國立交通大學

電機學院通訊與網路科技產業研發碩士班

碩 士 論 文

視訊編碼標準 H.264/AVC 於 DSP 平台上之實現

Implementation of Video Coding H.264/AVC on DSP Platform

研 究 生：游育瑋

指導教授：王忠炫　教授

中 華 民 國 九十七 年 十 月

視訊編碼標準 H.264/AVC 於 DSP 平台上之實現
Implementation of Video Coding H.264/AVC on DSP Platform

研 究 生：游育瑋　　Student：Yu-Wei You

指導教授：王忠炫　　Advisor：Chung-Hsuan Wang

國 立 交 通 大 學

電機學院通訊與網路科技產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

Communication Engineering

October 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年十月

視訊編碼 H.264/AVC 於 DSP 平台上之實現

學生：游育瑋　　指導教授：王忠炫

國立交通大學電機學院產業研發碩士班

摘　　要

由於系統晶片 (System on a Chip，SOC) 技術的快速發展，因此藉由將系統架構導入晶片中開發設計的方式已蔚為趨勢。此種作法不僅可有效的達成系統整合的目的，而且也能在實作上大幅降低產品尺寸。藉此，本論文將針對 H.264/AVC 的視訊編碼、解碼系統，就移植於德州儀器硬體上所衍伸的相關問題進行探討。首先，我們將 H.264/AVC 的軟體編碼器、解碼器移植至德州儀器提供的數位訊號處理平台上去執行，並且利用該公司所提供的整合型開發環境，分析和解決在處理移植過程中所發生的軟、硬體相容性問題。此外，針對 H.264/AVC 編碼器中計算複雜度較高的移動估測 (Motion Estimation) 部分，我們也利用過去視訊標準中所使用過的各式估測演算法，比較在 H.264/AVC 下使用之計算複雜度與成像品質，試圖了解兩者相互影響的關連性，以期能在折衷之間達到最佳的效能。

Implementation of Video Coding H.264/AVC on DSP Platform

student：Yu-Wei You　　Advisors：Chung-Hsuan Wang

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

ABSTRACT

Due to the advance of system-on-chip (SOC) technology, we can achieve system integration by porting huge and high complexity systems to single chip. However it also produces system incompatible and complexity increasing through the process of system integration. The thesis aims at the complexity reduction for video multimedia system. We port H.264 encoder and decoder to the TI DSP platform and employ the code composer studio (CCS) to solve the problems of porting. Moreover, we replace the original motion algorithm of H.264 by a low complexity alternative, called the three step algorithm (TSS), which is employed as the motion estimation algorithm of MPEG-1、MPEG-2 and H.261. Verified by the simulation results, the proposed scheme provides a great improvement on PSNR performance while only increases complexity slightly.

致謝

首先非常感謝我的指導教授王忠炫老師在研究上給予我極大的幫助, 在 *metting* 時總是很有耐心的糾正我報告時邏輯上的缺點。同時也讓我建立起對自己的自信心, 讓之後人生的旅途中遇到困難時不再懼怕它。也要感謝老蔡、小白與阿尼在一起 *metting* 時給予我研究上的建議, 感謝蓬麟學長給予我修課上的建議, 感謝實驗室大師兄對於研究上的鼎力幫助 ,感謝施施與 *duck* 在我心情沮喪時給予我鼓勵, 感謝 *lab708* 實驗室的所有人帶給我充實且快樂的研究所時光。也要感謝在交大所有對我有幫助的老師與同學, 因爲有你 *(妳)* 們的幫助使得我能夠的順利的渡過很多修課與研究上的難關。最後我要感謝我的家人給予我最大的支持與鼓勵, 讓我能無後顧之憂的扮演好學生的角色。

目錄頁

圖目錄

Chapter 1

緒論

1.1 研究動機與目的

打從有電話出現人們都是透過耳朵來聆聽對方的聲音, 但是藉由科技進步所賜, 人們不必在侷限於只用耳朵去聆聽對話者的訊息, 而是可以眞實的在一個行動視訊的螢幕上眞實的看到對方的表情與肢體動作, 就如同是與對話者是面對面眞實的交談一樣。不過如果就單純利用攝影機或一些影像擷取的器材取得該畫面, 必須至少要一秒鐘取得30張連續的畫面, 人類的眼睛才會感覺是連續的動作, 而不是斷斷續續的畫面。所以可想而知畫面所佔的空間是比可傳送的頻寬還要大很多, 因此便需要將畫面所佔的空間盡量壓縮到人類眼睛可容許的失眞範圍內, 用來節省在傳輸時所需要的頻寬, 即是本篇論文所要探討的視訊壓縮標準 H.264 系統被發展出來的原因 。

然而在視訊壓縮標準的推陳出新下, 人們便希望可以將所有的多媒體資訊都傳輸在可傳送的頻帶上, 這樣使得需要傳送的資料量越來越大且又必須在視訊品質不降低的前提下去達到, 所以也相對的衍伸出一些問題, 即是運算複雜度的問題與功耗的問題 (用於行動視訊上), 因此如何將運算複雜度壓低且又要維持相對的視訊品質, 是行動視訊在開發上的一個主要目標。

而本文所追求的是利用德州儀器專爲視訊編碼與通訊編碼所推出的一系列數位訊號處理器, 所以除了將視訊編碼 H.264/AVC 平行的移植到 CCS 平台上之外, 還必須要系統可以相容於數位訊號處理器中, 以達成行動視訊系統的開發與移植的目的。之後則是利用一些前人所提出的演算法, 將原本 H.264/AVC 系統中計算複雜度較高的演算法用較低複雜度的演算法取代掉, 且還要維持一定的視訊品質的要求。並且也利用所一些常見的最佳化方法再去進一步的加快執行速度。使得視訊編碼 H.264/AVC 可以達到即時性 (Real-Time) 的一個需求。

1.2 論文架構

在本篇論文中, 第二章將介紹基本的影像與視訊處理的觀念作爲研讀後面章節的基礎, 第三章將簡單的描述視訊壓縮原理的演進與系統間的介紹, 包括 MPEG 視訊標準中的 MPEG-1 、MPEG-2 、MPEG-4 與 ITU-T 視訊標準中的 H.261 、H.263 之介紹 。第四章則針對本篇論文所要移植的系統 H.264 做完整的介紹。第五章則是本篇論文所要處理的問題, 第六章是結論與未來展望, 與第七章的參考文獻, 附件則是介紹整合型開發環境 (Code Composer Studio,CCS) 此模擬環境的環境介紹。

Chapter 2

基本的影像與視訊處理的觀念

2.1 影像處理導論

在現今的科技資訊社會中, 電腦在資訊處理中扮演著非常重要的作用, 影像處理大致的分類如 Figure 2.1 所示。對於不同種類的影像在處理上就必須依照處理結果的精細度, 處理速度的要求等, 選擇相對應的處理方式。在傳統類比影像爲主的電視世界中, 由於數位化技術的發展與電腦技術的介入, 現今以開始進入全面數位化的數位廣播時代了。本來只能處理文字的處理機現在已經發展到也可以處理影像, 且從處理黑白影像朝向彩色動態影像發展。且電腦圖學技術的成熟, 已經開始用於影像處理的各個領域, 如 Figure 2.2 所示。

接下來簡單的介紹實際生活常見影像處理的實例:

- 辦公用影像處理: 把文件、圖形影像化, 以黑白二值影像爲處理的文件。
- 醫療用影像處理: 針對手術中無法清楚觀察或者無法以肉眼觀察的部分進行影像處理的轉換, 以幫助手術的順利進行。
- 遙感影像處理: 對經由人造衛星所拍攝的遙感影像進行處理, 以方便取得相關地球資源或氣象變化的資訊, 可廣泛用於漁業、農業、環境污染調查與程式規劃等方面。
- 廣播電視與電影中的影像處理: 在廣播電視或電影領域中應用非常多的特效處理, 更能增加廣播電視或電影的可看性。

影像處理除了可經由電腦進行影像處理外, 還可使用光學系統設備進行類比處理。例如, 照相與攝影等, 其中包括使用濾色鏡以實現柔和氣氛, 利用長時間的快門曝光, 以便可清楚拍攝星辰的活動等。而數位處理, 可以透過利用電腦的的計算實現與此同樣的結果。且它的優劣如下:

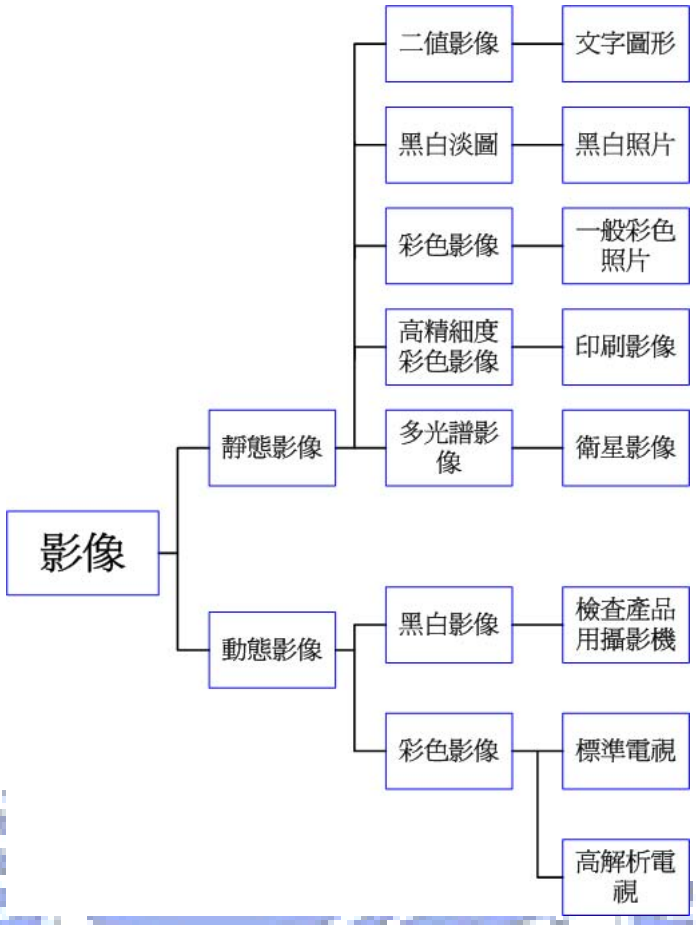

Figure 2.1: 各種影像

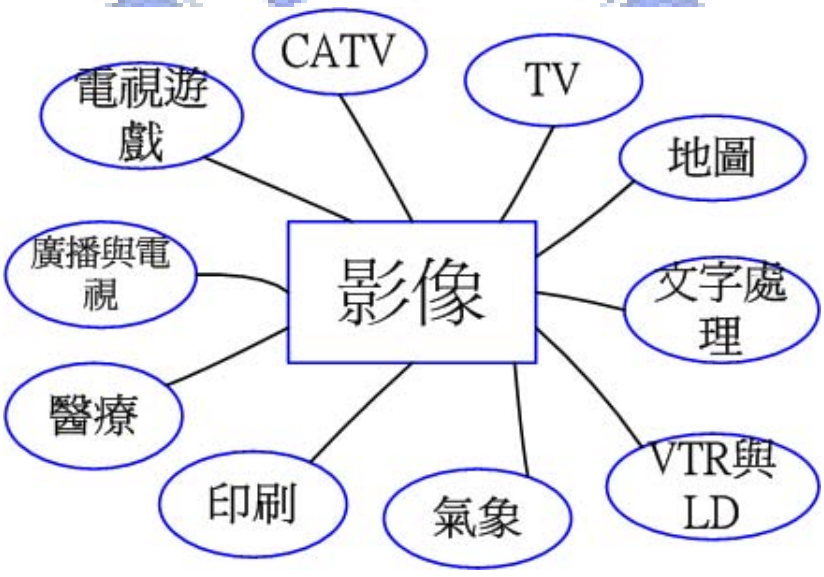

Figure 2.2: 影像環境

- 可保證處理結果的正確性與再現性: 因爲利用電腦進行處理, 程式可以反覆的執行直到結果正確滿意爲止, 且再現起來也十分的容易, 只需要讓同樣程式重複執行即可得到相同的結果。
- 易於控制: 透過程式可以自由的設定或變更各種參數。
- 處理的多樣化: 因影像處理都是透過程式來進行, 所以只需修改程式便可以實現各種的處理。
- 資料量過多: 影像經數位化後的資料量過於的龐大。
- 需佔用一定的時間: 由於資料量過於的龐大, 所以勢必處理上所花的時間也必然的增加。

2.2 基本的數位影像處理

2.2.1 影像的表示式

一個影像可定義成一個二維的函數 $f(x,y)$, 其中 x 和 y 爲空間 (平面) 座標 (Spatial Coordinates), 如 Figure 2.3 所示 , 而 f 在任一組座標 (x,y) 處的振幅稱爲影像在該點的強度 (Intensity) 。灰階 (Gray Level) 這名詞通常用來指單色影像的強度, 而彩色影像則是由各個二維影像組合而成。一個影像在 x 和 y 座標值以及振幅上都可以是連續的。將一個影像轉成數位形式, 需要將座標和振幅都數位化。將座標數位化稱爲取樣 (Sampling) ; 將振幅數位化則稱爲量化 (Quantization)。因此當 x 、y 和 f 的振幅值皆爲有限且爲離散值時, 我們稱這影像爲數位影像 (Digital Image)。一個影像 $f(x,y)$ 經過取樣後形成一個具有 M 列 N 行 的影像, 我們稱這個影像的大小爲 $M \times N$; 座標值 (x,y) 爲離散值 。其中 x 的範圍是從 0 到 M-1 , y 的範圍是從 0 到 N-1 , 且呈整數值增加。我們可得到一個數位影像函數的表示式:

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{pmatrix}$$

此式的右邊部分定義是數位影像, 此陣列的每一個元素稱爲影像元素 (Image Element) 、像素 (pixel) 或 像素點 (pel)。

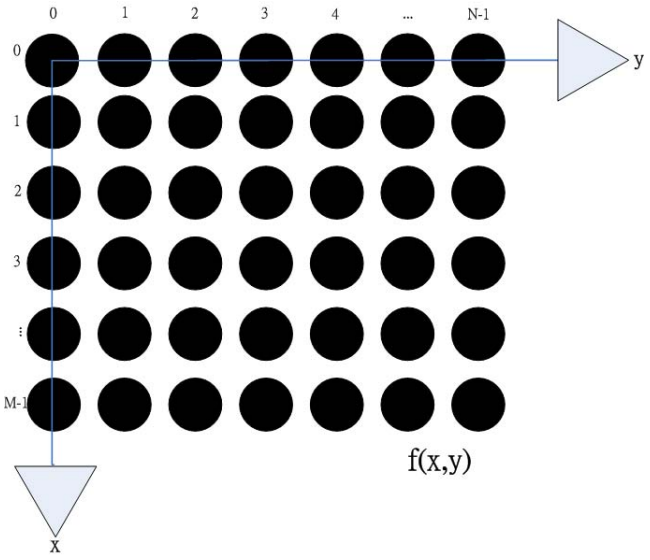

Figure 2.3: 影像座標

2.2.2 取樣與量化

爲了適用於電腦處理, 影像函數 $f(x,y)$ 在空間和振幅上都需要進行數位化。空間座標 (x,y) 的數位化稱爲影像取樣 (Image Sampling), 振幅的數位化稱爲灰階量化 (Gray-Level Quantization)。而影像的解析度可以分爲兩個部分來探討:

- 空間的解析度: 取樣越細, 越能表現出影像的細微變化。
- 灰階的解析度: 灰階越細, 越能表現出豐富的顏色。

但是原則上, 解析度越高, 越能表現實務的細節與色澤, 但解析度高到某一程度的話, 肉眼便難以分辨其差異, 不過解析度越高影像所帶的資料量便越大。

2.2.3 影像的形成

影像是指一個二維的光強度函數, 它用 $f(x,y)$ 來表示。f 在空間座標 (x,y) 處的值或振幅給出影像在某點的光強度 (亮度)。由於光是能量的一種形式, 故 $f(x,y)$ 必爲一個非零值, 並且是有限的一個值, 即是 $0 < f(x,y) < \infty$ 。人們在視覺活動中所感受到的影像通常是因爲物體反射光所形成的。$f(x,y)$ 可以用兩個分量來表示其特徵: 一. 光源所看到景物的入射光量 二. 場景中物體的反射光量。相對應的被稱爲照明 (Illumination) 與反射分量 (Reflectance Component) , 分別記爲 $i(x,y)$ 與 $r(x,y)$。函數 $i(x,y)$ 與 $r(x,y)$ 之乘積形成 $f(x,y)$: $f(x,y) = i(x,y)r(x,y)$, 且式中 $0 < i(x,y) < \infty$ 與 $0 < r(x,y) < 1$ 。

2.2.4 人類的視覺系統

人的眼睛有著接收及分析視像的不同能力, 從而組成知覺, 以辨認物象的外貌和所處的空間(距離), 及該物在外形和空間上的改變。腦部將眼睛接收到的物象信息, 分析出四類主要資料; 就是有關物象的空間、色彩、形狀及動態。有了這些資料, 我們可辨認外物和對外物作出及時和適當的反應。當有光線時, 人眼睛能辨別物象本體的明暗。物象有了明暗的對比, 眼睛便能產生視覺的空間深度, 看到物件的立體程度。同時眼睛能識別形狀, 有助我們辨認物體的形態。此外, 人眼能看到色彩, 稱爲色彩視或色覺。此四種視覺的能力, 是混爲一體使用的, 作爲我們探察與辨別外界資料, 建立視覺感知的源頭。眼睛除了要辨認物象的特徵, 還要知道物件的位置, 及其活動上的變化, 才可驅使身體其他部位作出相應的動作。在理解自身與外界之間的距離或深度, 人類的知覺, 可從視野所得的資料中, 抽出有關空間的提示, 從而知識到自己與各種物件的距離。視網膜是視覺的核心, 它是一片平面的薄膜, 獲得的物象是平板而缺乏立體感的。所以知覺需要組織起其他信息, 才能做出對深度的感知。人類的眼球天賦便有辨別立體深度和距離的本能, 因爲人類是用雙目平排而視。同時通過外物在視野範圍中所形成的物象大小, 以及排列或表現的狀態, 認知該物與我們的距離。甚至可通過形狀及色彩獲得有關距離的資料。眼睛能看到物體的移動, 有助辨別物體的方向和運動的速度。

視覺是指視覺器官眼睛(或眼球), 通過接收及聚合光線, 得到對物體的影象, 然後接收到的信息付會傳到腦部進行分析, 以作爲思想及行動的反應。要感知外在環境的變化, 要靠眼睛及腦部的配合得出來, 以獲得外界的信息。人類視覺系統的感受器官是眼球。眼球的運作有如一部攝影機, 過程可分爲聚光和感光兩個部份。

眼球是整個的包裹在一層鞏膜 (Selear) 之內, 此層鞏膜就如攝影機的黑箱, 並分爲前、後兩段。眼球前段是聚光的部份, 是由眼角膜 (Cornea) , 瞳孔 (Pupil) , 水晶體 (Lens) 及玻璃體所組成。它們的功能是調節及聚合外界入射的光線。光線首先穿過眼角膜這片透明薄膜, 經由瞳孔及水晶體, 將光線屈曲及聚合在眼球的後段。瞳孔是一個可透光的開口, 能因應光度的強弱, 而調節其圓週的大小。當在暗黑的情況下, 瞳孔的直徑會擴大, 可引入更多的光線。而在光線充足的情況, 瞳孔的直徑會收縮, 令入眼的光量不致太強。瞳孔和水晶體兩者配合之下, 眼球可接收強、弱、遠、近各種不同的光線來源。眼球內有睫狀肌 (Ciliary Muscles), 它的伸拉作用可使水晶體變形, 因而調節屈光度, 使光線能聚焦到視網膜上而形成影像。當光線來自近距離物件時, 水晶體變得較拱圓, 屈光度較大。當光線來自較遠的物件時, 水晶體變得較扁平, 屈光度則較低。以確保在不同的光度下, 進入眼球的光線水平能形成最高質素的影像。

眼睛後段是感光的部分。後段有視網膜, 它是由兩種感光細胞所組成, 這兩種細胞因其形狀

而名爲桿狀細胞 (Rod Cells) 和錐狀細胞 (Cone Cells), 作用是將水晶體聚焦而成的光線變成電信號, 並由神經細胞送往腦部。

	錐狀細胞	桿狀細胞
位置	視網膜中央黃點	黃點周邊
數量	約六百萬	約一億兩千五百萬
顏色辨別	可分為三種類型，分別針對紅、綠、藍色亮光敏感	只對亮度敏感
光暗辨別	只對中至強光敏感	能感受微弱光線變化
動作辨別	不敏銳	敏銳

Figure 2.4: 感光細胞

當外界的光線信息進入眼球後, 會被眼球內的神經細胞轉變爲電信號, 再被傳輸送到腦袋中。腦部接收電信號之後, 會引起連串的思維活動, 並作出適當的行動或反應。視網膜上的神經會聚並連結到大腦的一點, 由於沒有光線的受體, 所以大腦無法感知聚焦該處的影像, 故稱盲點。感覺光暗的桿狀細胞和感覺色彩的錐狀細胞在視網膜表面並不是平均分佈的, 在感知中起重要作用的錐細胞大部份集中在視網膜中的一小片稱爲黃點的地方。因此我們在觀看景物和閱讀時, 注意力只是集中在視野範圍一半不到的區域。

一個視能正常的人, 能分辨在視網膜上來自不同投影的影像。這種能力稱爲“視覺敏銳度”。接近視網膜的中央, 距離眼角膜最遠的地方, 這位置稱爲黃點 (Fovea) , 是感光細胞最密集, 視覺敏銳度最高的位置。當我們要看清一件物件時, 我們會轉動眼球, 直至影像聚焦在黃點上。離開黃點越遠, 感光細胞越少, 影像越不清晰, 如影像聚焦在黃點以外的地方, 我們可看見一件物件的存在, 但未必知道這件物件是甚麼。

