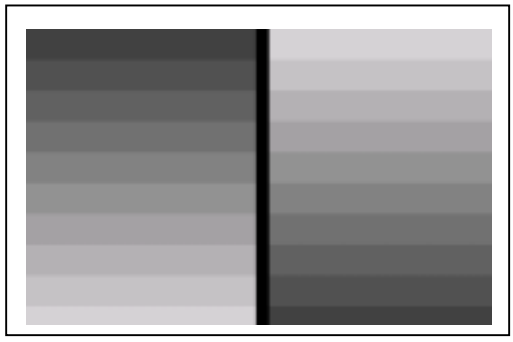
（a） 原始圖片



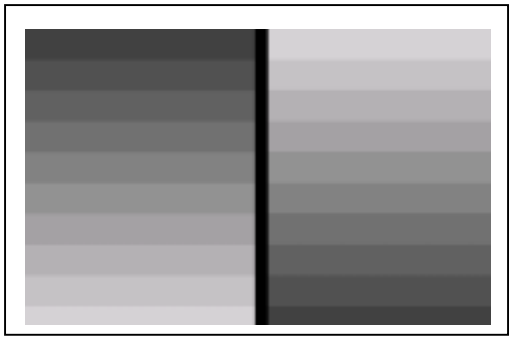
（b）空間掃描線垂直平均解交錯



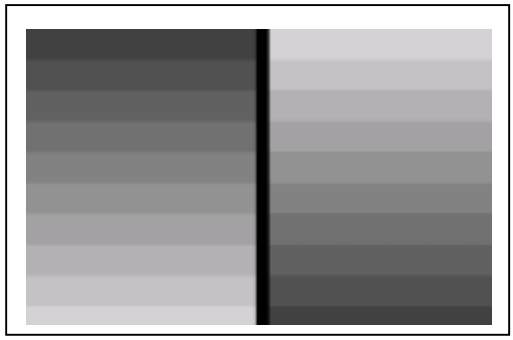
（c）時間平均解交錯



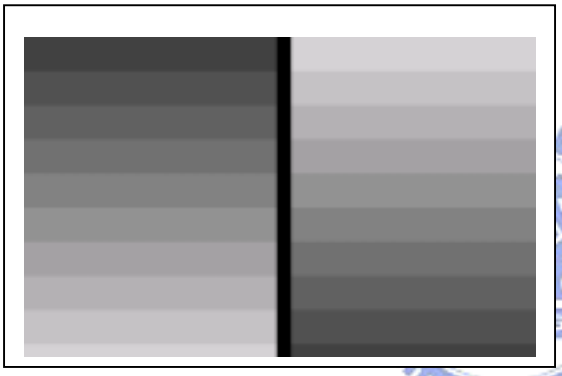
(d) 移動可適性解交錯



(e) 時間遞迴解交錯



(f) 可適性遞迴解交錯

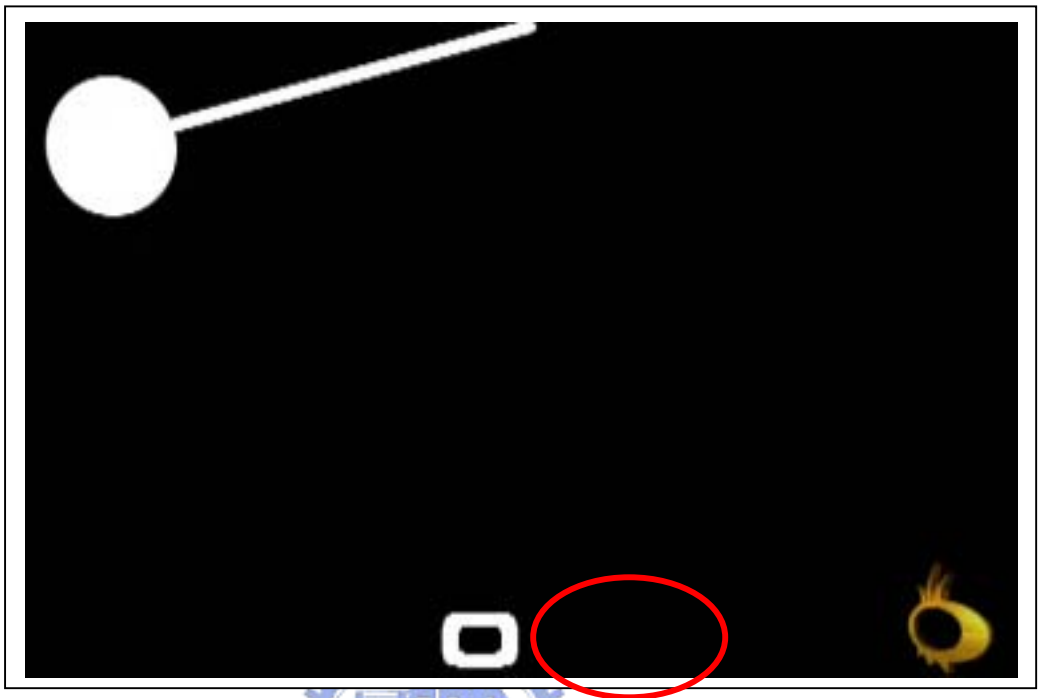


(g) 整合式解交錯

圖 5.7　第一種模擬方式的解交錯結果(二)