2.3 彩色影像處理

2.3.1 RGB影像

所謂的 RGB 彩色影像其實是一個色彩像素的 $M \times N \times 3$ 的陣列, 其中色彩像素是特定的一個空間且對應到一張 RGB 影像上的三個不同成分, 分別是紅、藍、綠三種成分影像, 藉由這三種成分影像合成爲 RGB 影像上的一個色彩像素點。且一張 RGB 的影像可以看成是一個三個灰階影像的堆疊, 使得當我們輸入紅、藍、綠成分進顯示器時, 會在螢幕上顯示一張彩色的影像。而用來表示成分影像的像素數值區間, 便要決定 RGB 影像的位元深度。例如, 如果每個成分影像都被表示成8位元的話 ,則整個 RGB 影像所佔的位元便爲24位元, 且可以表示的色彩數目爲 $(2^8)^3$ 色。RGB 色彩空間通常是以 Figure 2.5 所示之 RGB 彩色立方圖來表示, 立方體的頂點表示光線的主要顏色 (紅色、綠色、藍色) 及次要顏色 (藍綠色、紫紅色、黃色)。

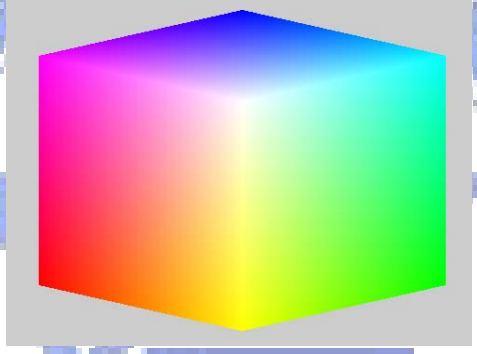

Figure 2.5: RGB色彩立方圖

2.3.2 NTSC色彩空間

NTSC 色彩系統用於美國的電視。此色彩空間的優點是將灰階資訊從彩色資訊中分離出來, 因可以將相同的訊號提供給彩色與黑白電視機使用。在 NTSC 色彩空間中, 影像資訊包括了三個主要成分: 照度 (Luminance)(Y)、色調 (Hue)(I) 以及飽和度 (Saturation)(Q)。可以由 RGB 成分取得 YIQ 成分, 利用以下轉換式:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ I &= 0.596R - 0.274G - 0.322B \\ Q &= 0.211R - 0.523G + 0.312B \end{aligned} \tag{2.1}$$

且在電視 6MHz 的頻寬中,4MHz 分配給 Y 使用,1.5MHz 分配給 I 使用,0.6MHz 分配給 Q 使用。

2.3.3 YCbCr色彩空間

YCbCr 色彩空間用於數位視訊中, 亮度的資訊以單一成分 Y 表示, 而色度資訊則是儲存成兩種色差成分, 即 Cb 與 Cr 。其中 Cr 成分表示藍色成分與參考值的差距, 而 Cr 成分則被表示爲紅色成分與參考值的差距。可利用 RGB 成分取得 YCbCr 成分, 利用下面的轉換公式:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= 0.5R - 0.419G - 0.081B \\ Cr &= -0.169R - 0.331G + 0.5B \end{aligned} \tag{2.2}$$

2.3.4 HSV色彩空間

HSV 是人們從色盤中或色彩輪中選擇色彩 (油墨或油漆之顏色) 時所使用的幾個色彩之一, 而所謂的 HVS 指的是色調 (Hue)、飽和度 (Saturation) 與色深度 (Value)。因爲此色彩比起 RGB 系統更爲接近人類所描繪與體驗的色彩感受。若是用藝術的術語來說, 色調、飽和度與色深度, 大致上指的是色澤、濃淡與色感。

- Hue 色調: 例如紅、黃、藍、橙、綠、紫等顏色, 範圍變化由0 - 360度。
- Saturation 飽和度: 範圍從0 - 100% (0.0 - 1.0),S=1色彩最純淨 (不含任何白光), 色彩也最鮮豔。
- Value 值: 範圍從0 - 100% (0.0 - 1.0),V=0爲最黑 (RGB=0)、V=1爲最亮 (RGB=255)。

且一樣可以利用 RGB 成分去取得 HSV 成分, 利用以下的轉換方法:

$$MAX = \max(R, G, B) \tag{2.3}$$

$$MIN = \max(R, G, B) \tag{2.4}$$

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} + 0 & \text{if } MAX = R, G \geq R \\ 60 \times \frac{G-B}{MAX-MIN} + 360 & \text{if } MAX = R, G < R \\ 60 \times \frac{B-R}{MAX-MIN} + 120 & \text{if } MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240 & \text{if } MAX = B \end{cases}$$

$$S = \frac{MAX - MIN}{MAX} \tag{2.5}$$

$$V = MAX \tag{2.6}$$

if MAX = MIN ,the H is undefined

if MAX = 0 ,then S is undefined

2.3.5 CMYK色彩空間

藍綠色、紫紅色與黃色是光的次色彩, 也就是顏料的主顏色。例如, 當一個塗上藍綠色顏料的平面被白色光照射時, 此平面並不反射紅色光。也就是, 藍綠色顏料在被反射的白色光中消滅掉紅色光, 而此白色光就是由等量的紅、綠、藍色組成。大部分將彩色顏料置於紙上的設備, 如彩色印表機與影印機, 都需要 CMY 資料輸入或執行 RGB 至 CMY 的轉換。如下所示:

$$\begin{aligned} C &= 1 - R \\ M &= 1 - G \\ Y &= 1 - B \end{aligned} \tag{2.7}$$

其中假設是所有的色彩值都已經正規化至 [0,1] 範圍內, 此公式展釋出從塗有純藍綠色平面的反射光線並不包含紅色 (即爲公式中的 C = 1-R)。同理, 純紫紅色並不反射綠色, 而純黃色也不反射藍色。

理論上將印刷三原色混合之後, 應該可以將紅綠藍光通通吸收而得到黑色, 只是現實生活中並找不到這種光線吸收、反射特性都十分完美的顏料, 將三種顏色混合後還是會有些許光線反射出來, 而呈現暗灰色或深褐色。事實上除了黑色外, 用顏料三原色也無法混和出許多暗色系的顏色, 爲了彌補這個缺點, 因此實際印刷的時候會額外加入黑色的顏料, 以解決無法產生黑色的問題。因此就有所謂 CMYK 的色彩模式,K 表示黑色。

黑色的加入雖然增加可印刷的顏色範圍, 卻也使顏色的調整更爲複雜, 例如用50%的 CMY 可以混合成灰色, 但我也可以直接用50%的黑色來產生, 變成同一種顏色有不同的混和方法, 在加上顏料的透明度、乾燥速度、紙張吸墨程度及作業流程種種條件的不同, 使得顏色的控制成爲印刷的一大問題, 如果你的作品需要送印刷廠的話, 一定要對 CMYK 有相當程度的瞭解才行。

2.3.6 HSI色彩空間

HSI分別代表著色調 (Hue)、飽和度 (Saturation) 與強度 (Intensity), 它將彩色影像裡將帶有色彩資訊的強度成分分離出來。所以 HSI 模型是在發展以色彩描述爲基礎之影像處理演算法的理想工具, 而且此描述方法對人類來說是很自然且符合直覺, 因爲畢竟人們是這些演算法的開發著跟使用者。HSI 色彩空間與 HSV 色彩空間有點相似, 但它的焦點著重於呈現當透過色彩藝術家的調色盤來解讀時, 是一個有意義的的色彩。同樣也是可以由利用 RGB 色彩空間將 HSI 成分抽取出來, 利用以下的轉換方法:

$$H = \begin{cases} \Theta & \text{if } , B \leq G \\ 360 - \Theta & \text{if } , B > G \end{cases}$$

$$\Theta = cos^{-1}\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)^{1/2}]}\} \tag{2.8}$$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)] \tag{2.9}$$

$$I = \frac{1}{3}(R+G+B) \tag{2.10}$$

2.4 視訊的基本觀念

2.4.1 組成視訊 (Component Video)

高階視訊系統利用紅 (R) 藍 (B) 綠 (G) 的影像平面所形成的三個訊號, 每一個顏色通道有單獨的視訊訊號。

- 大多電腦系統使用三個分別的 RGB 組成視訊。
- 對所有顏色分開的機制而言, 因爲三個通道間沒有干擾 (Crosstalk) , 使用組成視訊有最好的顏色再生。
- 組成視訊需要較多的頻寬及需要良好的三原色同步。

2.4.2 混合訊號 (Composite Signal, 單一訊號)

顏色 (Chrominance) 與亮度 (Luminance) 混合成單一訊號

- 顏色是由兩個顏色元素所組成 (I 及 Q 或 U 及 V) 。
- NTSC 電視系統中,I 及 Q 結合成一個 Chroma 訊號, 而且一個顏色副載波 (Subcarrier) 用來將這個 Chroma 訊號放在與亮度訊號共享的高頻尾端。
- 接收端可以將顏色與亮度元素分開, 然後兩個元素就可以還原。

2.5 類比視訊

一個類比視訊將隨著時間變化的影像取樣。所謂漸進式 (Progressive) 的掃瞄是將一個完整的影像 (Frame) 在每個時間區間內做一列一行的追蹤掃瞄。在電視、一些監視器與一些多媒體標準還有其他系統, 則使用交錯式 (Interlace) [1] 的掃瞄方式:

- 先做奇數線條的掃瞄, 然後再做偶數線條的掃瞄。這樣奇偶交錯方式的掃瞄可以組成一個完整的影像掃瞄。
- 事實上, 奇數行最後是結束在奇數行範圍內最後一行的中間點, 而偶數行的開始是在中間。

因爲有時必須改變畫速 (Frame Rate) , 改變大小, 或者要從一個交錯的視訊中顯示該靜止畫面, 會有各式各樣的方法來還原交錯畫面 (De-Interlace) 。

- 最簡單的方式是忽略其中奇或偶數行範圍的一個, 而且複製另一個範圍。所以某一個範圍資訊會遺失。
- 另一個方法是將所有範圍的資訊都考慮進來。

2.6 數位視訊

2.6.1 NTSC視訊

NTSC (National Television System committee) 電視標準在北美及日本最常用。所採用的是熟悉的 4:3 寬高比例 (Aspect Ratio) 並且使用一個畫面 525 條掃瞄線, 每秒鐘有 30 張畫面 (實際上是29.97張畫面)。

[1]交錯式 (Interlace) : 因爲交錯, 奇數與偶數行在交錯的時段顯示, 通常看不出來, 除非有非常快的動作在畫面中顯現, 才會出現殘影。

- 每一個畫面被分成兩個範圍, 每個範圍有 262.5 條掃瞄線。(交錯式掃瞄)
- 因此水平線掃瞄頻率是 $525 \times 29.97 = 15{,}734$ lines/sec , 所以每條掃瞄線要花 1/15734 秒 $= 63.6\mu sec$。

因爲 Horizontal Retrace 要花 $10.9\mu sec$, 所以只剩下 $52.7\mu sec$ 給影像顯示時的掃瞄線訊號。且 Vertical Retrace 的啓動需要奇偶行範圍內的前面 20 條掃瞄線當作是控制資訊, 因此畫面內實際負責顯示的掃瞄線只有 485 條。同樣地, 光柵 (Raster) 左邊的 1/6 是不亮的, 原因是用來要做 Horizontal Retrace 及 Sync。剩下可以顯示的像素稱爲 Active Pixels。NTSC 使用 YIQ 顏色模型, 而且利用弦調變技術來將共用頻譜的 I (In-phase) 與 Q (Quadrature) 訊號結合成一個單一的顏色訊號 C :

$$C = I\cos(F_{sc}t) + Q\sin(F_{sc}t)$$

調變後的 Chroma 訊號就是所謂的顏色副載波 Color Subcarrier , 它的量就是 $\sqrt{I^2 + Q^2}$, 而且相位 (Phase) 是 $\arctan(Q/I)$,C 的頻率就是 F_{sc} 近似 3.58 MHz 。而 NTSC 的混合訊號是一個將亮度訊號 Y 以及色度訊號 (Chroma Signal) 再一次混合, 如下:

$$composite = Y + C = Y + I\cos(F_{sc}t) + Q\sin(F_{sc}t)$$

NTSC訊號的解碼是在接收端將 Y 和 C 訊號分開, 利用 low-pass filter 將 Y 分出, 剩下的 chroma signal C 可以還原調變並分別抽出元素 I 和 Q 抽出的方式如下:

1. 將訊號 C 乘上 $2\cos(F_{sc}t)$ 。
2. $C \times 2\cos(F_{sc}t) = [I \times 2 \times \cos^2(F_{sc}t)] + [Q \times 2 \times \sin(F_{sc}t) \times \cos(F_{sc}t)] = I \times (1 + \cos(2F_{sc}t)) + [Q \times 2\sin(F_{sc}t)\cos(F_{sc}t)] = I + [I \times \cos(2F_{sc}t)] + [Q \times \sin(2F_{sc}t)]$ 。
3. 利用 low-pass filter 來獲得 I 並將其他高頻成分忽略掉$(2F_{sc}t)$ 。
4. 抽出 Q 同上步驟。

2.6.2 PAL視訊

PAL (Phase Alternating Line) 是在北歐, 中國大陸, 印度及其他國家廣泛使用的一個標準。PAL使用 625 條的視訊框架, 每秒可顯示 25 個框架, 使用 4:3 的寬高以而且採用交錯顯示。

- PAL 使用 YUV 顏色模型, 它採用一個 8MHz 通道 (Channel) 而且將 5.5MHz 的頻寬配給 Y , 而 1.8MHz 的頻寬分別給 U 及 V , 而色彩的副負載頻率 f_sc 是 4:43MHz。

- 爲了要提高影片的品質,Chroma 的訊號在連續的掃瞄線上有不同的正負號, 這也就是 Phase Alternating Line 的由來。
- 這樣的方式就可以讓接收端方便利用到一個 (Line Rate) Comb Filter , 也就是說到這個連續掃瞄線的訊號可以相加平均後去掉 Chroma 訊號而分別得到 Y 及 C , 並可得到高品質的 Y 訊號。

2.6.3 SECAM視訊

SECAM (System Electronique Couleur Avec Memoire) , 排名第三的主要廣播電視標準,SECAM 也是使用 625 條掃瞄線, 可每秒顯示 25 視訊框架, 寬高比是 4:3 而且是交錯顯示。SECAM 與 PAL 非常相似, 他們在顏色的編碼上有一點不同:

- SECAM 中,U 及 V 訊號是調變在不同的色彩副載波上, 分別是 4.25MHz 與 4.41MHz 。
- 分別是在不同的掃瞄線送出, 也就是只有一個 U 及 V 訊號送到一條掃瞄線上。

Figure 2.6 則是針對所介紹的三種數位視訊做一個完整的比較。

規格	NTSC	PAL	SECAM
制訂國家	美國	英國、德國等西歐國家	法國
掃瞄線數	525	625	819
每秒圖像數	30	25	25
視訊調變	AM	AM	AM
視訊頻寬	4.2MHz	5~6MHz	5~6MHz
音訊調變	FM	FM	FM 或 AM
圖場頻率	59.94Hz	50Hz	50Hz
掃瞄方式	間歇式	間歇式	間歇式
寬高比	4：3	4：3	4：3

Figure 2.6: 視訊比較表

2.6.4 HDTV (High Definition TV)

HDTV 的觀念是由寬銀幕的電影而來的, 也就是家庭劇院的概念。當寬銀幕的電影剛剛在市場上推出時, 電影製片發覺, 寬銀幕讓坐在前幾排的觀衆比以前就的窄銀幕更有 “參與

感”, 尤其是豐富的視角更滿足了觀衆“臨場感”。在 80 年代初,Sony 與 NHK 像電影界的製片推出了一整套叫 NHK Hi-vision 的 high-definition television 系統, 它是由 SONY 與 NHK 在 70 年代未發展出的一整套的電視製片系統, 從攝影、剪輯 、錄音、到特效的系統; 它的解析度幾乎與 35 厘米的影片一樣的細微。用這套系統, 影片可即拍即看, 立即剪輯並可立即轉換成 35 厘米的影片。這一來, 使得當時一些遲遲無法推出的影片得以在此系統的輔助下順利推出。此系統同時也可以在製作中加入一些特效, 這是以往用傳統影片拍攝方式所做不到的。在 Hi-vision 順利推入電影界後, 他們就開始著手於將此 Hi-vision 發展到商用廣播系統上; 此系統的解析度無論是水平解析度或垂直解析度都比傳統電視大上二或三倍。首先面對的問題就是如同 1954 年時黑白電視轉換成彩色點是的問題一樣,“如何相容於傳統的彩色電視系統呢?” 相容? 短時間內取而代之? 或並存呢? 相容或短時間內取而代之都是有前例可循的, 在 1957 年美國的彩色電視就採相容於黑白電視的方式; 採相容方式在當時的技術上的確產生了不少大大小小的干擾問題。在英國來說, 英國在 1936 年開播的黑白電視視 405 條掃描線的, 在 1967 年英國開始播送 625 條掃描線的 PAL 彩色電視, 在當時彩色與黑白電是系統並存了 5 年, 在 5 年以後 405 條的黑白電視便走入了歷史。

HDTV (High Definition TV) 主要的目的是不增加每一個單元領域內的定義, 而是要增加視覺範圍, 特別是寬度。HDTV電視的概念不僅在於提昇 5 ～ 8 倍的解析度而已, 而是除此之外, 是視角的增加由原先的4:3變成了16:9, 並增加了劇院效果的高傳眞杜比5.1聲道的傳送, 同時也要消除傳統電視所困擾的鬼影與雜訊等等影響畫質的可能因素。所以整個 HDTV 的傳送可以說是眞正的將電影院正式的帶入家庭, 讓家庭劇院不再只是個空洞且難以定義的名詞。

具有類比與數位技術的 HDTV , 它有 1125 條掃瞄線, 交錯掃瞄 (每秒60個範圍) , 寬高比爲 16:9 。傳統電視與 HDTV 最明顯的差別在於:

- 頻寬的限制: 提高了解析度, 相對的增加了頻寬, 以 1980 年代類比的技術來說, 電視的頻道必須相對的由 6MHZ 增加到 20MHZ, 以現有幾乎占了滿滿的頻道來說 (VHF 與 UHF), 就必須犧牲掉一半以上的頻道才得以滿足 HDTV 的要求。這個問題幾乎到了 20 世紀末數位化的 MPEG II 發表後才得以解決, 所以目前 DTV 與 HDTV 的影像與聲音都是採 MPEG II 數位傳送的。

- 廣播的方式: HDTV 支持者的看法大致分成二派, 而且爭議的非常激烈, 一派的人認爲 HDTV 要成功的撥出一定不是在現有的無線或有線頻道; 而幾乎相等的另一派則認爲 HDTV 必須使用現有的無線及有線頻道, 也就是 6MHZ 的頻寬撥出才會得到大多數的支持。在 1987 年美國的 FCC 發表了一項聲明, HDTV 的規格必須要相容於現有的 NTSC 系統。在 1990 年 FCC 又發表了 HDTV 系統將會與 NTSC 共存; 這

幾乎是推翻了前面的共容聲明, 也就是 HDTV 將會與 NTSC 共存一段時間後漸漸的由 HDTV 取代直到 NTSC 完全消失, 但有趣的是 FCC 只管到無線電視廣播, 無法管到有線電視的播放系統, 所以有線系統的業者可以自己決定自己 HDTV 或 DTV 所要播放的模式, 所以至今在美國有線業者所提供的 Set-Top Box (座上接收盒) 有可能不同於無線接收器, 也可能不同於其他有線系統所提供的 STB (Set-Top Box)。

- 交錯式掃描 (Interlaced) 與非交錯式掃描 (Non-Interlaced): 交錯與非交錯各有優缺點, 交錯式是低成本 . 低頻寬而且能提高解析度最好的選擇; 但根據實驗得知, 由於人眼在身理上與心理上的感受能力對交錯式掃描的解析度只有感受到相當於實際交錯掃描的掃描解析度的 70% 而已, 在動態畫面時, 其感受則降到只有相較於實際掃描條數 50% 的解析度而已。而且交錯式掃描容易產生閃爍及斜線的羽毛現象, 但有趣的是現在的 HDTV 四種解析模式 1080i . 720p . 480p 及 480i (P 代表 Progressive,I 代表 Interlaced) 就兩種掃描方式都含括了。

- 視頻訊號的壓縮: 在 1990 年日本 NHK 衛星播送的 Hi-vision 是採用類比與數位混合方式的 MUSE 系統, 其所需要的頻寬是 20MHZ, 但衛星的每一頻道的頻寬限制是 8.15MHZ, 所以它必須採用2個附加頻道才得以播送 20MHZ 的頻寬, 所以相對於 NTSC 系統的 6MHZ 頻寬, 視頻頻寬的進一步再壓縮是有必要的; 至於採用何種的壓縮方式在 1989 年至 1993 年間日美各大廠都個別提出了各種不同版本的建議案, 其中有類比數位混合的也有純數位的建議方案, 各方案間個別都有其優缺點, 所以也同時造成爭論不休的紛爭局面。

而 HDTV 主要的顯示模式如下圖 2.7 所示:

	水平解析度	銀幕比例	交錯式 / 非交錯式	每秒畫面數
720p	1280	16:9	非交錯式	24、30 或 60
1080i	1920	16:9	非交錯式	60
1080p	1920	16:9	非交錯式	24 或 30

Figure 2.7: 顯示模式

HDTV 的影像與聲音的壓縮決定採用已開發完成的 MPEG II, 因爲 MPEG II 已普及到了

電腦及多媒體的應用, 且符合 HDTV 將相容於 PC 的準則, 至於聲音部分則採用 Dolby (杜比) AC-3 的數位壓縮方式, 同時也包括 5.1 聲道的數位環繞音場。