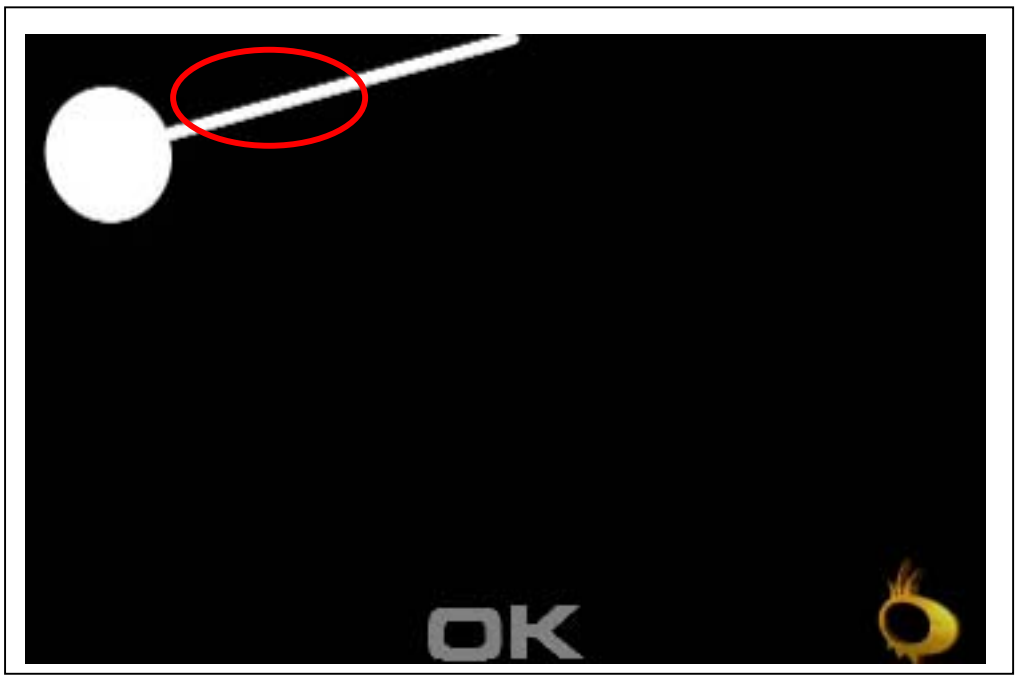
5.2.2 第二種解交錯模擬方式的模擬結果

(1) clock 測試圖片

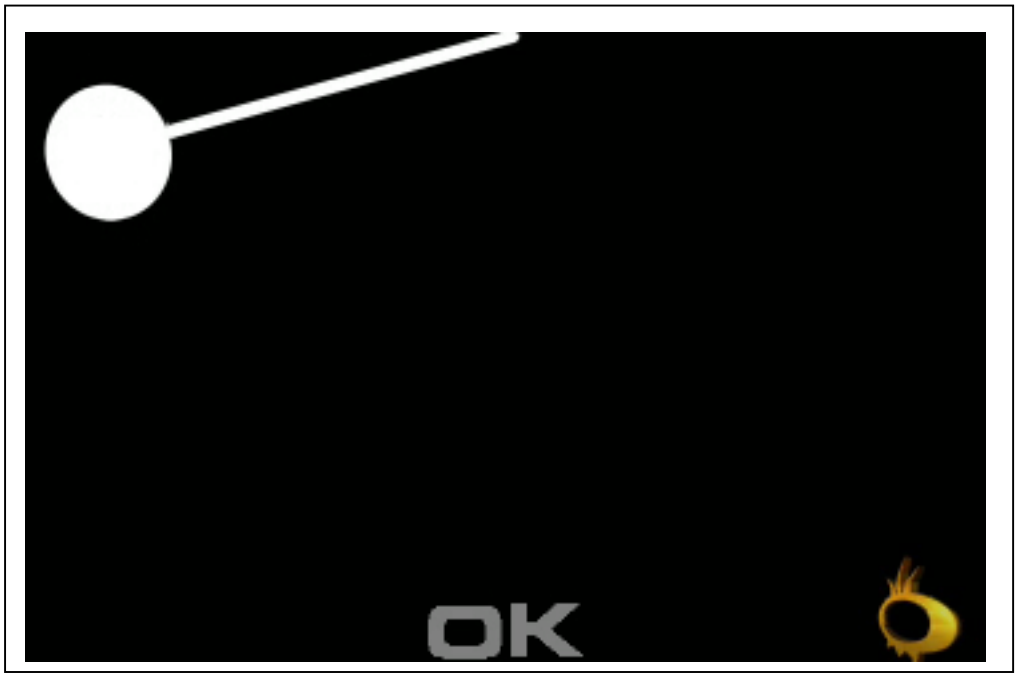