2.6.5 視訊取樣

Chroma 部分取樣 (Subsampling) , 因爲人對於色彩相對於黑白有比較少的空間解析度, 所以將色彩訊號相對地大量消去是合理的。並且一個 Chroma 部分取樣機制 " 4:4:4 " 表示並沒有使用到 Chroma 部分取樣, 每個像素的 Y , Cb 及 Cr 值被送出, 每個 Y , Cb , Cr 都有 4 個。而 " 4:2:2 " 表示 Cb , Cr 的水平部分取樣 (Horizontal Subsampling) 比例爲 1/2 。也就是說如果將四個像素水平地分別標上 0 到 3 , 所有四個 Y 值會被送出, 而且每兩個 Cb 與 Cr 會被送出。" 4:1:1 " 水平部分取樣的比例爲 1/4 。" 4:2:0 " 是同時有水平與垂直的部分取樣, 而其比例爲 1/2 。通常 " 4:2:0 " 方式與其他方式常常用在 JPEG 與 MPEG 壓縮方式中。以 CCIT 601 格式爲例: Y(720 × 480)、Cb(720 × 480)、Cr(720 × 480) 在經過 4:2:0 的次取樣後分別爲 Y(720 × 480)、Cb(360 × 240)、Cr(360 × 240)。經過 4:2:2 的次取樣後格式爲 Y(720 × 480)、Cb(360 × 480)、Cr(360 × 480)。經過 4:4:4 的次取樣後格式爲 Y(720 × 480)、Cb(720 × 480)、Cr(720 × 480)。

2.6.6 視訊編碼中的冗餘

視訊編碼主要是藉由移除影片中的冗餘(Redundancy) 以減少資料量, 多數的視訊編碼都是屬於失眞 (Lossy) 的演算法, 也就是說是以視訊上品質的好壞做爲編碼的最終目標。失眞的壓縮演算法比較無失眞演算法其壓縮的倍率較高, 且碼率-失眞取捨 (Rate-Distortion Tradeoff) 的彈性較大。一般而言影片中的冗餘可以分爲以下幾種:

- 時間 (畫面間) 冗餘 (Temporal or Interframe Redundancy): 由於影片是由一個時間序列的影像所組成, 同一場景前後張畫面通常有相當高的相關性。
- 空間 (畫面內) 冗餘 (Spatial or Intraframe Redundancy): 一張畫面空間上鄰近的像素點通常具有相似的亮度或顏色。
- 編碼冗餘 (Coding Redundancy): 量化後符號機率並不是完全相等, 且符號間也存在著相關性。
- 感官冗餘 (Perceptual Redundancy): 人類視覺系統對於不同色或空間頻率的敏感度不盡相同, 某些圖像成分較不容易被人類視覺系統所察覺。

Chapter 3

視訊標準的演進

3.1 MPEG視訊標準

MPEG 其實剛開始是一個委員會, 它於1988年成立, 爲的是發展出一個可以在不同的位元率下, 將視訊與音訊一起放入數位儲存設備的標準演算法。原本這個委員會有三個工作團隊, 即 MPEG-1 、MPEG-2 、MPEG-3 , 分別將目標位元率放在 1.5 、10 及 40 百萬位元/秒。後來因爲 MPEG-2 演算法就足以提供 MPEG-3 的位元率, 因此 MPEG-3 就被丟棄了 。MPEG-1 工作團隊首先於1992年完成工作並制訂 ISO 標準 11172 , (**Coding of moving pictures and associated audio-for digital storage media at up to about 1.5 Mbits/s**) 。這個標準就被稱爲 MPEG-1 。在發展這個標準的過程中, 委員會覺得沒有必要再做數位儲存設備的限制, 因此 MPEG-2 工作團隊的研發方向轉向將 MPEG-1 擴充, 允許更有彈性的輸入格式及更高的位元率以配合高品質電視的需要。MPEG-2 工作團隊首先於1994年完成工作並且制訂了 ISO 標準 13818 , (**Generic coding of moving pictures and associated audio information**) 。這個標準就被稱爲 MPEG-2 。同年 MPEG 委員會開始著手制訂 MPEG-4 , 此標準的目標是用物件導向 (Object - Oriented) 的觀念來編碼多媒體資料。整個標準 ISO/IEC 14496 於1998年完成最後的工作並且1999年正式成爲國際標準。所有由 MPEG 委員會所訂定的標準部分都分成幾個部分發表, 例如 MPEG-1 包含六個部分 [1] 、MPEG-2 包含十一個部分等 [2]。其中視訊的部分定義了視訊訊號的編碼位元串及重建 MPEG 畫面的解碼器。

[1]MPEG-1 Systems 、MPEG-2 Video for CD 、MPEG-1 Audio 、Conformance 、Software 、Specification for IDCT implementation

[2]System 、Video 、Audio 、Conformance 、Technical Report 、DSM CC - Digital Storage Media Cmd and Cntl 、AAC - Advanced Audio Coding 、RTI - Real Time Interface 、Conformance Extensions 、IPMP on MPEG-2 Systems

3.1.1 MPEG-1

MPEG-1 是 ISO 第一項視訊壓縮演算法, 主要用來儲存和讀取數位媒體的電影和音訊, 如採用 SIF (352×240) 解析度、速率約爲 1.15Mbit/s 的 VCD 應用。MPEG-1 與 H.261 十分相似, 由於電影動態畫面遠超過普通視訊電話應用, 所以 MPEG-1 編碼器需要更強大的運算效能。MPEG-1 允許採用 B 圖框, 且可使用適應性感知量化, 對每個頻段使用不同量化比例, 以獲得最完美的視覺感受。MPEG-1是針對多目標的視訊壓縮而設計的, 它的應用範圍包括交談式多媒體應用、CD-ROM 之儲存、電影、KTV、購物等。它的輸入視訊先將 Y 、C_b、C_r 經過次取樣成爲 SIF 的格式: Y (360×240) 、C_bC_r (180×120) , 然後才做編碼。輸入視訊一般爲每秒30個畫面的訊號。在介紹 MPEG-1 系統之前有以下幾點需注意:

- 輸入視訊之資料格式不一定限制爲每秒30個畫面之 Y 、C_b 、C_r 。每秒24個畫面也可以被接受, 而每個畫面的解析度如 HDTV 之 1920×1080 同樣地也可以被接受。
- MPEG-1的目標位元率設定在1.5百萬位元/秒, 其中視訊大約佔1.2百萬位元/秒而音訊則大約佔250千位元/秒。
- 應用包括 VCD 與 MP3 。
- MPEG-1的架構與 H.261 類似, 主要的不同包括:
  - MPEG-1 的畫面尺寸比較有彈性, 可以大到 4096×4096 。
  - MPEG-1 的畫面速率有比較多的選擇。ex: 23.976、24 (電影)、25 (PAL)、29.97、30、50、59.94 及60 。
  - MPEG-1 採用雙向動作補償 (B 畫面),H.261則否 。
  - MPEG-1 提供更精確的半像素動作補償,H.261則否 。
  - MPEG-1 用 VLC 來編碼動作向量之差值。
- 在動作補償上與 H.263 有相當程度的類似, 包括前述的提供更精確的半像素動作補償、及兩者都是在 16×16 的灰度 (**Luminance**) 方塊中求動作向量 (彩度 , **Chrominance** , 上的動作向量則將灰度方塊中所求得之動作向量除以 2) 等, 但是 H.263 並不是用 VLC 來編碼動作向量之差值。
- H.261 與 H.263 雖然也是視訊標準, 但是主要的應用對象是視訊會議, 適合小幅度運動之視訊使用。

MPEG-1 也認識到, 在電視訊號裡眞正有重要影響性的不是掃瞄線的數目或是每秒鐘所使用的場畫面 (**Fields**) 數 , 而是類比域裡的訊號頻寬以及在數位域裡每秒鐘的像素數目。就記憶體的需求來說, 眞正有重要影響性的是一個畫面裡的總像素數目: 再就處裡速度的需求上來說, 眞正有重要影響性的巨方塊 (**Macroblock**) 的數目, 亦即編碼 16×16 像素的數目。MPEG-1 採用 3 種方式來壓縮一個畫面: I 畫面 (Intra Frame) 、P 畫面 (Predicted Frame) 、與 B 畫面 (Bi-Directional Frame) 。I畫面的編碼方式是採用類似於 JPEG DCT 的方式處理, 並不考慮與其他畫面間的關係, 所儲存的是一張完整的畫面。舉一個例子 : 每 15 張畫面後就有一張是 I 畫面 (這個數目可以因位元率之需求而改變) 。這是很必要的, 它保證每 15 張畫面中必然有一張完整的畫面, 因此即使中間有很大而且很突然的畫面改變使得動作補償編碼沒有辦法有效地預測 P 畫面 及 B 畫面 , 我們也還是可以很快地在下一個 I 畫面 恢復正常的放映。P畫面 是利用前面的 I 或 P 畫面 爲參考畫面, 畫面中不動的部分就不要儲存, 只儲存不一樣的部分, 其具體做法便是動作補償編碼。至於 B 畫面 , 它的原理和 P 畫面 一樣, 只不過 B 畫面 可以參考前面的畫面, 也可以參考後面的畫面。如下圖所示的一串 MPEG-1 畫面我們稱爲一個畫面群 (Group Of Pictures, 簡稱 GOP) , 每一個畫面群以 I 畫面 爲開頭, 然後每間隔幾個畫面就有一個 P 畫面 , 而在 P 畫面 與 P 畫面 (或 I 畫面) 間則穿插著 B 畫面。這些 I、P、B 畫面的排序與數目比例並不是固定的, 可以根據需要彈性地做選擇。

MPEG-1 編碼工作的前置處理, 包括將 RGB 的色彩空間轉換成 YCbCr 色彩空間、將交錯畫面轉換成非交錯畫面以及次取樣。接著編碼器需要給輸入畫面做編碼型態的選擇。I 畫面不需要動作估計或動作補償。且在 I 畫面中將畫面本身切割成 16×16 的方塊, 稱之爲巨方塊 (Macroblock)。每個巨方塊都是由四個 8×8 的 Y 方塊、一個 8×8 的 Cb 方塊、及一個 8×8 Cr 方塊所組成。其中 Cb 與 Cr 方塊都是由16×16 的解析度做次取樣所形成的 8×8 方塊。即爲 SIF 格式。之後則是將 8×8 的方塊去執行 DCT 轉換, 然後正常化、量化、再以串長編碼 (Run Length Coding) 或 Huffman 編碼方式去做進一步的壓縮。而量化後的方塊也需要去執行逆量化 ($Q^{-}1$)、做 IDCT 。也就是必須產生一張編碼過、解碼後的畫面。且將這張畫面存於畫面記憶體中供後面的編碼做預測編碼使用。如果輸入的是 P 畫面, 則需要儲存該畫面與參考畫面的不同處即可。P 畫面的編碼方式就是動作補償編碼法。對於目前所編碼的畫面的每個巨方塊, 利用動作估計器從過去畫面中的搜尋視窗找出最匹配的巨方塊, 將這兩個巨方塊相減所得到的差值 (誤差方塊), 以 DCT 編碼此誤差方塊。如果輸入的是 B 畫面, 則必須做兩次動作估計, 一次針對過去的畫面另一次則是針對未來的畫面, 因此便會產生兩個動作向量。接下來編碼器便可以利用這兩個最佳匹配的巨方塊或是它們的平均值算出預測誤差巨方塊, 此種方法稱爲畫面間內插編碼 (Interframe Interpolative Coding) 。

接著便經過 DCT 做編碼, 預測誤差的 DCT 量化係數以及動作向量便經過多工器並且使用 VLC 編碼。在壓縮效果上, 以 B 畫面的壓縮比最高, 其次是 P 畫面, 最後才是 I 畫面。

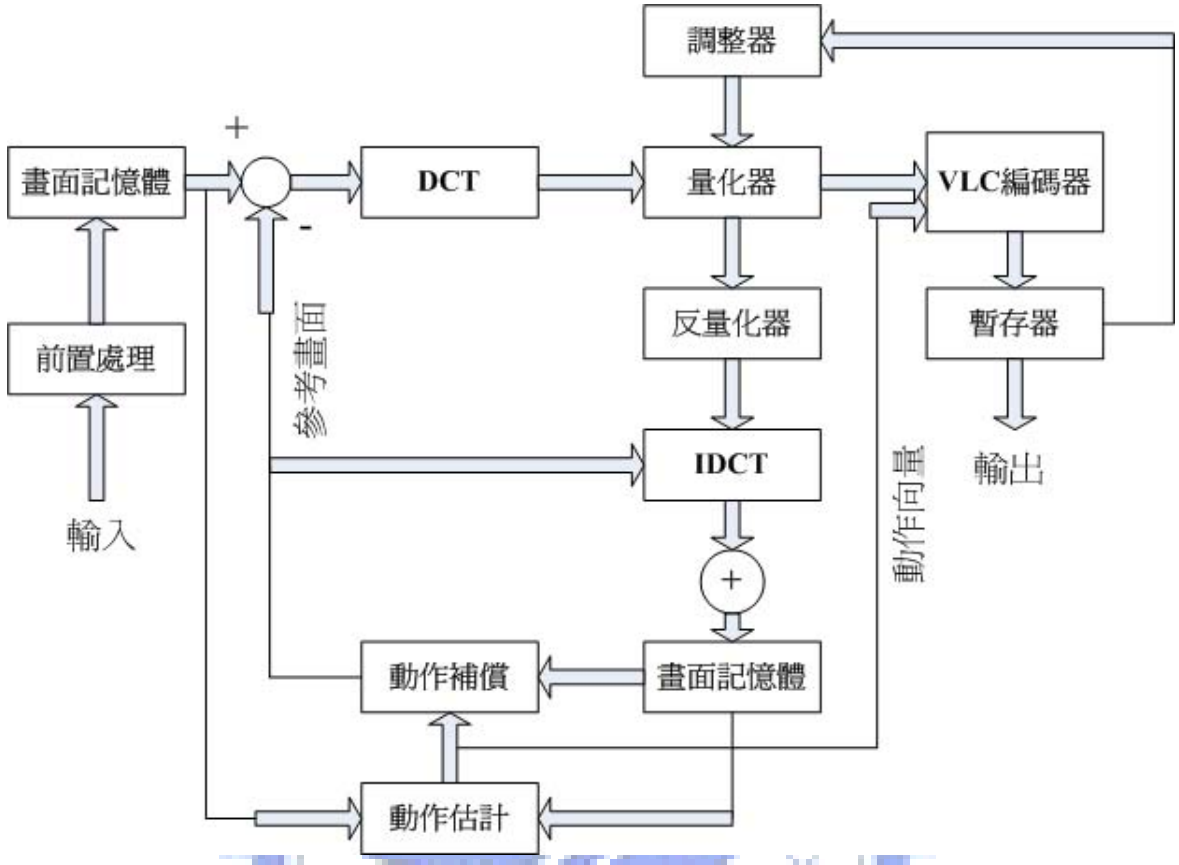

Figure 3.1: MPEG編碼器方塊圖

可想而知 I 畫面雖然壓縮比最差, 但是它卻是必要的, 因爲爲了要能隨機取出任何一個畫面。因爲 P 畫面與 B 畫面都必須藉助參考畫面才能解碼出來, 所以必須要找到離 P 畫面與 B 畫面最近的一個 I 畫面, 然後再依序解碼 I 畫面後的畫面。但是如果 I 畫面的間隔過大, 那麼取得一個隨機畫面的時間將會相當費時。另一個原因則是, 考慮資料會出錯的問題。由於只有 I 畫面才可以獨立的編碼, 而 P 畫面與 B 畫面都是直接或間接的被 I 畫面所決定, 所以只要畫面群中有任何一個畫面資料發生錯誤, 則所有參考到該 I 畫面的其他畫面都會出錯。而這個錯誤會傳遞到下一個新的 I 畫面出現爲止。所以 I 畫面越多壓縮比會越低, 但是相對而言隨機存取的速度則會加快, 且錯誤更正的速度也會比較快。

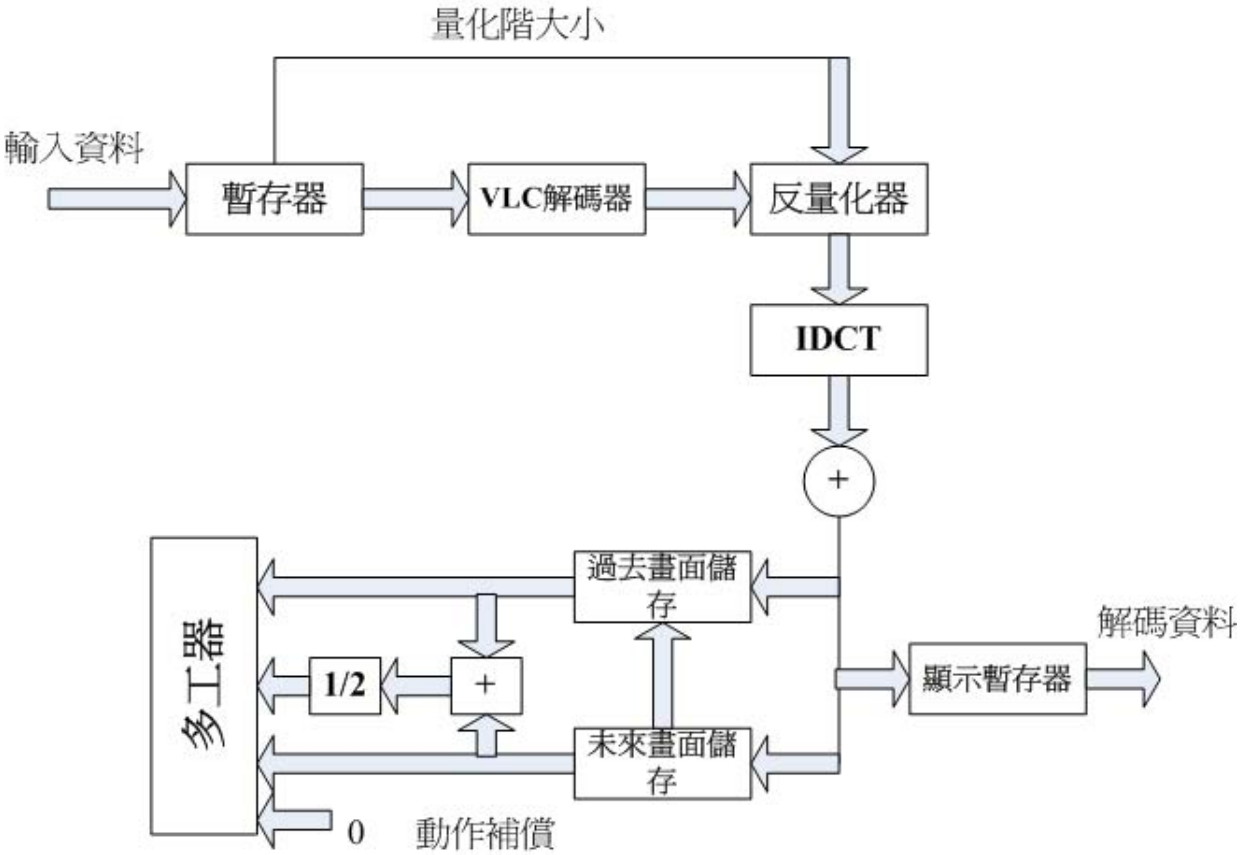

Figure 3.2: MPEG解碼器方塊圖

3.1.2 MPEG-2

MPEG-2 則是爲了數位電視所發展是定義在一個可以應用到各種不同應用的標準, 並將壓縮位元串定在5Mb/s 或是接近於10Mb/s。且必須包括幾個規格, 與 MPEG-1 相容、畫面品質好、輸入格式有彈性、可以隨機擷取畫面、有快轉與逆向播放及慢速播放的功能、位元串可調整、雙向傳輸時的延遲低與位元錯誤的免疫力。MPEG-2 的壓縮位元串大致可以分爲兩大類:

- 不可調整的格式: 此格式就是 MPEG-1 壓縮位元串格式再加上之援交錯畫面的編碼功能。
- 可調整格式: 此格式的壓縮位元串可允許編碼器選擇不同訊號品質等級。

MPEG-2 同時支援標準的循序掃瞄視訊, 視訊序列由一連串相隔固定時間擷取的圖框組成以及電視應用常見的交錯視訊。交錯視訊會把畫面分爲兩個圖場, 一個包含奇數行的畫素, 另一包含偶數行的畫素, 然後輪流擷取並顯示這兩個場。輸入格式可接受的最低解析度爲 352×240 以及主要專門爲高品質電視訊號所提供的高解析度 High 1440 (1440×1152) 與 High 1920 (1920×1080)。畫面速率都是制訂爲30畫面/秒 (29.97畫面/秒)。而主要輸入格

式又可分爲 4:2:0、4:2:2與4:4:4三種取樣方式。該方法特別適合電視顯示器。MPEG-2支援標準電視解析度, 包括美國和日本所用的 NTSC 標準, 每秒 60 個交錯式 720 × 480 圖場, 以及歐洲與其他國家的 PAL 標準。

MPEG-2 以 MPEG-1 爲基礎, 提供多種擴充功能, 以支援交錯視訊以及更大的移動補償範圍。高解析度視訊是一項重要應用, 因此 MPEG-2 提供比 MPEG-1 更爲寬廣的搜尋範圍, 這使移動估測所需的效能遠超過先前的標準。編碼器若要充分更爲寬廣的搜尋範圍和更高解析度, 就須超過 H.261 和 MPEG-1 的強大處理能力。MPEG-2 交錯編碼工具將同時支援圖場式和圖框式預測的移動補償最佳化, 可同時支援圖場式和圖框式 DCT/IDCT 變換。MPEG-2 在壓縮比 30:1 表現良好, 許多視訊應用也能接受速率 4 ∼ 8 Mbit/s 的 MPEG-2 視訊畫質, 可應用在數位衛星電視、數位有線電視、DVD 等。

	MPEG-1	**MPEG-2**
制訂日期	1992	1994
應用	VCD	數位電視
解析度	SIF (352x360)	576x720
畫面速率	25/30 frames/s	50/60 frames/s
位元率	1.5 Mbps	4 Mbps
畫面品質	類似 VHS	類似 NTSC/PAL
壓縮比	20~30	30~40

Figure 3.3: MPEG-1與 MPEG-2的比較

3.1.3 MPEG-4

MPEG-4 是由 ISO 所提出, 以延續 MPEG-2 技術架構。最初目標包括提高容錯能力來支援無線網路、低位元率應用以及各種新工具, 以便將圖形物件與視訊整合。但 MPEG-4 大部分繪圖功能仍未在產品獲得普遍採用, 實際應用也集中在低位元率壓縮能力。MPEG-4 主要目標定在三項功能: 以內容爲基礎的互動性 (Content-Based Interactivity)、普遍的存取、壓縮。

所謂的“以內容爲基礎的互動性”指的是 MPEG-4 將一個畫面視爲物件的組合, 而不是像素

或移動中的方塊組合, 其中的物件可以是車子、一段音樂、文字、任意形狀、可以是二維、也可以是三維的。不同的物件可以用不同的編碼方法做壓縮, 在解碼器有一個組合器可以將所有的物件重新組合成重建畫面。“普遍的存取”指的是 MPEG-4 適合在各種應用, 包括有線網路與無線網路、嚴重錯誤等常發生的情況。可以視情況彈性地調整畫面內容、品質、及複雜度。最後的“壓縮”指的則是 MPEG-4 的壓縮效率。在相同的位元率下 MPEG-4 可得到比之前的任何一個視訊編碼都還要好的視訊品質且它的位元率可高可低。MPEG-4 的輸入畫面可以是交錯畫面或非交錯畫面, 所提供的次取樣格式爲 4:2:0, 因此 Cb 與 Cr 的取樣點在水平方向與垂直方向都只有 Y 的一半。而位元率可以低到 5 ∼ 64Kb/s 也可以高到 20Mb/s , 不過在三個位元率下 MPEG-4 可以將整體表現調到最佳:

- 低於 64Kb/s
- 64Kb/s ∼ 384Kb/s
- 384Kb/s ∼ 4Mb/s

3.2 ITU-T 視訊標準

3.2.1 H.261

ITU 制訂的 H.261 是第一種視訊壓縮標準, 以雙向視訊會議應用爲主, 並針對速率從 40kbit/s 到 2Mbit/s 的 ISDN 網路而設計。H.261 支援 CIF (352 × 288) 及 176 × 144 (QCIF) 解析度, 色度解析度則採用 4:2:0 次取樣。視訊會議須同時進行即時編碼與解碼, 所以在設計時必須將複雜度降低。由於 H.261 具備延遲時間十分敏感的雙向視訊架構, 因此只允許使用 I 和 P 圖框, 不可使用 B 圖框。H.261 以區塊式餘弦變換 (DCT) 進行殘值編碼變換, DCT 把每個 8 × 8 畫素區塊反射到頻域, 以產生 64 個頻率分量。如將 DCT 數量化, H.261 會對所有交流係數進行固定線性量化。量化係數會進行變動長度編碼 (VLC) , 可把量化頻率係數視爲一個非零係數的絕對值, 後面是連續零係數個數和最後一個非零值後面的區塊結束代碼。

而 H.261與 MPEG-1 視訊標準演算法主要的不同爲:

- H.261只使用 I 及 P 巨方塊而不使用 B 巨方塊, 但是 MPEG-1 三種巨方塊都使用, 同時也使用 I、P 及 B 三種畫面

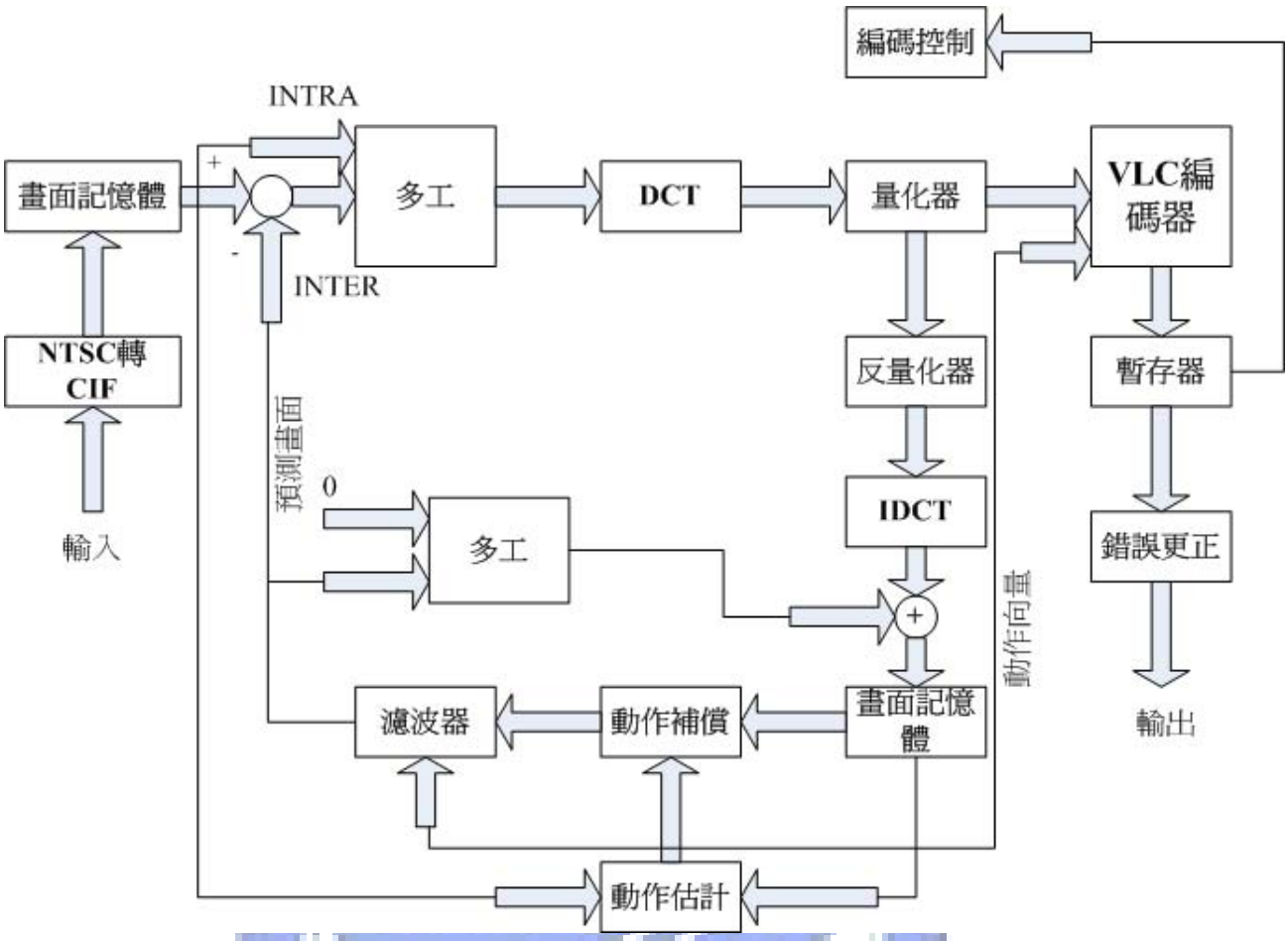

Figure 3.4: H.261編碼器方塊圖

- H.261有一項較特殊的限制, 就是每132個 P 方塊 (相當於三分之一張的 CIF 畫面) 至少要有一個 I 方塊。爲了讓 H.261在低位元率下有更好的的編碼效能, 大多數的 H.261 編碼器都會避免整張畫面的所有巨方塊都使用 I 巨方塊, 而只是循環的使用少數幾個。相較於 MPEG-1則是使用 GOP, 而且不限定 GOP 的大小。

- MPEG-1並不在意延遲方面的問題, 而 H.261則是非常在意。如果是互動式應用下,H.261的視訊編碼器及視訊解碼器的延遲必須事先知道, 這樣才能將音訊的延遲補足, 也才能維持同步。

- MPEG-1的動作補償可以精確到半像素的程度, 而 H.261只有要求到整數像素的精確度。

- H.261畫面使用固定的長寬比, 而 MPEG-1則允許好幾種長寬比。

- H.261的畫面率可以根據頻寬做調整, 最多可以跳過三張畫面。而 MPEG-1是使用 B 畫面的個數來調整位元率, 因爲 B 畫面的位元數是遠小於 P 畫面的位元數。

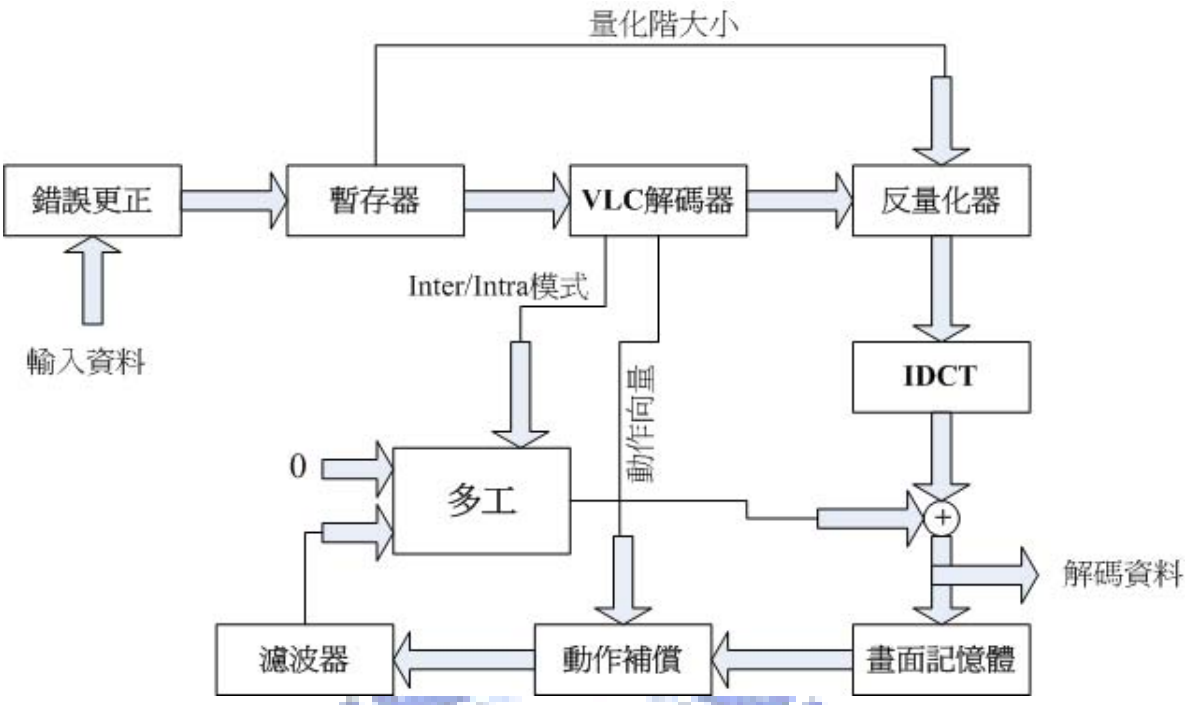

Figure 3.5: H.261解碼器方塊圖

3.2.2 H.263

H.263 是繼 H.261 之後所產生的標準, 提供更低位元率最佳畫質, 主要透過 28.8kbit/s 普通電話數據機傳送視訊, 解析度範圍從 SQCIF 到 CIF , H.263 基本技術則與 H.261 大同小異。H.263 在水平和垂直方向都能使用二分之一畫素移動向量, 還能透過內插計算提高參考圖框的解析度。該做法可得到更高移動補償精確度和壓縮比, 移動向量範圍也變得更大, H.263 還提供一組新選項, 包括:

- 4個移動向量, 一個區塊一個移動向量, 不是整個巨集區塊共用一個移動向量。
- 3D 可變長度編碼, 採用 Huffman 編碼, 可將區塊結束 (EOB) 指示碼與每一對執行層級結合在一起, 這項功能用於低位元率應用, 經常只有1或2個編碼係數。

由於 H.263 效率通常高於 H.261 並已成爲視訊會議最常用的演算法, 但這些技術也會繼續支援 H.261 , 以便與早期系統保持相容。H.263 又演進爲 H.263+ , 並增加許多可供選用的附件, 可支援更高的壓縮比, 以確保封包網路的可靠性, H.263 與其附件已成爲許多 MPEG-4 編碼工具的核心。H.263與 H.261雖然基本編碼器與解碼器架構是一樣的, 不過還是有一些不同處, 也就是說 H.263可以看成是 H.261的擴充。主要的擴充部分有下列幾點:

- 位元率:H.263沒有位元率的限制, 不過它的目標位元率是64Kb/s 或者更低。而 H.261 的目標位元率是 $p \times 64Kb/s$, 其中的 P 是1 $\sim$ 30 的整數。

- 畫面格式: 下表爲 H.263可支援的畫面格式。
- 方塊群結構:H.261與 H.263都將視訊分解成畫面、方塊群 (GOB)、巨方塊 (MB)、以及方塊。不過 H.263爲了顧及錯誤更正力, 設計成每一個 GOB 都只包含一個巨方塊列。所以對於 QCIF 格式而言, 每一個 GOB 只包含11個巨方塊。而相較於 H.261的一個 GOB 包含了 $11 \times 3 = 33$ 個巨方塊實在是少滿多的。
- 錯誤更正: 因爲 H.263主要應用是在 PSTN 或找者是行動通訊上, 所以發生的錯誤情形是一連串的錯誤 (burst errors), 因此 H.261所採用的 BCH 單一位元錯誤更正碼並不適用。不過 H.263並沒有對錯誤更正或者是偵測做嚴謹的規定。在 H.263的正式文件裡, 雖然有提供錯誤更正的能力, 但並不是硬性規定。
- 動作補償的精確度:H.261所採用的是整數像素精確度之動作估計, 而 H.263是採用跟 MPEG-1/2一樣的半像素精確度之動作估計。
- 濾波器:H.261在編碼迴路中使用了一個濾波器, 爲了是降低因爲動作估計所產生的方塊效應。H.263則是沒有使用濾波器, 主要理由是因爲 H.263在做半像素精確度之動作估計時就已經有低通濾波器的功能。
- 動作向量之編碼: 假設 MV 是我眞正的動作向量值而MV_p是預測值的話, 那麼編碼的差值就是 $MV_d = MV - MV_p$ 。H.261所使用的預測值 MV_p 是前一個巨方塊的動作向量, 而 H.263則是使用前三個編碼方塊之動作向量的中間值。

H.263與 H.261的不同處其目的都是爲了要降低位元串的負擔讓 H.263能夠更適合於超低位元率的應用。

Chapter 4

視訊標準 H.264/AVC 的介紹

回顧視訊編碼標準的制訂過程,MPEG 與 ITU-T 兩個委員會都是獨立的運作, 各自爲了自己的標準所努力。但是在1990年時兩個委員會共同制訂了 MPEG-2/H.262 標準, 是由 MPEG與 ITU-T 共同努力的成果, 如 Figure 4.1所示。這種情形在2001年十二月又再次的發生, 而這次則是制訂了視訊標準 H.26L, 其中的 L 代表長時段 (long term) 的意思。H.26L不再要求與之前的標準相容, 而是要求基本架構必須是通用的、最簡單的、最直接的方法。而 H.26L 在 ITU-T 被稱爲 H.26L, 而在 MPEG 則是將其納入 MPEG-4 變成 MPEG-

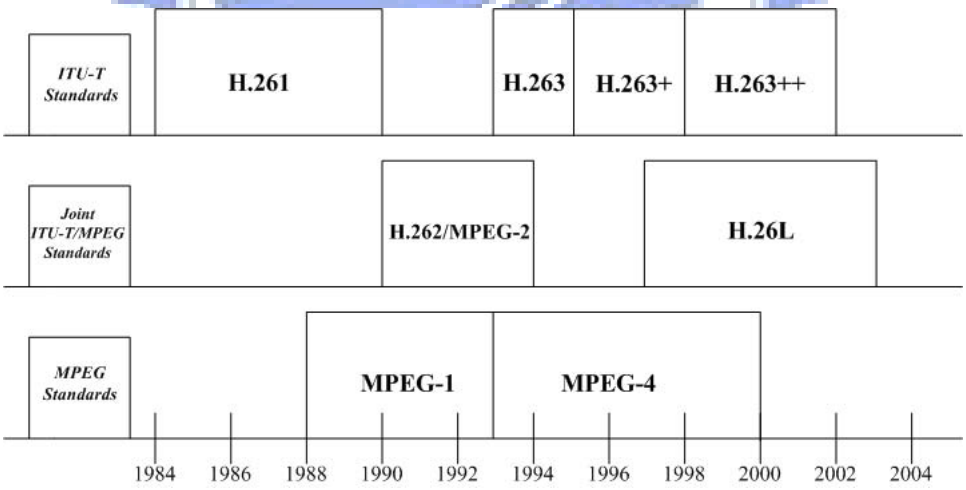

Figure 4.1: 視訊編碼制訂的時程

4 的 Part 10。H.26L整體的效能比 H.261及 H.263都來的好很多: 包括了更高的編碼效率、複雜度較低允許用軟體來實現、支援更多的功能、及更強的錯誤免疫力。潛在應用包括了網路視訊電話、即時聊天服務、及多點的通訊等等。更吸引人的地方是特別鎖定了第三代行動網路與廣播。H.26L已於2002年11月正式定案, 且命名爲 H.264/AVC。而其本身的演算法與先前的 H.261及 H.263有著很大的不同, 所以跟之前所有的 H.26X 標準都不相容。

H.264與之前的標準相比有幾個不同的特性, 也有跟現有的標準有相同的部分。而下列則是H.264主要的特徵:

- 高效能壓縮: 跟先前所制訂的標準比較起來, 最多可降低 50% 的位元率。
- 更好的錯誤免疫力: 針對封包遺失及行動通訊通道的損毀所產生的錯誤率可降低。
- 有彈性地使用於有延遲限制的應用: 針對即時交談服務可做到低延遲, 而對於儲存或者以伺服器爲基礎的 VOD 應用可允許更高的延遲以換得更高的壓縮效能。
- 簡單化與基礎的方法: 採用最基本的建構方塊, 且利用一般化、簡單化、及直接的設計方法。
- 編碼器與解碼器複雜度爲可調性: 編碼器與解碼器的複雜度不對稱, 編碼器可針對其運算量與視訊品質做相對應的調整。
- 高視訊品質的應用: 在高位元率時的視訊品質相當的棒。
- 網路的友善: 容易做封包、優先權控制、及應用於視訊串流服務等。

從演算法的角度來看,H.264/AVC 是跟之前的標準有著非常多類似的地方, 主要包括下列幾點:

- 將每一個視訊畫面切割成方塊, 每張畫面都利用方塊爲基本單位做處理。
- 利用視訊畫面內所存在的空間冗餘, 將原方塊做轉換、量化和熵編碼。
- 由於相鄰畫面的相關性非常高, 所以只需要編碼相鄰畫面的不同處即可, 而實際做法是利用動作估計與動作補償來完成。
- 利用編碼後殘餘值方塊找出視訊畫面上所存在的空間累贅。

H.264標準架構上包含了 VCL(Video Coding Layer) 與 NAL(Network Adaptation Layer) 兩層架構, 如 Figure 4.2所示。VCL爲視訊壓縮的部分, 其技術核心包括了動作估計、轉換編碼、預測編碼、去區塊效應濾波、及熵編碼等技術。而 NAL 則是提供 VCL 編碼資料與時計網路之間的連接, 其中必須進行編碼資料的格式化, 加入必須的標頭檔資訊, 封裝成適當的傳輸單元。H.264的 NAL 提供了多種的封裝方式, 以便可適用於各種不同的通訊傳輸協定。包括 H.320/H.324、MPEG-2傳輸串流、IP 網路、或 MP4檔案格式等等。NAL區隔 VCL 壓縮層與傳輸網路, 可以大大的簡化核心壓縮演算法社計時的考量。H.264在2002年定案時共制訂了 Baseline、Main 和 Extender 等3個層次, 其主要工具與應用如下:

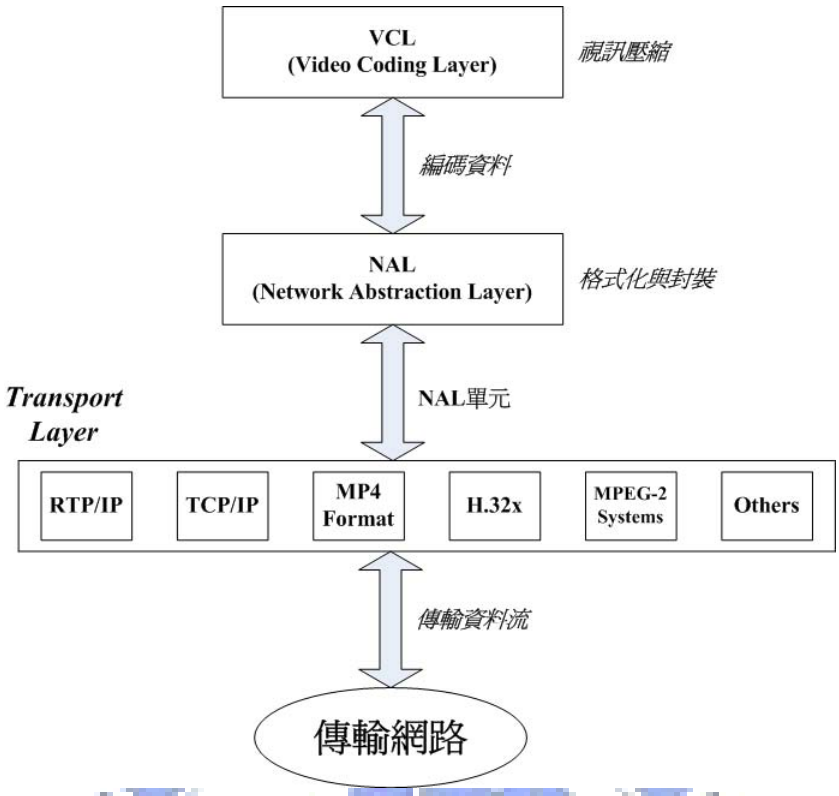

Figure 4.2: H.264的分層架構

- Baseline 層次: 此層次主要考慮複雜度要低但抗錯能力要強的行動視訊應用, 侷限於沒有 B 畫面的漸進式影像, 只使用 CAVLC; 提供了靈活巨方塊排序 (Flexible Macroblock Ordering;FMO)、任意畫面片段排序 (Arbitary Slice Ordering;ASO)、冗餘畫面片段 (Redundant Slices) 等抗錯功能。已知的應用包括了行動電話、行動電視、視訊電話等通訊服務。