(a) 空間掃描線垂直平均解交錯

(b) 時間平均解交錯



(c) 移動可適性解交錯

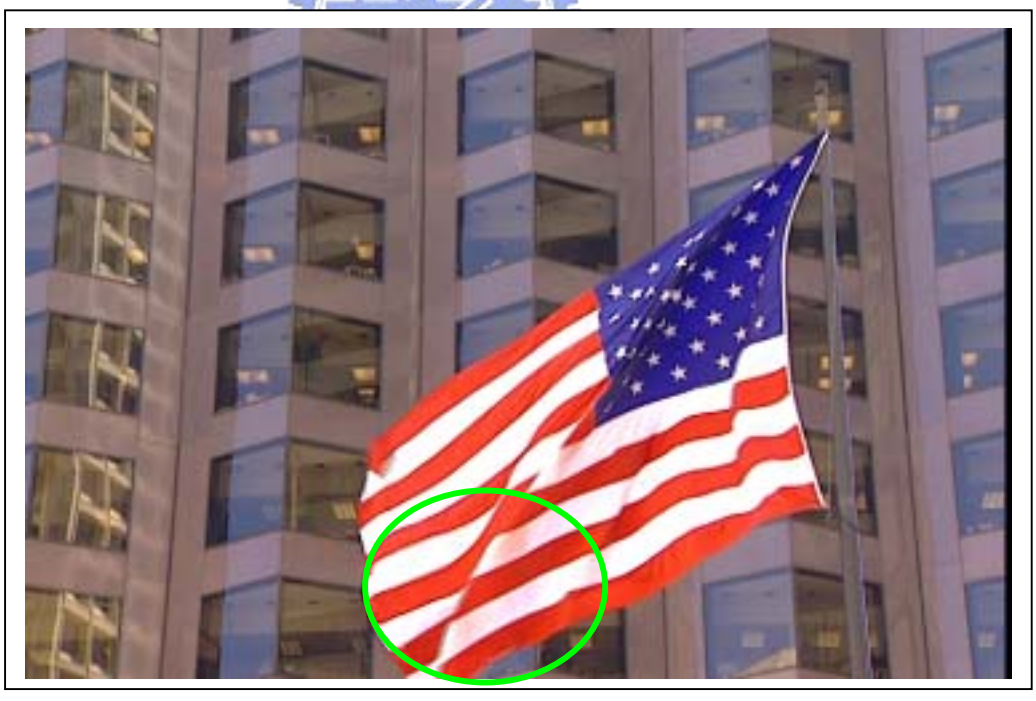


(d) 整合式解交錯

圖 5.8 第二種模擬方式的解交錯結果(一)