- Main 層次: 此層次主要考慮壓縮效能要好但比較不在乎抗錯能力的視訊應用。相較於 Baseline 層次, 此層次加入了交錯式 (Interlaced) 影像、B 畫面、及 CABAC 等工具。所以壓縮效果會較好, 不過也相對的複雜度也提升了很多。目前的主要應用爲標準畫質數位電視 SDTV 及部分的儲存媒體中 (如 Sony 的 PlayStation Portable)。

- Extended 層次: 此層次包含了 Baseline 層次的所有功能, 另外還增加了資料分割與 SI 及 SP 畫面片段等錯誤回復的工具。

H.264/AVC於2005年針對了高畫質視訊的相關應用, 增加了 High、High10、High4:2:2、High4:4:4 等層次 (通稱爲 Fidelity Range Extensions;FRExt), 而這些層次基本上都是 Main 層次的擴充, 主要應用如下:

- High 層次: 此層次提高了 Main 層次使用位階的上限, 放寬採用 8×8 的 DCT 轉換,

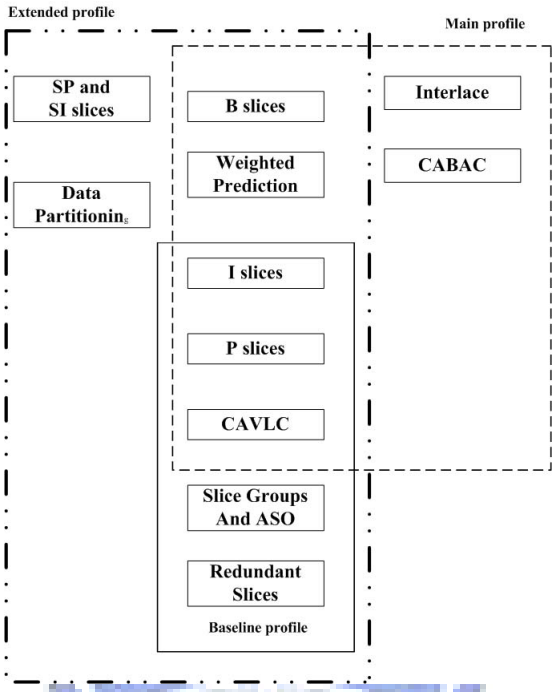

Figure 4.3: H.264的層次架構

並提供較好的量化位階調整模式。High 層次已經逐漸取代 Main 層次, 成爲數位電視地面廣播的標準, 目前包括 DVB-T 與日本 ISDB-T，均以採用 High 層次做爲高畫質數位電視視訊邊馬的規格。

- High 10層次: 此層次進一步放寬轉換係數量化時最多可使用 10 位元的精確度。
- High 4:2:2層次: 相較於 High10層次, 此層次進一步放寬顏色取樣至 4:2:2。
- High 4:4:4層次: 相較於 High10層次, 此層次進一步放寬顏色取樣至 4:4:4, 並放寬轉換係數量化時至12位元精確度。

4.1 H.264/AVC的核心架構

由下圖可得知 H.264/AVC 主要包括了六大部分, 本章節將針對六大部分一一的將每個部分拆解, 仔細的了解每個部分在 H.264/AVC 中扮演的角色爲何。

H.264與之前的 MPEG 標準有許多相類似的的部分, 如 H.264也是利用動作估計與動作補償去除時間上的冗餘, 利用 DCT 轉換與純量量化去除殘留剩餘上的冗餘, 用熵編碼去除編碼的冗餘。

新增的是 H.264/AVC 利用畫面內的編碼 (Intra Coding) 去除空間上的冗餘, 利用畫面間的編碼 (Inter Coding) 去除時間上的冗餘, 在轉換編碼方面將原本的 8×8 巨方塊切成更小的 4×4 巨方塊去做處理, 使得在進行轉換編碼時殘差影像的空間相關性較小, 且使用較小的區塊可以減少區塊效應的發生。在動作補償則是新增了“可變區塊大小 (Variable Block Size)”, 不再是像之前的演算法只是利用單獨尺寸的區塊。新增了不同尺寸區塊的選擇, 可依據目前處理的影片動作複雜度來做選擇。同時也新增了1/4像素的精確度與1/8的像素精確度, 藉以提高動作補償的準確度。

在熵編碼方面則是利用新的編碼方式 “內文適應性變動長度編碼”(Context-Adaptive Variable Length Coding ; CAVLC) 與“內文適應性算術編碼”(Context-Adaptive Binary Arithmetic Coding;CABAC) 更加增進了編碼的效能去除了編碼的冗餘。利用“迴路濾波器的架構”(In-Loop Deblocking Filitering), 使得在編碼方就可進行濾波, 去除區塊效應 (Blocking Artifact)[1]的發生。

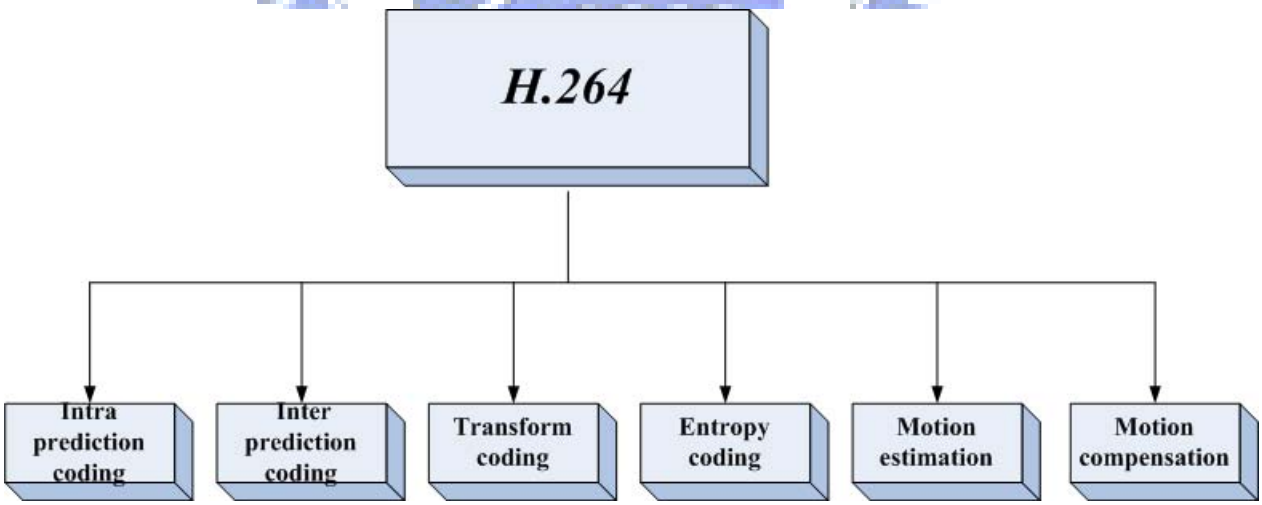

Figure 4.4: H.264/AVC的主架構

[1]區塊效應: 主要是因爲以區塊爲基本單位去做影像編碼時, 在失眞壓縮時所造成的邊界不連續現象。

4.1.1 動作估計(Motion Estimation) 與動作補償 (Motion Compensation)

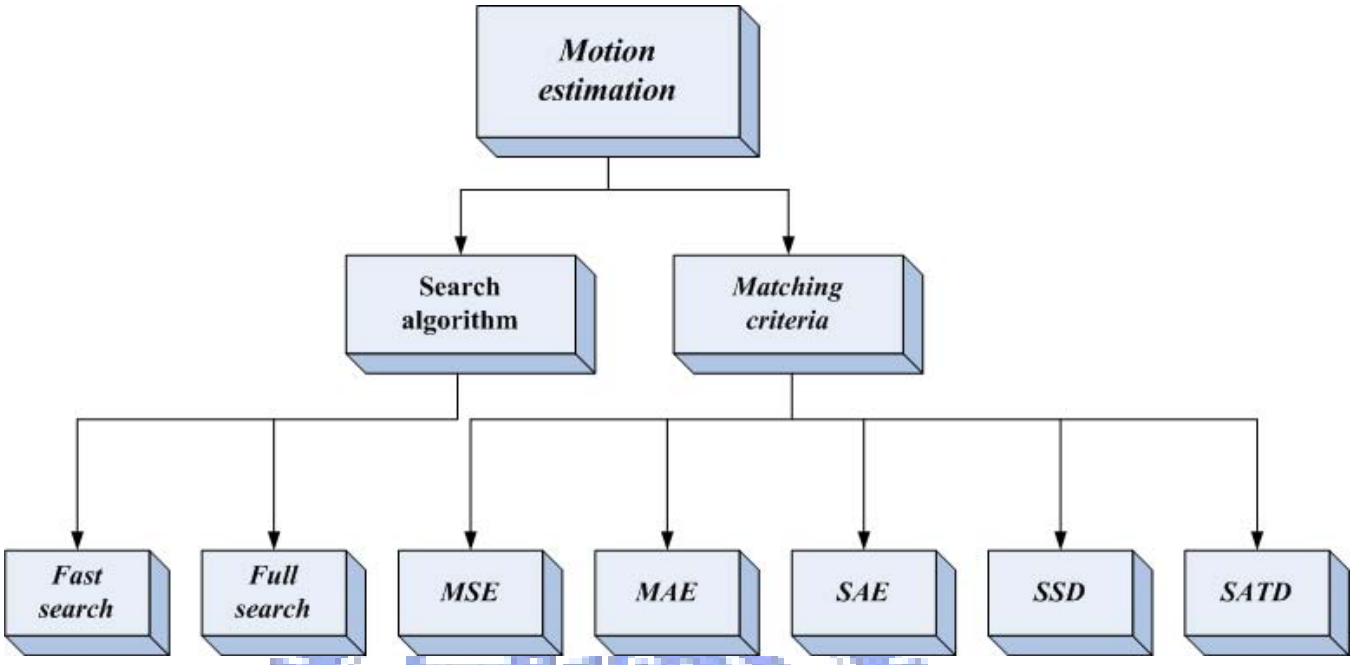

Figure 4.5: 動作估計的架構

由圖可得知動作估計主要包括了兩大部分, 一部份是 "比較準則" (Matching Criteria) 用來找尋最佳匹配的巨區塊, 一部份是 "搜尋演算法" (Search Algorithm) 搜尋圖像區塊位置的範圍。其中 Matching Criteria 常用爲下列幾種:

- Mean Square Error(MSE):

$$\frac{1}{N \times M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (C_{ij} - R_{ij})^2$$

- Mean Absolute Error(MAE):

$$\frac{1}{N \times M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \mid C_{ij} - R_{ij} \mid$$

- Sum of Absolute Error(SAE):

$$\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \mid C_{ij} - R_{ij} \mid$$

- Sum of Absolute Transform Differences(SATD):

$$\frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} H \times (C_{ij} - R_{ij}) \times H}{2}$$

其中 H 表示哈達碼轉換 (Hadamard Transform), 矩陣表示爲:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

- Sum of Squared Differences(SSD):

$$\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (C_{ij} - R_{ij})^2$$

上述所有的C_{ij}定義爲目前要編碼方塊的像素,R_{ij}定義爲參考畫面的方塊像素, 而 N 與 M 表示方塊的水平與垂直大小, i 與 j 則分別代表水平與垂直的像素點。

由以上的比較準則找到最小的比較數值, 找到的最小動作向量必須送到熵編碼, 如 Figure 4.6 所示, 在參考畫面中相對應與編碼畫面的方塊去尋找最佳的動作向量 ,且必須在搜尋視窗中的每一個方塊都必須要去比較, 藉由選擇最小的比較準則找到最小值的方塊即爲最佳的動作向量, 如 Figure 4.7 所示。而熵編碼會將此最佳的動作向量編碼後送出。 不過如果

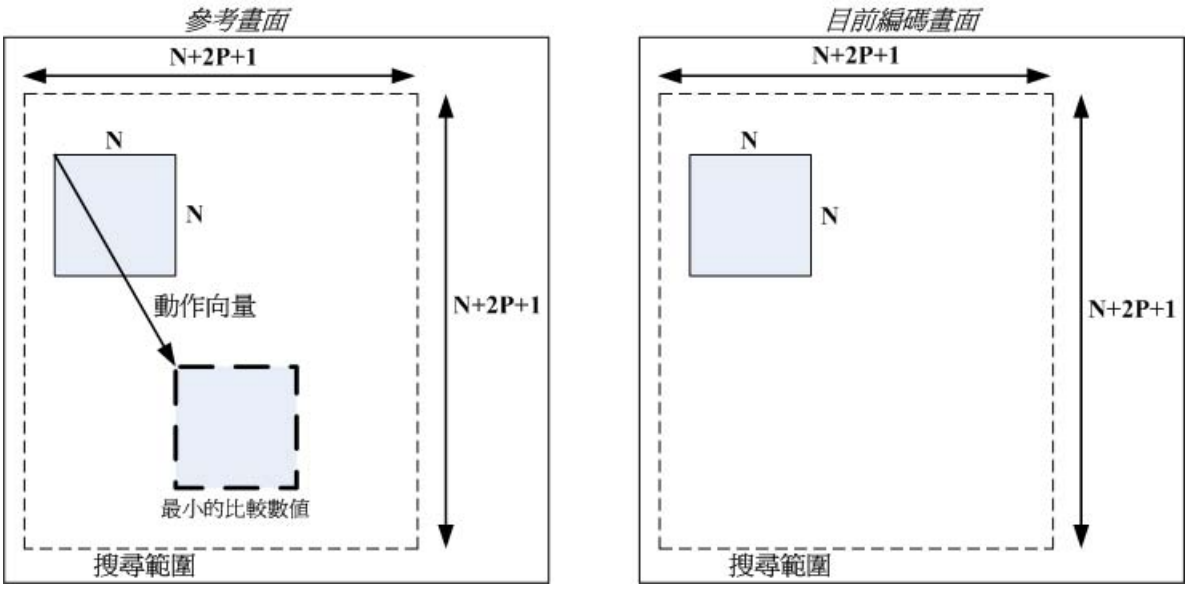

Figure 4.6: 動作估計示意圖

每一個方塊都要去比較的話, 也就是使用完全搜尋演算法 (Full Search), 這樣所耗費的計算量會非常的大, 但是卻保證可以找到最小的比較準則之值, 也就是最佳的動作向量。不過也相對付出了運算複雜度過高的代價, 所以必須研究出運算複雜度較低的搜尋演算法, 也就是快速搜尋演算法 (Fast Search)。

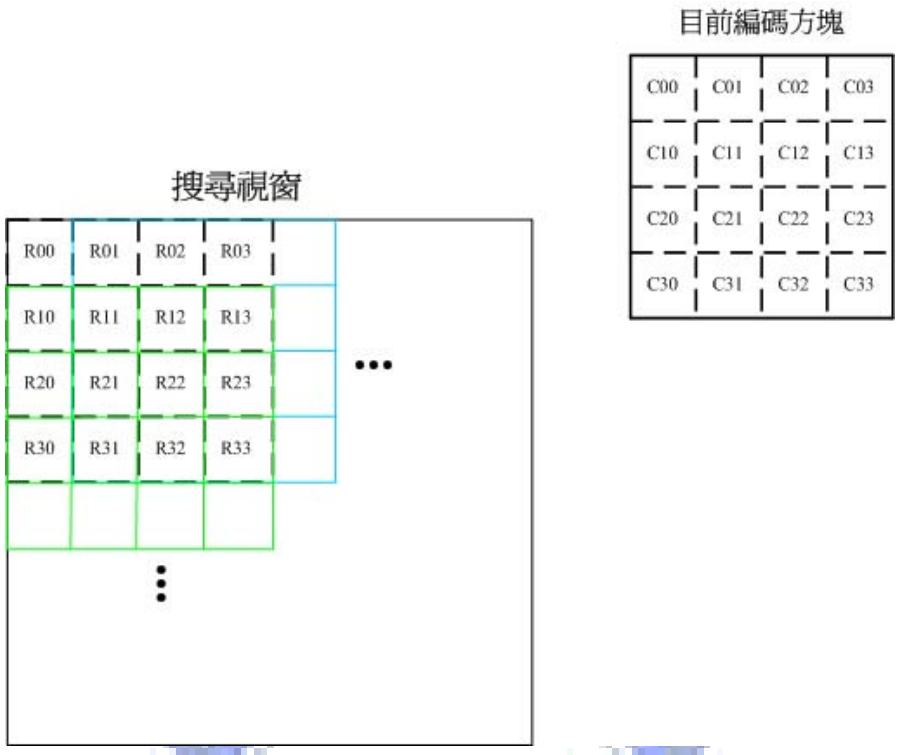

Figure 4.7: 動作估計搜尋視窗示意圖

雖然快速搜尋演算法不保證可以找到最小的比較準則之值, 不過卻可以降低整體的運算複雜度。雖然沒找到最佳的動作向量, 不過因爲人眼的視覺是相當的遲鈍, 就算沒找到最佳的動作向量, 只要不要失眞太多, 人眼是可以辨別出來的。所以可以利用快速演算法降低整體演算法的複雜度。

而一些常見的快速搜尋演算法有下列幾個種類:

- 第一種爲減少搜尋視窗中方塊的數目, 也就是只搜尋某些特定的方塊。
  - 3-step search algorithm (3SS)
  - 4-step search algorithm (4SS)
  - New 3-step search algorithm (N3SS)
  - Diamond search algorithm (DS)
  - Cross-search algorithm (CSA)
  - Conjugate Direction search algorithm (CDS)
  - Two-dimensional logarithm search algorithm (TDL) . . .
- 第二種是減少像素點數與每個方塊的比較數目。

  - Hierarchical search algorithm (HSA)
  - Partial distortion search algorithm (PDS)
  - Alternative sub-sampling search algorithm (ASSA)
  - Adjustable PDS (APDS) . . .
- 第三種是利用動態的搜尋視窗與可適性的門檻。
  - Dynamic search window algorithm (DSWA)
- 第四種則是混合型。
  - Motion vector field adaptive search technique (MVFAST)
  - Predictive motion vector field adaptive search technique (PMVFAST)
  - Unsymmetrical-cross multi-hexagon-grid search (UMHex)
  - Enhanced predictive zonal search (EPZS)

本篇僅就現有的快速演算法去做介紹, 並沒有深入的去研究其方法, 可能還有一些新的快速演算法, 本篇論文就不多做介紹。

這些演算法主要的目的爲了是要能夠降低完全搜尋演算法 (Full Search Algorithms) 的複雜度, 但是又要維持影像的品質, 所以這些演算法從一開始的減少搜尋視窗中搜尋的方塊數, 到減少像素點跟方塊比較個數, 到利用動態搜尋視窗, 到最後則是利用前人所做的方法整合出更好的演算法。所以動作估計的演算法有非常之多, 就看研究者如何天馬行空的去想像更新更快且維持影像品質的動作估計演算法。

一般來說如果要將每個像素的運動軌跡都描述出來, 只需要編碼送出第一個畫面與每一個像素的運動軌跡之資訊即可。而重建影像的話, 只需要將每個像素依自己的運動軌跡重建即可。所以實際作法是利用畫面與畫面之間每個方塊的運動向量來近似, 稱之爲動作向量 (Motion Vector)。而可利用動作向量幫助去做畫面間的動作補償預測編碼 (Motion Compensation Prediction), 而動作補償做的好壞將被動作估計所決定, 主要是取決於動作估計是否可以正確且快速的估計出物體的動作向量。動作補償的作法是以前一個畫面加上每一個方塊的動作向量建立出現在畫面的預測影像。將原畫面減去預測的影像便可以得到誤差影像, 之後將誤差影像送入編碼器去進行編碼的動作。如下圖所示: 所以一般來說, 果動作

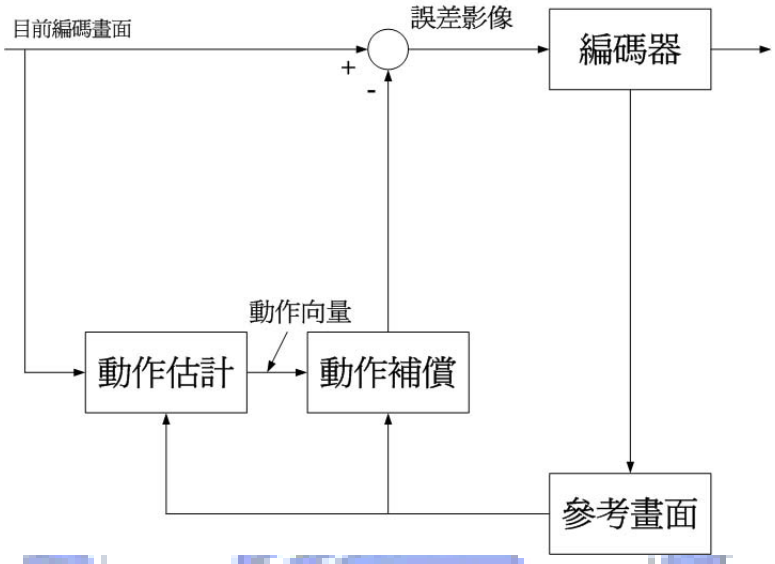

Figure 4.8: 動作補償方塊圖

補償編碼的成功與否關鍵在於動作估計, 只要動作估計做的夠好, 誤差影像便可以小到可以忽略。

4.1.2 畫面內的預測編碼(Intra Prediction Coding)

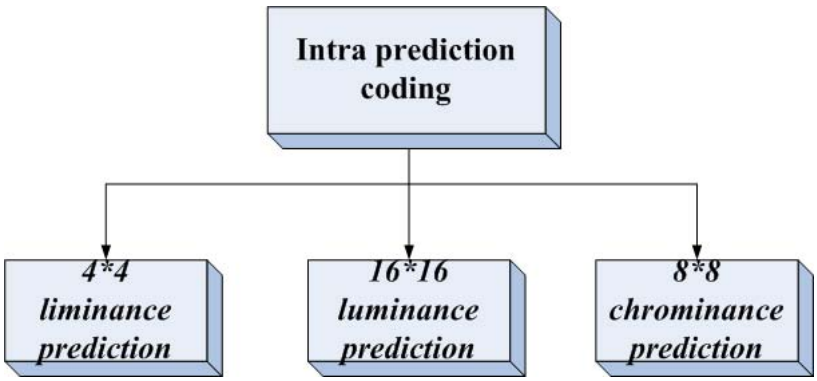

Figure 4.9: 畫面內預測編碼的架構

在畫面內預測編碼主要是利用每張畫面中像素的相關性, 利用鄰近已編碼過的方塊預測將要編碼的方塊, 如 Figure 4.9 所示, 利用 $A \sim M$ 像素去做編碼。而 H.264中主要是將亮

Figure 4.10: 利用鄰近方塊編碼

度 (Luma) 與彩度 (Chroma) 方塊分開來處理, 且針對亮度方塊有兩類方塊預測模式, 彩度則只有支援一類方塊預測模式。

首先將介紹 4×4 亮度預測模式, 在 H.264中此模式共有9種不同預測的方式。

第一種是垂直 (Vertical) 預測方式, 如 Figure 4.10 所示: 此預測方式, 主要是將目前要編碼

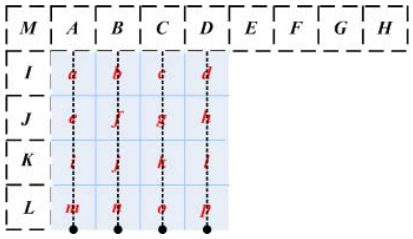

Figure 4.11: 垂直預測方式

的方塊像素 $a \sim m$ 預測成 A 像素的值、$b \sim n$ 預測成 B 像素的值、$c \sim o$ 預測成 C 像素的值, 最後則是將 $d \sim p$ 預測成 D 像素的值。

第二種是水平 (Horizontal) 預測方式, 如 Figure 4.11 所示: 此預測方式, 主要是將目前要

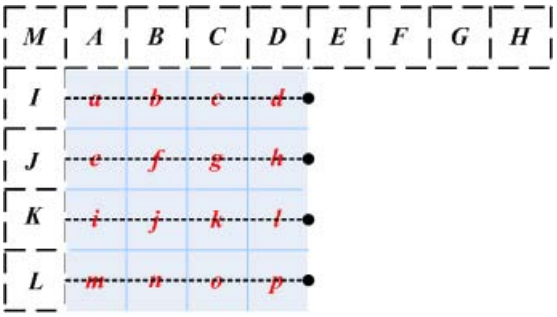

Figure 4.12: 水平預測方式

編碼的方塊像素 $a \sim d$ 預測成 I 像素的值、$e \sim h$ 預測成 J 像素的值、$i \sim l$ 預測成 K 像素的值, 最後是 $m \sim o$ 預測成 L 像素的值。

第三種是直流 (DC) 預測方式, 如 Figure 4.12 所示: 此種預測方式主要是將整個目前要編

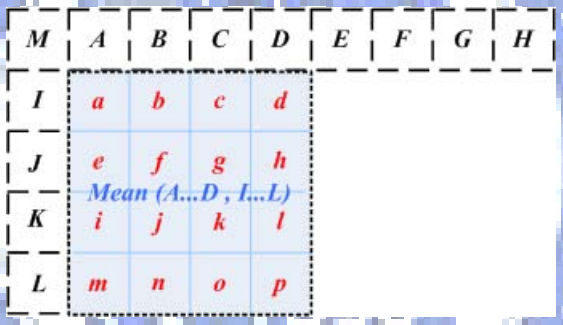

Figure 4.13: 直流預測方式

碼的方塊都預測成 $A \sim D$ 和 $I \sim L$ 像素值相加取平均的值。

第四種是對角下到左下 (Diagonal Down-Left) 預測方式, 如 Figure 4.13 所示:

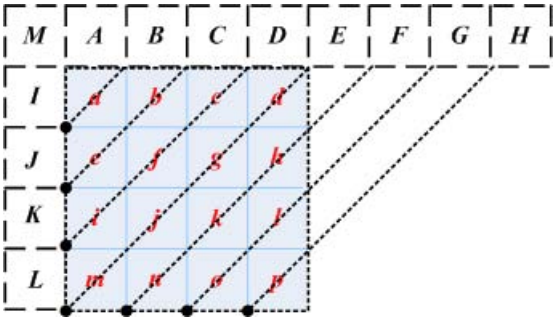

Figure 4.14: 對角右上到左下預測方式

第五種是對角下到右下 (Diagonal Down-Right) 預測方式, 如 Figure 4.14 所示:

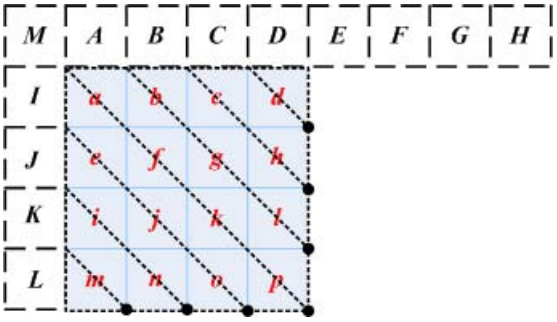

Figure 4.15: 對角下到右下預測方式

第六種是水平到下 (Horizontal-Down) 預測方式, 如 Figure 4.15 所示:

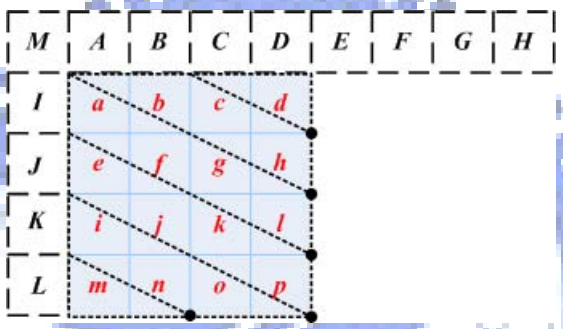

Figure 4.16: 水平下預測方式

第七種是垂直到左 (Vertical-Left) 預測方式, 如 Figure 4.16 所示:

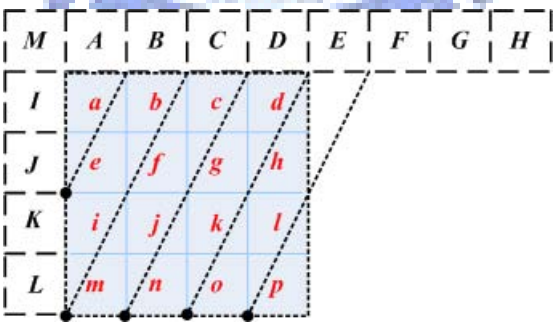

Figure 4.17: 垂直到左預測方式

第八種是垂直到右 (Vertical-Right) 預測方式, 如 Figure 4.17 所示:

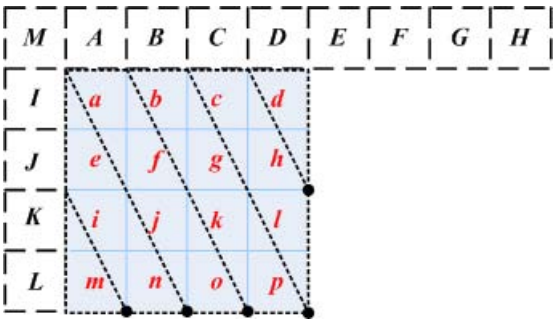

Figure 4.18: 垂直到右預測方式

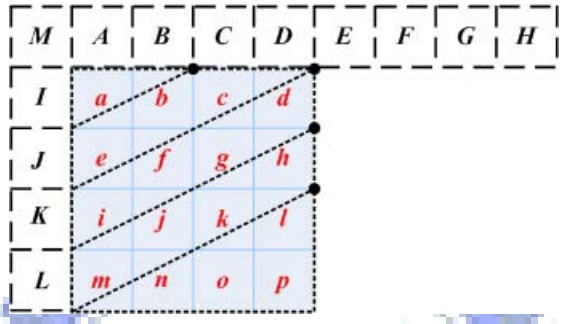

Figure 4.19: 水平到上預測方式

第九種是水平到上 (Horizontal-Up) 預測方式, 如 Figure 4.18 所示:

第二類則是 16×16 亮度預測模式, 此模式共有四種預測的方式。如下所示:

第一種是垂直 (Vertical) 預測方式, 如 Figure 4.19 所示:

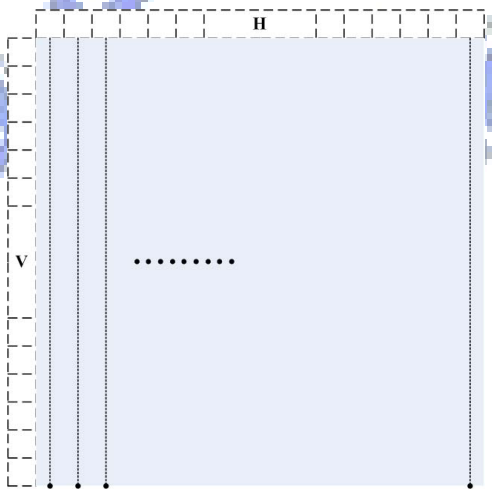

Figure 4.20: 垂直預測方式

第二種是水平 (Horizontal) 預測方式, 如 Figure 4.20 所示:

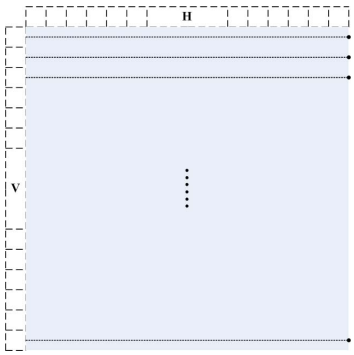

Figure 4.21: 水平預測方式

第三種是直流 (DC) 預測方式, 如 Figure 4.21 所示:

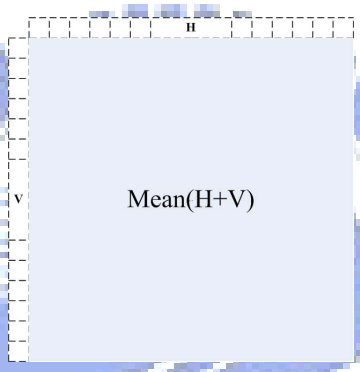

Figure 4.22: 直流預測方式

第四種是線性平面 (plane) 預測方式, 如 Figure 4.22 所示:

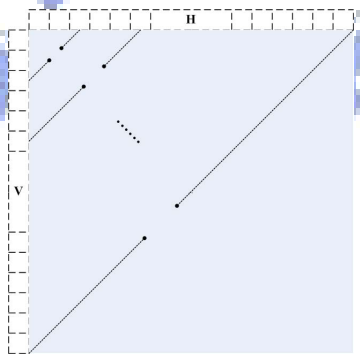

Figure 4.23: 線性平面預測方式

上述所介紹的是 H.264中所支援的亮度方塊預測方式, 而再來要介紹的是彩度預測方式, 大抵上都與 16×16 亮度預測方式差不多, 一樣都是提供了四種預測的方式, 包括了水平預測、直流預測、垂直預測與線性平面預測等方式, 唯一不一樣的地方是方塊的大小。在 H.264/AVC 中彩度所支援的是 8×8 的方塊大小。所以便不多加介紹彩度預測方式。

4.1.3 畫面間的預測編碼(Inter Prediction Coding)

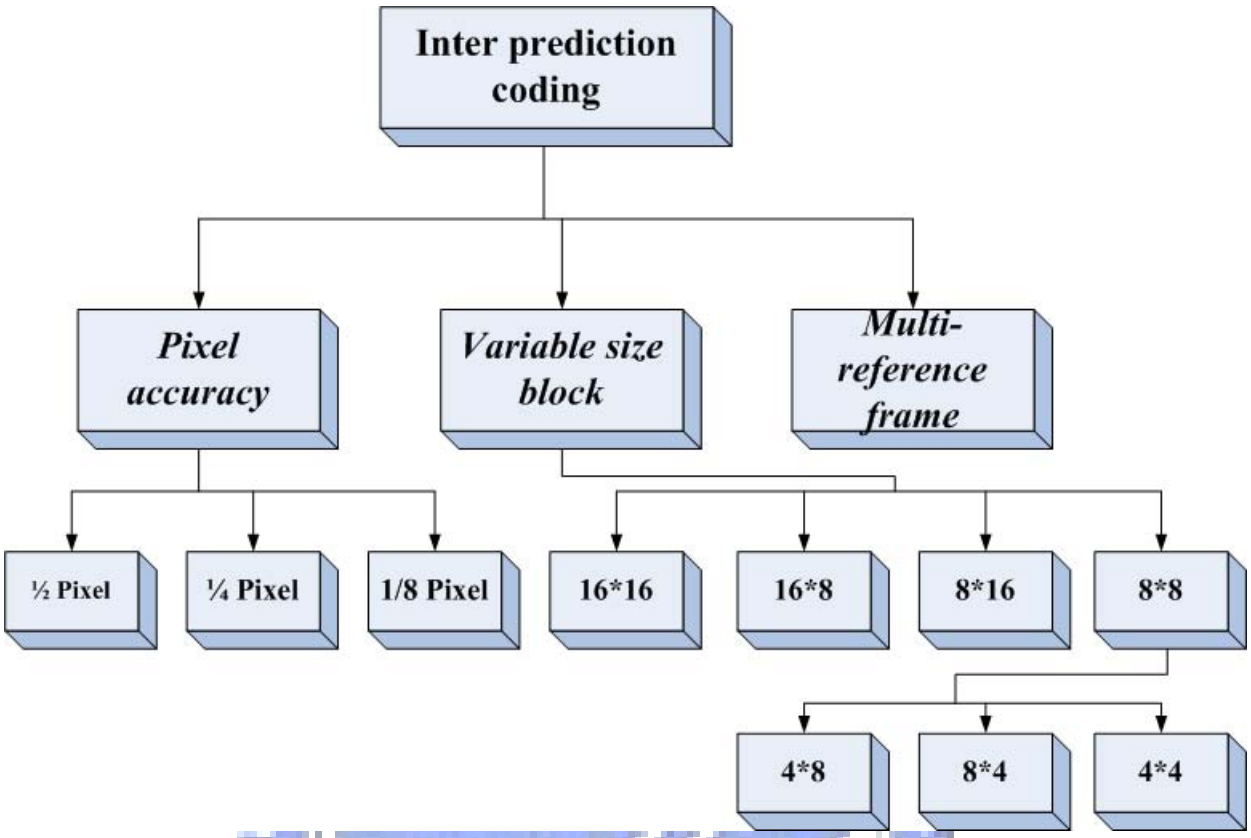

Figure 4.24: 畫面間預測編碼的架構

可變方塊大小與先前的視訊壓縮標準採用固定大小的差別在於,H.264/AVC 爲了增加比對的彈性並降低比對的誤差, 共提供了七種不同大小與外型的方塊可供選擇 ,如 Figure 4.25 所示。編碼器便可以依據影片的動作與材質的複雜程度, 彈性的選擇最佳的方塊類型。就如

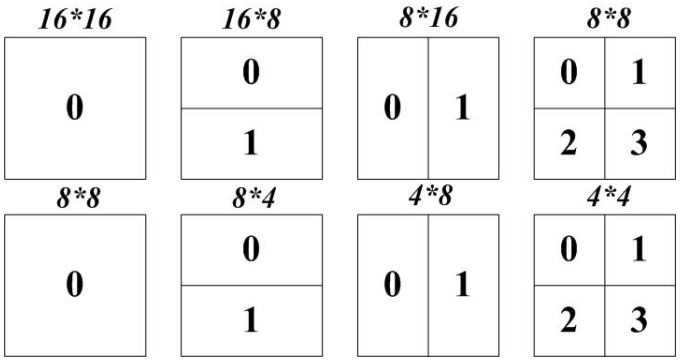

Figure 4.25: 可選擇的方塊類型

同我們可以在運動較單純或材質較平滑的區域使用較大的區塊, 以減少需要編碼的動作向量; 而在運動較爲複雜與材質較爲細膩的區域使用較小的方塊大小, 以增加比對的正確性。

使用多個參考畫面的原因在於, 有時候影像序列未必是相鄰者最相似, 所以 H.264/AVC 因此開放可保留更多張畫面當做爲參考畫面 (最多可以到31張), 另外 H.264/AVC 中的 B 畫面也可以當參考畫面。一般來說使用多個參考畫面可以得到較好的視訊品質及較有效能的編碼, 也可以提升 H.264/AVC 的錯誤免疫力, 不過從實作的角度來說, 使用多個參考畫面將會同時增加編碼器與解碼器的計算延遲與記憶體的需求。

先前的視訊壓縮標準動作向量最多精確到1/2像素, 但在 H.264/AVC 中提供了1/4像素的精確度搜尋甚至到1/8像素的精確度, 藉以提高動作補償的準確度。1/2與1/4像素的位置的值是由鄰近得整數像素位置內插得到的。一般1/2像素點的做法是利用一維6-栓的 FIR 濾波器內插得到, 而當中的係數爲 (1,-5,20,20,-5,1)。而1/4像素點則是由整數像素點與1/2像素點取平均得到。

$$b1 = (E - 5F + 20G + 20H - 5I + J)$$

$$h1 = (A - 5C + 20G + 20M - 5R + T)$$

$$b = (b1 + 16) \gg 5$$

$$h = (h1 + 16) \gg 5$$

$$a = (G + b + 1) \gg 1$$

$$e = (b + h + 1) \gg 1$$

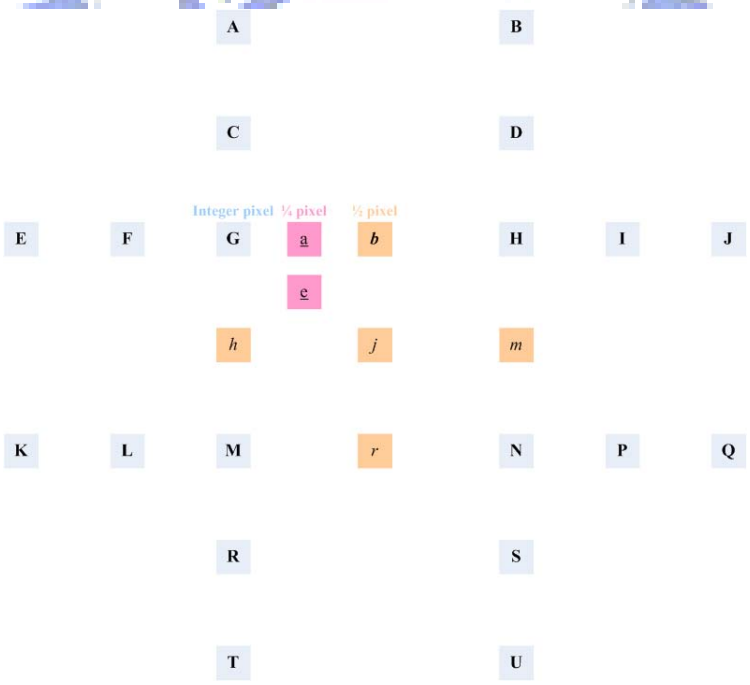

Figure 4.26: 分數像素點取樣

4.1.4 轉換編碼(Transform Coding)

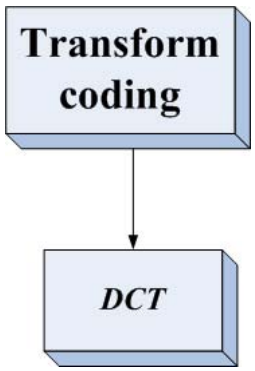

Figure 4.27: 轉換編碼的架構

一般影像在轉換時是將原本的 $N \times N$ 影像先切割成不重疊的 $n \times n$ 方塊, 之後再對每個 $n \times n$ 方塊去做單位轉換 (Unitary Transform)。轉換編碼的主要的目的是將本來應該在空間域 (Spatial Domain) 處理的影像, 將它轉到另一個域 (Domain) 去做處理, 將本來相關性很高的影像, 利用此轉換將影像的相關性降低且盡可能的將影像中的資訊集中於少數的轉換係數上, 使得之後在量化時可以用最少的係數去量化, 且之後可利用加入人類視覺系統 (Human Visual System ; HVS) 之對比敏感函數於轉換係數的量化之上, 以達到視覺上的無失眞壓縮。

一般來說希望利用此轉換可達到:

- 將影像內像素間彼此的相關性打散。
- 可獨立於影像的基底函數。
- 具有快速完成的特性。

而具有以上特性常見的轉換編碼有下列幾種:

- 離散傅立葉轉換 (Discrete Fourier Transform ; DFT)
- 哈達碼轉換 (Hadamaed Transform)
- 瓦氏轉換 (Walsh Transform)
- 離散餘弦轉換 (Discrete Cosine Transform ; DCT)

雖然常見的編碼有以上幾種, 但是眞正在影像中用的最多的是離散餘弦轉換 (DCT), 所以本文只針對離散餘弦轉換去做介紹, 至於其他轉換方式, 本文就不多加介紹, 留待給有興趣

的讀者自行閱讀。

其中理由爲 DCT 在資訊的集中能力與計算複雜度之間取得了很好的折衷點。DCT 主要的優點有: 已用單晶片電路實現、在係數最小的方塊中集中了最大的資訊量與區塊 (Blocking Artifact) 效應的現象最小。

而相對於 DFT 隱含了 n 點週期特性使得在邊界上產生不連續性, 以致於出現了高頻成分量。但 DCT 則減弱了此種現象, 主要因爲 DCT 隱含了2n 點的週期性, 所以本質上並不會產生邊界不連續現象。而一個 $n \times n$ 方塊二維的 DCT 正轉換定義如下:

$$F(u,v) = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) cos \frac{(2j+1)u\pi}{2n} cos \frac{(2k+1)v\pi}{2n}$$