(2) flag 測試圖片

(a) 空間掃描線垂直平均解交錯



(b) 時間平均解交錯

(c) 移動可適性解交錯

(d) 整合式解交錯

圖 5.9 第二種模擬方式的解交錯結果(二)

(2) ball 測試圖片

(a) 空間掃描線垂直平均解交錯

(b) 時間平均解交錯

(c) 移動可適性解交錯

(d) 整合式解交錯

圖 5.10 第二種模擬方式的解交錯模擬結果(三)

5.3 解交錯演算法的模擬結果分析

在圖 5.6 和圖 5.7 是第一種模擬方式的解交錯結果，表四是不同解交錯的 MSE 直方圖，以平均值來說，本文所設計解交錯系統，有最好的效果，它的 MSE 值是最低的，當然這數據會因為不同的圖片而有數值的差異。在圖 5.6 中，它是動態的圖片，所以用時間平均解交錯的 MSE 值特別高，而其他非移動的解交錯的 MSE 值雖然不高，但在球形的邊緣仍有一些鋸齒狀，使用移動補償解交錯的方式能減少鋸齒狀的發生。在圖 5.7 中，它是靜態的圖片，反而圖場間合併平均解交錯的 MSE 值是零，垂直線平均解交錯(LA)的 MSE 值是最高的。

解交錯的效果，除了比較峯值訊號雜訊比(PSNR)及錯誤平方平均值(MSE)之外，最重要的視覺的感受，在圖 5.8 鐘擺測試圖中，它包含靜態的區域(字 OK 的部份)和動態區域(鐘擺的部份) ，其中 O 在偶數圖場，K 在奇數圖場，在圖 5.8(a)空間掃描線垂直平均解交錯的輸出圖中，因為它是空間解交錯的方式，所以它只有顯示 O 字而少 K 字圖。在圖 5.8(b) 中，使用時間平均解交錯，因為鐘擺在每一個時間點移動，所以會有嚴重的鋸齒狀。在圖 5.9 國旗飄揚測試圖中，主要是觀察國旗邊緣的是否平滑的移動。而圖 5.9(a)和 5.9(b)整個國旗的線條會被加粗，而且在國旗飄動的方向和國旗旗竿，會有輕微的鋸齒狀，而本文所提移動補償對這種現像有很大的改善。

圖 5.10 主要是觀察整個球的移動，因為是非常快速的轉動，所以用非移動補償的解交錯，因為無法找到正確的移動方向，所以畫面看起來會很模糊，移動補償解交錯的方式會根據移動向量找到正確補償的像素，所以畫面的解析度會提高很多。

第六章 結論與未來的工作

在本篇論文中，所提出整合式解交錯(Motion Compensated De-interlacing)的演算法，主要以移動補償 (Motion Compensation)為基本架構，並結合移動估計(Motion Estimation)、移動向量平滑處理(Motion Vector Smoothing) 、影片偵測及影像加強等演算法設計，將各類的圖片，再經過電腦模擬程式的測試後，都有不錯的結果。

但是在演算法設計過程中，移動補償解交錯的演算法有兩個很大的困難點。第一個困難的地方，移動偵測能否正確地辨別每一像素是移動或是靜止，若偵測的結果是靜止的話，圖場插點的像素則使用時間平均解交錯(Temporal Averaging De-interlacing)。反之，則使用移動補償或是角度偵測線平均(Edge Line Average)的方式。但移動偵測的辨別，卻很容易受雜訊的干擾或者是畫面本身的模糊所造成的影響。另外一個困難點，則是移動估計時，在圖片中無法找到相似的區塊時，或是受雜訊干擾無法找到正確的移動向量，經由移動向量平滑處理(Motion Vector Smoothing)之後，仍然無法修正錯誤的移動向量，則移動補償(Motion Compensation)後的結果，會使得顯示的畫面變得很差。所以要設計完美的移動估計及錯誤移動向量偵測，仍是未來努力的目標。

另外，以商業產品來看，論文所提的整合式解交錯演算法，仍有一些演算法尚未加入研究，我列出下面幾項，可作為日後研究的方向。

第一項 消除串色(Cross Color)演算法：

不論是電視訊號或是 DVD 播放機所輸出的訊號，都是先經過視訊解碼器(Video Decoder)後，才會傳送畫面到整合式解交錯中，但是訊解碼器若無法將 YC 作完全的分離，則會發生串色的現象，所以增加消除串色的演算法，會減少對整合式解交錯的影響。

第二項 消除雜訊(Noise Reduction)演算法：

一般的電視訊號或 DVD 播放機的訊號在影片拍攝時會有雜訊摻雜在畫面中，或是傳送中當中混入雜訊，若無法降低雜訊干擾，對作整合式解交錯時，會造成很大的偵測的誤判。

第三項 縮放比例(Scaling Ratio)演算法：

輸入訊號格式和輸出顯示格式，一般來說都是不相同的寬高，所以會有縮小或放大的比例關係存在，因此在整合式解交錯之後，若無法正確地作縮放比例(Scaling Ratio)，則輸出的結果，會出乎預期。

最後，在論文的研究當中，當整合越多的不同功能的演算法時，整個系統就會變得很複雜，而且要付出更多的努力，及加上實驗時的細心與耐心，才能達成目標。

參 考 文 獻

[1] E. B. Bellers and G. de Haan, De-interlacing – *A Key Technology for Scan Rate Conversion ,* Elsevier, 2000.

[2] G. de Haan and E. B. Bellers, "De-interlacing: an overview," Proceedings of the IEEE, vol. 86, pp. 1839-1857, September 1998.

[3] G. de Haan and E. B. Bellers, "De-interlacing of video data," IEEE Trans. On Consumer Electronics, vol. 43, pp. 819-825, August 1997.

[4] E. B. Bellers and G. de Haan, "Majority-Selection De-interlacer; An advanced motion-compensated spatio-temporal interpolation technique for interlaced video," *in Proc. of the Image and Video Communications and Processing Conference 2000*, vol. 3974, (San Jose, USA), pp. 386-395, January 2000.

[5] F. M. Wang, D. Anastassiou, and A. N. Netravali, "Time-recursive deinterlacing for IDTV and pyramid coding," Signal Process.: Image Commn. 2, pp. 365-374, 1990.

[6] P. Delogne, L. Cuvelier, B. Maison, B. V. Caillie, and L. Vandendorpe, "Improved Interpolation, Motion Estimation and Compensation for Interlaced Pictures," IEEE Tr. on Image Processing, vol.3, No. 5, pp. 482-491, September 1994.

[7] L. Vanderdorpe, L. Cuvelier, B. Maison, P. Quelez, and P. Delogne, "Motion – compensated conversion from interlaced to progressive formats", in Signal Processing: Image Communication 6. 1994, pp. 193-211, Elsevier.

[8] A. A. C. Kalker, "Motion Estimation and Compensation for Interlaced Video", to be in IEEE Tr. on Signal Processing.

[9] O. Kwon, K. Sohn, and C. Lee, "Deinterlacing using Directional Interpolation and Motion Compensation", IEEE Transactions on Consumer Electronics, Vol.49, No. 1,

February 2003.

[10] Y. Y. Jung, S. Yang, and P. Yu, “An Effective De-Interlacing Technique Using two Types of Motion Information”, IEEE Transactions on Consumer Electronics, Vol.49, No.3, August 2003.

[11] S. F. Lin, Y. L. Chang, and L. G. Chen, “Motion Adaptive Interpolation with Horizontal Motion Detection for Deinterlacing”, IEEE Transactions on Consumer Electronics, Vol.49, No.4, November 2003.

[12] B. T. Choi, S. H. Lee, and S. J. Ko, “New Frame Rate Up-conversion using Bi-directional Motion Estimation”, IEEE Transactions on Consumer Electronics, Vol.46, No.3, August 2000.

[13] K. Jack, *Video Demystified,* 3rd ed. , HARRIS, 2000.

自　　傳

曾坤源，男，民國五十八年一月十九出生於台灣省新竹縣，家中兄妹五人。民國八十五年畢業於國立台灣工業技術學院後，服務於工業界，並從事影像處理及影像壓縮相關的研發工作。民國九十年就讀國立交通大學電機資訊學院專班，亦從事影像處理相關的研究。