而逆轉換則定義在下:

$$f(j,k) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} c(u)c(v)F(u,v) cos \frac{(2j+1)u\pi}{2n} cos \frac{(2k+1)v\pi}{2n}$$

其中

$$c(w) = \begin{cases} 1/\sqrt{2} & \text{若 } w = 0 \\ 1 & \text{若 } w = 1, 2, \ldots, n-1 \end{cases}$$

DCT快速演算法可以在$\emptyset(n \log n)$的時間內完成 n 點的轉換, 所以是目前最適合用於影像處理上的轉換。

4.1.5 熵編碼(Entropy Coding)

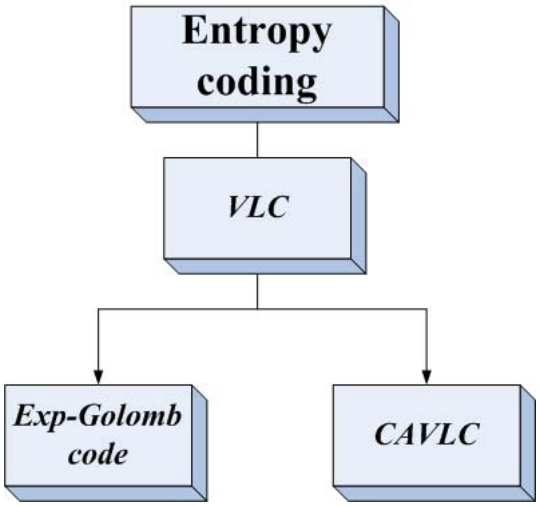

Figure 4.28: 熵編碼的架構

影像畫面的方塊經過了前面幾個章節的處理 (包括畫面內預測編碼、畫面間預測編碼與轉換編碼), 產生了殘留的影像像素。這些殘留的影像像素, 呈現了一些特性, 包括了低頻到高頻的大小呈現遞減的情形, 有許多影像像素大小爲零。所以這章節將利用這些殘留影像像素的特性去設計特別的熵編碼, 藉此將使用傳輸的位元數降至最低。

不過送到熵編碼去進行編碼的資訊包括了兩大類, 第一類是經過預測編碼轉換編碼, 最後到熵編碼的畫面方塊殘餘像素。第二類則是其它資訊的值, 包括了動作向量、使用的預測模式、參考畫面的索引號、量化的參數與在畫面中選擇的方塊型態。在 H.264 中第一類是使用 CAVLC(Context-Adaptive Variable Length Coding) 去進行編碼, 至於第二類則是使用 Exponential Golomb Code 去進行編碼。但是在進行熵編碼之前必須先將方塊內的數值掃瞄成序列, 常用的方法是使用 Z 掃瞄, 如 Figure 4.29 所示。掃瞄出的序列便會是$a \rightarrow b \rightarrow e \rightarrow i \rightarrow f \rightarrow c \rightarrow d \rightarrow g \rightarrow j \rightarrow m \rightarrow n \rightarrow k \rightarrow h \rightarrow l \rightarrow o \rightarrow p$

之後便可針對這些序列中的數值去進行編碼。一般來說 CAVLC 共可分成五個步驟:

- 第一步是找到序列內“非0”係數的個數與“正負1”的個數, 之後針對這些找到的個數去作相對應的編碼。
- 第二步是將這些正負1去作編碼。
- 第三步是針對非零係數去作編碼。
- 第四步去編碼整個序列中所有“0”的個數值。
- 第五步是去編碼序列中“0”的位置。

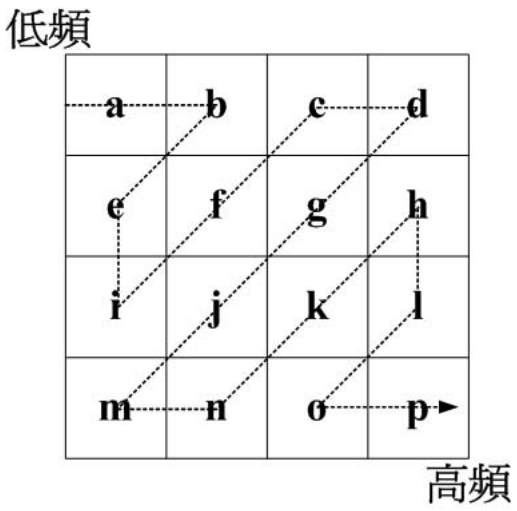

Figure 4.29: Z掃瞄

第一步首先要找到整個序列中“非0”係數的個數 (TotalCoeffs) 與序利中“正負1”的個數 (T1)。一般來說 TotalCoeffs 的可能範圍爲 $0 \sim 16$, 而 T1 可能的範圍則是 $0 \sim 3$, 不過如果 T1 的個數超過 “3” 的話, 則在此步編碼中便將 T1 設爲 “3” 。

在第一步的編碼中共有四個表格可查詢對照編碼, 表格的選擇將針對已經編碼過鄰近的方塊中非“0”的個數決定, 如 Figure 4.30 所示。當中的 “C” 表示目前正要編碼的方塊, “U” 與

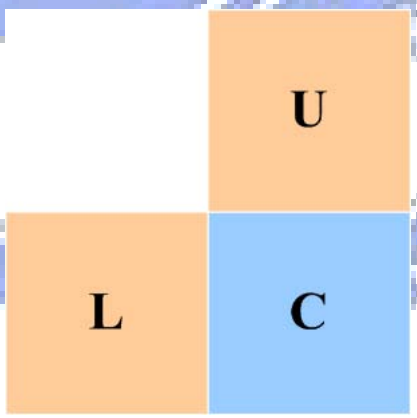

Figure 4.30: 鄰近的方塊

“L” 表示已經編碼過的鄰近方塊。

如果方塊 U 與 L 都可得到, 則

$$N = \frac{N_U + N_L}{2}$$

。如果只有方塊 U 可得到, 則

$$N = N_U$$

。如果只有方塊 L 可得到, 則

$$N = N_L$$

。最後如果方塊 U 與 L 都不可得到, 則

$$N = 0$$

。得到 N 的數值後, 便可利用查表得到使用的表格該是哪一個, 如 Figure 4.31 所示。所以

N	Table
0,1	*Num-VLC0*
2,3	*Num-VLC1*
4,5,6,7	*Num-VLC2*
8 or Others	*FLC*

Figure 4.31: Table 1

由以上介紹便可得知 CAVLC 有前文可適性的特性出現, 也就是可依據前面編碼過的方塊動態的調整目前要編碼的方塊。

第二步是針對序列中的"正負 1"去進行編碼, 不過編碼的順序是從序列中的高頻處往低頻處,"+1"編成"0"、"-1"編成"1"。

第三步則是針對序列中非零的係數去進行編碼, 直接將要編碼的數值進行編碼即可。

第四步是編碼整個序列中零的個數的數值。

第五步是編碼序列中係數不是零的係數 (run_before) 與編碼該係數左邊序列中零的個數 (ZeroLeft), 利用得到的兩個係數去進行編碼且一樣是從高頻處往低頻處去進行編碼。本文將在附件中舉一個例子讓讀著更了解此編碼方式。

在 H.264/AVC 的編碼器中會根據方塊的型態、預測的模式等特徵查詢相關表格找尋出各欄位的的索引號碼 (Index Number)。之後再根據索引號碼找出對應的 codeNum , 再利用 codeNum 查詢 Exponential Golomb Code 的表格找出對應的二進位碼 (Codeword), 即可完成其它資訊的熵編碼。所以編碼的過程順序爲索引號碼 → codeNum → Exponential Golomb Code 。Figure 4.31 爲 Exponential Golomb Code 與 codeNum 的對應關係。不過依照 H.264/AVC 中語法的定義, 索引號碼對應 codeNum 有三類的關係:

Exponential Golomb Code	codeNum
1	*0*
010	*1*
011	*2*
00100	*3*
00101	*4*
00110	*5*
00111	*6*
0001000	*7*
0001001	*8*
0001010	*9*
.....	.....

Figure 4.32: Table 2

- ue():Unsigned Exponential Golomb Code, 在這層關係下 codeNum 等於索引號碼。
- se():Signed Exponential Golomb Code, 這層關係是將有號的索引號對應到無號的 codeNum。
- me():Mapped Exponential Golomb Code, 此層關係是特別爲 Intra 4×4 或 Intra 方塊所定義的。

Chapter 5

處理的問題

5.1 移植系統過程所遇到的問題與解決問題

在一般將 H.264/AVC 系統移植到 CCS 開發平台上所會遇到的問題, 大概有下列幾項:

- 開放式程式碼的選擇
- 資料型態不相容的問題
- 函示庫不相容的問題
- 記憶體配置的問題

H.264/AVC 在開放式源碼的選擇上主要共有三種選擇, 分別是 JM code、X264 code 與 T264 code, 在此對這三種開放式源碼做個簡單的介紹。首先是 JM code , 此開放式源碼是 H.264/AVC 的官方測試碼, 用來提供個學術研究者在開發新演算法的一個比較基準, 但是因爲其編碼複雜度太高, 所以在系統移植上並不考慮此程式碼。

而 X264 code, 則是較注重實用與 JM code 相比的話, 可以在不降低編解碼性能下, 努力的降低編解碼的計算複雜度, 且 X264 code 摒棄了在 JM code 中一些計算複雜度極高的特性, 所以在實際系統移植上是可以考慮此程式碼。

至於最後的 T264 code , 其實它整個設計的出發點與 X264 code 非常相似, 不過它吸收了前面兩大開放式源碼的特點, 所以在某些層面上來說, 對系統移植上是一個較好的程式碼。所以本篇論文是利用 T264 code 當作是移植上的選擇。

資料型態不相容的問題, 主要是因爲 CCS 平台是用來模擬數位訊號處理板的各種問題, 藉由在 CCS 平台上將所有問題解決後, 可以順利的移植到數位訊號處理板上。而由於數位訊號處理板只有支援 32 位元的資料, 所以原先在 Win32 Visual C 環境下 64 位元的資料

便會出現問題, 所以必須將原先的 64 位元資料都替換成 32 位元資料後, 才可以順利的在 CCS 平台上執行該程式碼。而函示庫不相容的問題也是因爲 Win32 Visual C 環境與 CCS 平台環境的不同, 導致原本在 Win32 Visual C 環境上的函示庫移植到 CCS 平台上會出現不相容的問題, 所以必須用適當的函示庫替換掉, 才可以在 CCS 平台上順利的執行該程式碼。

而最後的記憶體配置的問題, 主要是因爲在 Win32 Visual C 環境上記憶體是不需要自行去配置的 (因爲只有一個記憶體 (RAM) 可使用), 但是在 CCS 平台上就不一樣了, 必須自行配置資料該放那個記憶體與程式該放那個記憶體, 因爲在 CCS 平台上資料與程式是被分別放在不同記憶體當中, 且必須自行撰寫組譯命令檔, 而組譯命令檔主要包含了組譯命令列、記憶體配置與區塊規劃。如果有需要的話還需要配置快取記憶體的使用, 用來加快資料與程式的存取速度。

如果需要做最佳化的話需要配置更多不同的記憶體位址的使用情形, 用來驅動數位訊號處理板周邊的介面來幫助加快程式執行的速度。

5.2 快速估計演算法的簡單介紹

5.2.1 三步驟搜尋演算法 (Three Step Search Algorithm;TSS)

三步驟搜尋演算法, 如 Figure 5.1 所示:

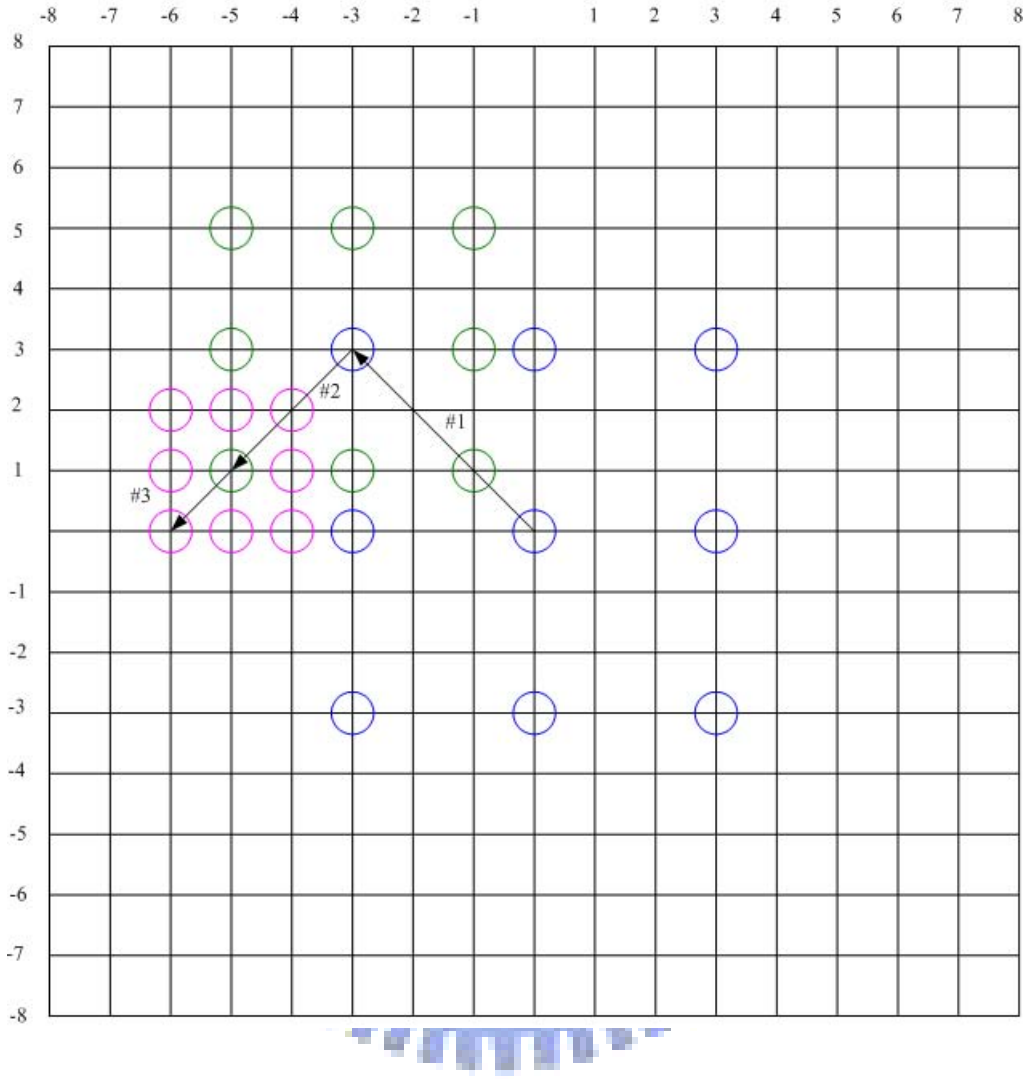

Figure 5.1: 三步驟搜尋演算法示意圖

一開始第一步需要比對的方塊爲九個方塊位置 (藍色圈圈所示), 在此九個位置中找到最小的比較數值 (其中每個位置的間隔爲3), 作爲是第二步驟一開始的搜尋位置, 以此位置間隔爲2擴展成類似 4×4 的方塊 (綠色圈圈所示), 一樣共九個位置要去比對找到最小的比數值, 作爲是第三步驟一開始的搜尋位置, 以此位置間隔爲1擴展成類似 2×2 的方塊 (紫色圈圈所示), 在此九個位置中找到最小的比對位置, 當作是最後的動作向量。

5.2.2 新三步驟搜尋演算法 (New Three Step Search Algorithm;NTSS)

新三步驟搜尋法, 如 Figure 5.2 所示:

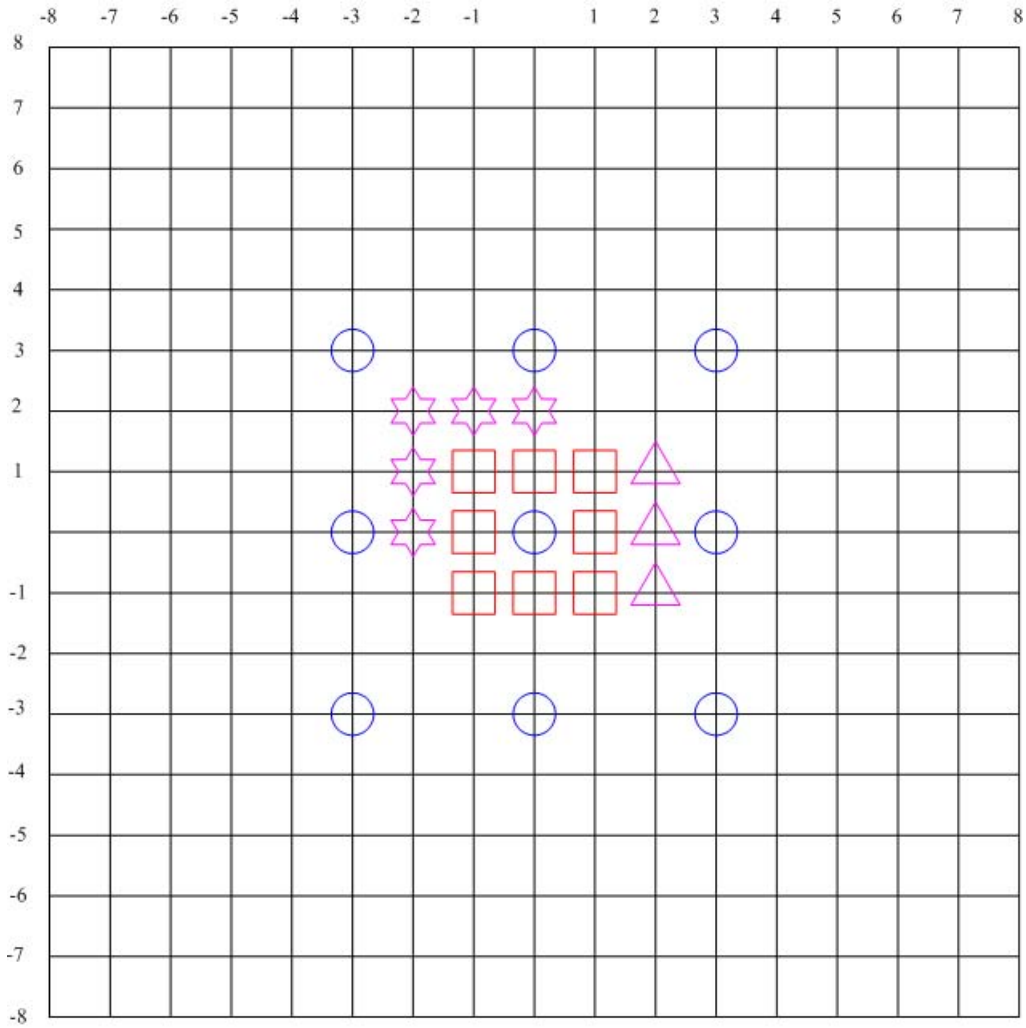

Figure 5.2: 新三步驟搜尋演算法示意圖

其實新三步驟搜尋法與原本的三步驟搜尋法非常類似, 只是一開始新三步驟搜尋法從原本的搜尋九個方塊位置增加爲搜尋17個方塊位置 (多增加的方塊位置爲標示爲正方形的八個方塊位置) , 所以共有17個方塊位置需要去比對數值。

如果最小的比對數值落在中心點的話, 其最後動作向量便是 (0,0) 這個位置。

如果比對的最小數值是落在八個正方形位置的四個角落位置的話, 便需要進行第二次比對, 第二次比對共增加了五個方塊位置 (星形位置), 所以第二次比對的方塊位置爲九個方塊位置 (加原本已經比對過的四個位置) , 可看成是一個 3×3 的方塊, 在此九個位置中比對出最小的數值, 當作是最後動作向量。

如果比對的最小數值是落在八個正方形位置的另四個位置的話, 便需要進行第二次比對, 第二次比對共增加了三個方塊位置 (三角形位置), 所以第二次比對的方塊位置也是九個方

塊位置 (加原本已經比對過的六個位置) , 一樣可看成是一個 3×3 的方塊, 在此九個位置中比對出最小的數值, 當作是最後的動作向量。

如果比對的最小數值是落在九個圓形位置中的外面八個位置的話, 即操作的步驟跟原本的三步驟搜尋法一樣。

5.2.3 四步驟搜尋演算法 (Four Step Search Algorithm;FSS)

四步驟搜尋法, 如 Figure 5.3 所示:

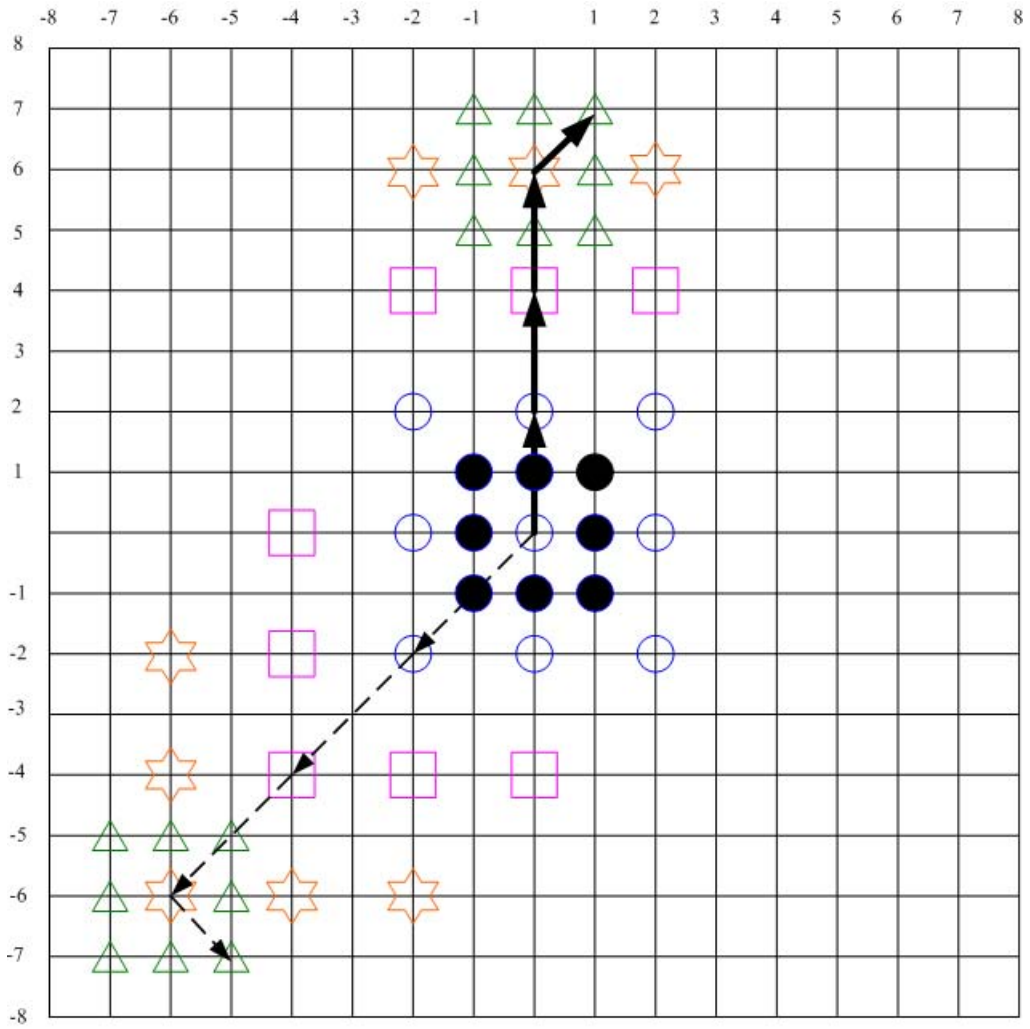

Figure 5.3: 四步驟搜尋演算法示意圖

一開始比對的方塊爲圓形的九個位置, 如果比對的最小值爲中間圓形的位置, 便在周遭新增加八個比對位置 (如圓形黑點的位置), 在這九個位置中 (包含原點的位置) 找到比對數值最小的值, 當作是最後的動作向量。

(虛線箭頭的方向) 如果搜尋的位置是九個圓形位置中的四個角落位置的話, 則需新增五個比對位置 (如正方形的位置), 以上步驟需重複兩次。直到第四次比對位置時需新增如三角

形所示的八個位置, 在此九個位置中去比對找到數值最小的位置, 當作是最後的動作向量。

(實線箭頭的方向) 如果新增的位置落在其他四個位置的話, 則需新增三個比對位置 (如正方形的位置), 以上步驟一樣需重複兩次。直到第四次比對位置的時候需新增加三角形所示的八個位置, 當作是最後比對的位置, 在此九個位置中比對找到數值最小的值, 當作是最後的動作向量。

5.3 演算法的比較

利用之前視訊標準所採用的快速搜尋演算法(也就是上一章節所提的演算法), 將它們用於H.264之中且實現於 DSP 平台之上去討論視訊品質的優劣的優劣與計算複雜度的高低。

首先先介紹所使用的開放式源碼與各個測試序列的模擬條件。

T264 開放式源碼	
QCIF(176x144)	
測試序列	模擬條件
Akiyo	搜尋範圍 = 30
Bridge_close	搜尋範圍 = 4
Coastgiard	搜尋範圍 = 65
Claire	搜尋範圍 = 16
foreman	搜尋範圍 = 30
編碼的畫面 = 10	
比較準則 = SAD	
畫面內預測編碼 = 4x4	
參考畫面 = 2	
半像素精確度	
DCT	
CAVLC	

Figure 5.4: 模擬環境介紹與條件設定

一開始先針對三步驟演算法與原本開放式源碼現存的鑽石演算法進行複雜度的比較, 如 Figre 5.5 所示爲三步驟演算法在編碼時實際的時脈數與實際在 DSP 平台上計算所需的時間,

三步驟搜尋演算法(TSS)		
測試序列	時脈數	實際時間
akiyo	87771117430	～145.5857 sec
bridge	83679618757	～139.7449sec
coastguard	108421106548	～181.0632sec
claire	90972259542	～151.923sec
foreman	111511966611	～186.2249sec

Figure 5.5: 編碼端三步驟演算法之計算複雜度

Figre 5.6 所示爲三步驟演算法在解碼端實際的時脈數與實際在 DSP 平台上計算所需的時間, Fiure 5.7 所示爲鑽石演算法在編碼端實際的時脈數與實際在 DSP 平台上計算所需的

三步驟搜尋演算法(TSS)		
測試序列	時脈數	實際時間
akiyo	8482754917	～14.166 sec
bridge	8643688166	～14.4349sec
coastguard	9344876685	～15.60594sec
claire	8666679312	～14.47335sec
foreman	9435315370	～15.75697sec

Figure 5.6: 解碼端三步驟演算法之計算複雜度

時間, Figure 5.8 所示爲鑽石演算法在解碼端實際的時脈數與實際在 DSP 平台上計算所需的時間,

鑽石搜尋演算法(Diamond)		
測試序列	時脈數	實際時間
akiyo	78058482992	～130.3576sec
bridge	74462251868	～124.3519sec
coastguard	86038298904	～143.6839sec
claire	78718105171	～131.459sec
foreman	93021613007	～155.346sec

Figure 5.7: 編碼端鑽石演算法之計算複雜度

鑽石搜尋演算法(Diamond)		
測試序列	時脈數	實際時間
akiyo	8468640402	～14.142sec
bridge	8631539823	～14.4146sec
coastguard	9282411594	～15.5016sec
claire	8666325389	～14.4727sec
foreman	9333470941	～15.5868sec

Figure 5.8: 解碼端鑽石演算法之計算複雜度

雖然在三步驟演算法在計算複雜度的比較上有明顯增加, 但是實際於 DSP 平台上所需的計算時間, 除了測試序列 foreman 編碼一張畫面差2秒以上 其餘在編碼端平均一張畫面差不到了2秒。因爲我們是在使用於 DSP 平台上的關係, 所以這些計算複雜度其實並沒有增加太多。在這邊我們試著測試五種不同的序列去觀察 PSNR 的變化與編碼端壓縮的資料的長度。

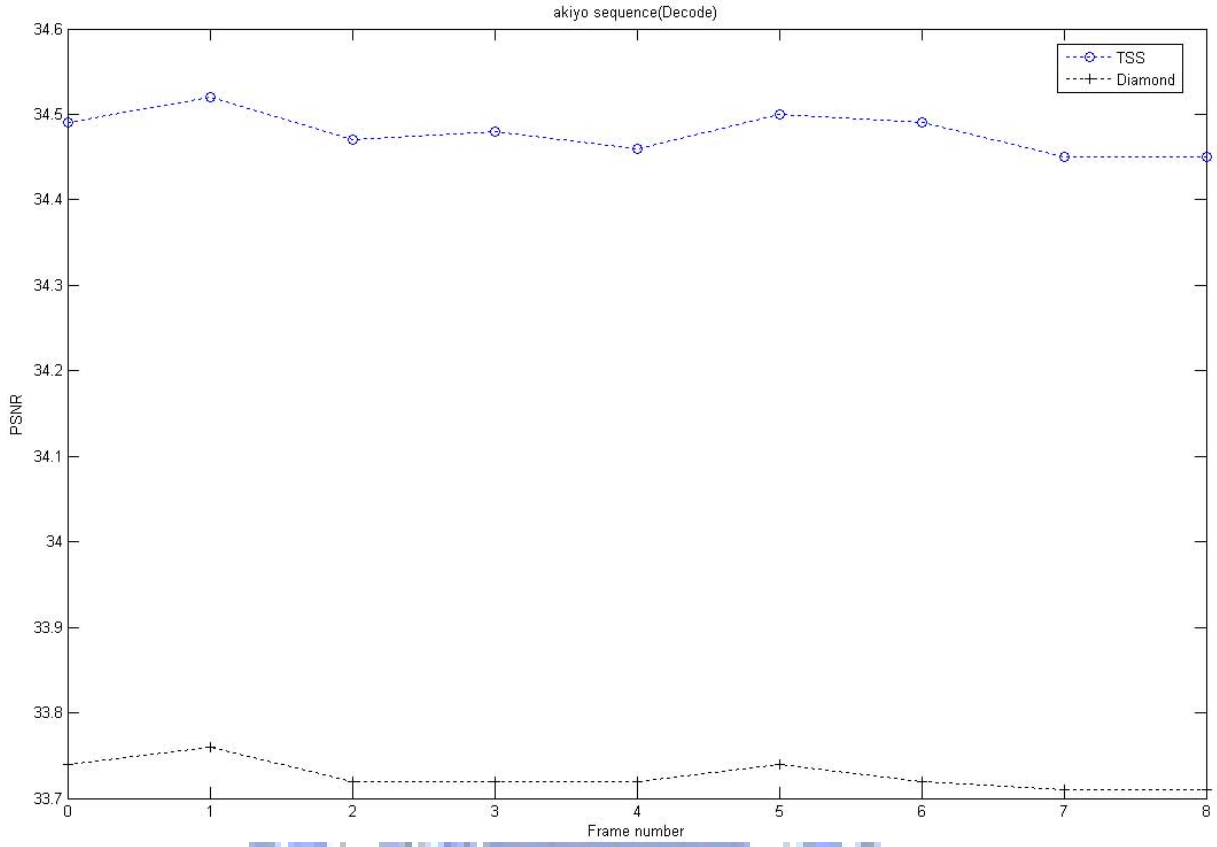

Figure 5.9: (akiyo)PSNR之比較

我們可以由模擬圖 Figure 5.9 與 Figure 5.10 看到, 其實這個測試序列在三步驟演算法的 PSNR 與鑽石演算法的 PSNR 大概只有差大約 1dB 而已, 不過在壓縮資料量來看其實壓縮後的大小三步驟演算法與鑽石演算法是幾乎一模一樣的。所以其實三步驟演算法對此序列在視訊品質上並有改善太多。

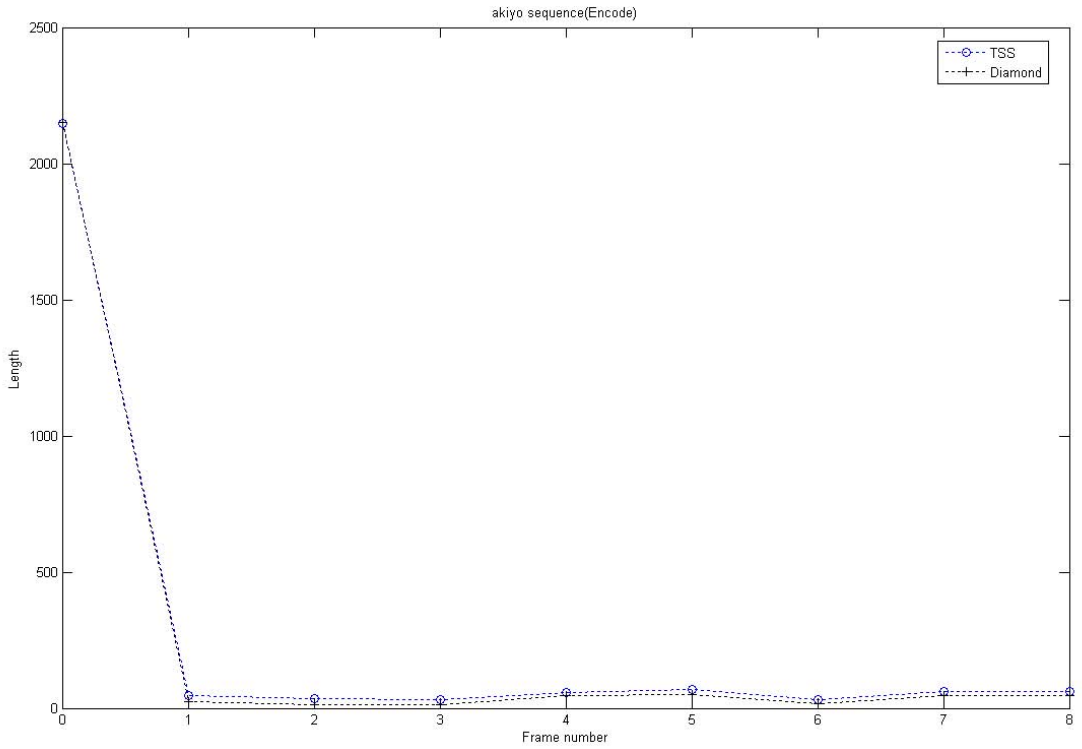

Figure 5.10: (akiyo) 壓縮長度之比較

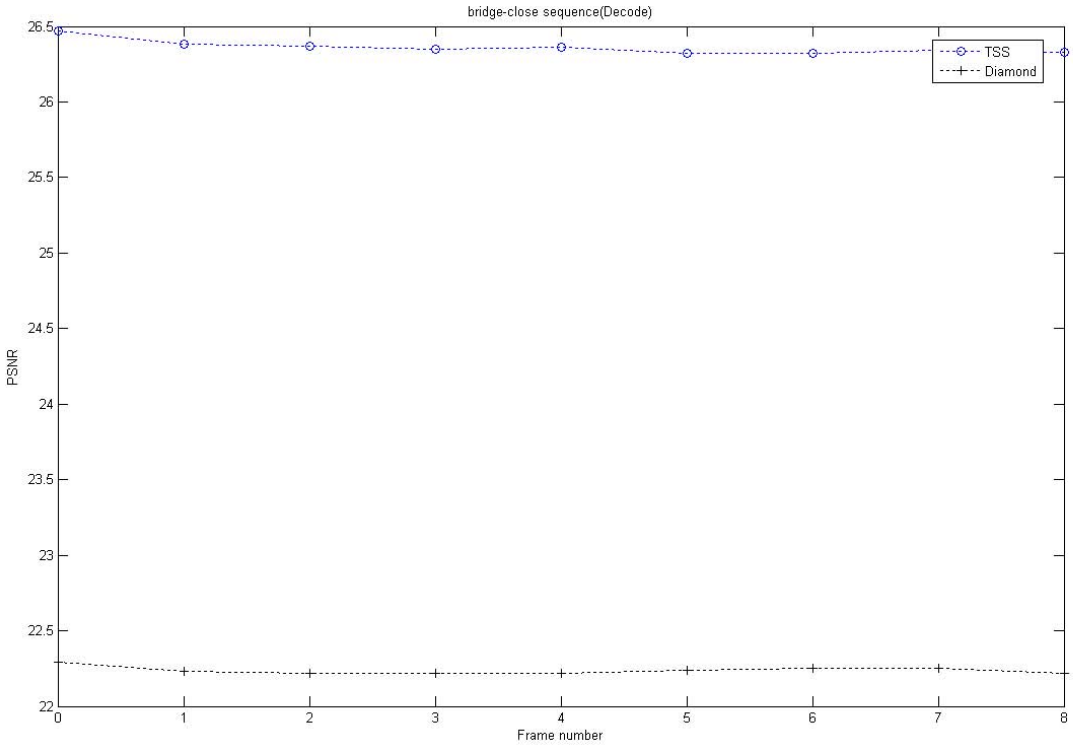

Figure 5.11: (bridge-close)PSNR之比較

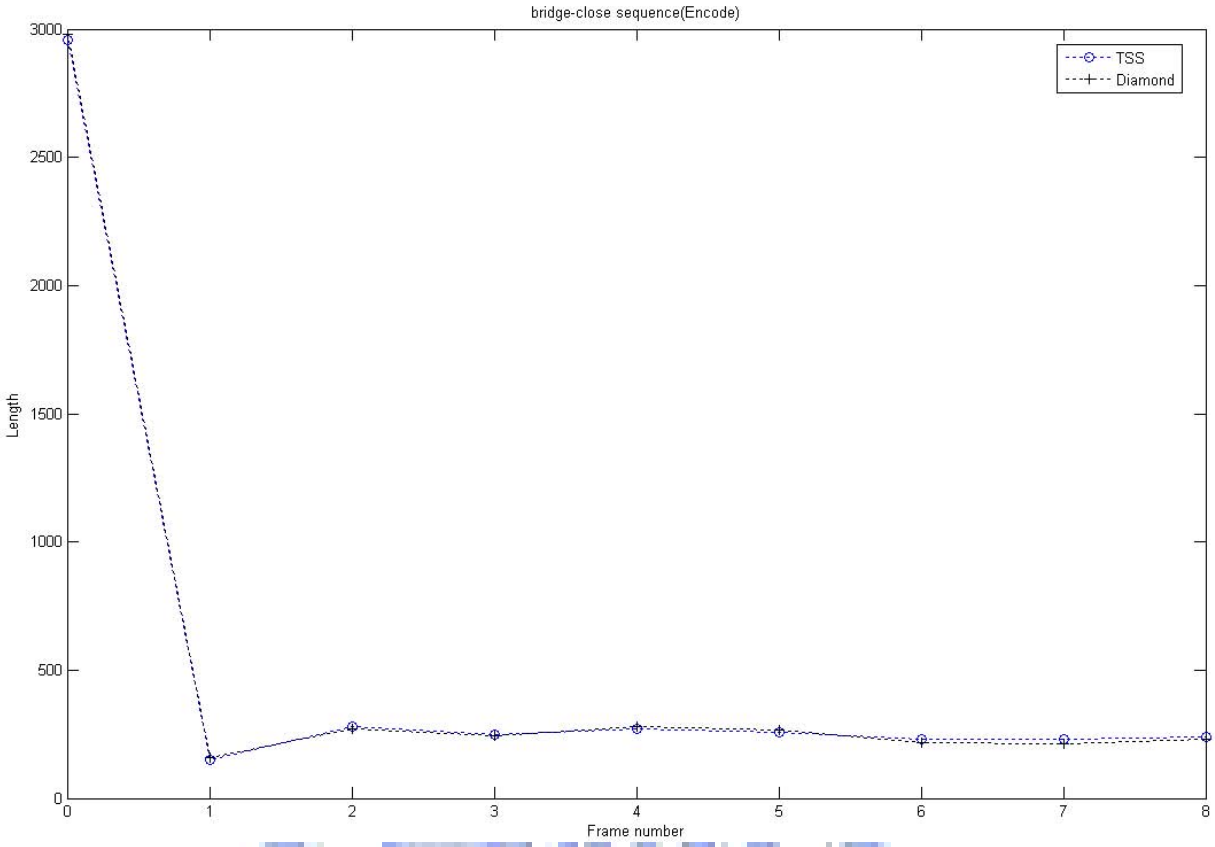

Figure 5.12: (bridge-close) 壓縮長度之比較

接下來的測試序列可以由模擬圖 Figure 5.11 與 Figure 5.12 看到 PSNR 差距高達 3dB 以上, 且並不會因爲 PSNR 的上升使得壓縮資料的長度變長。

接下來的測試序列模擬圖 Figure 5.13 與 Figure 5.14 , 可以看到更明顯的 PSNR 差距。

而模擬圖 Figure 5.15 ～ Figure 5.18 則是維持至少 3dB 的 PSNR 差距。

而低於 3dB PSNR 的測試序列, 主要理由是因爲畫面背景過於單調且佔據了大部分的畫面內容, 所以導致在使用此演算法計算動作向量時獲得了較不準確的向量值, 以致於在重建時的視訊畫面之品質會如此的差。

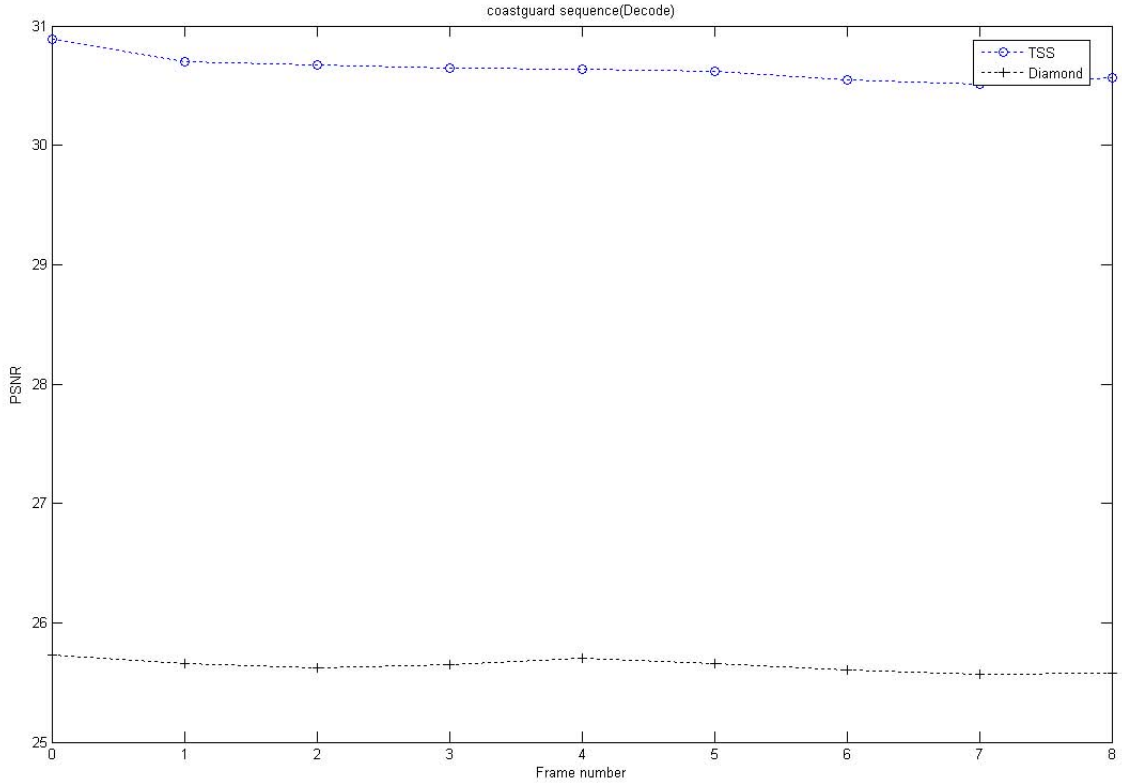

Figure 5.13: (coastguard)PSNR之比較

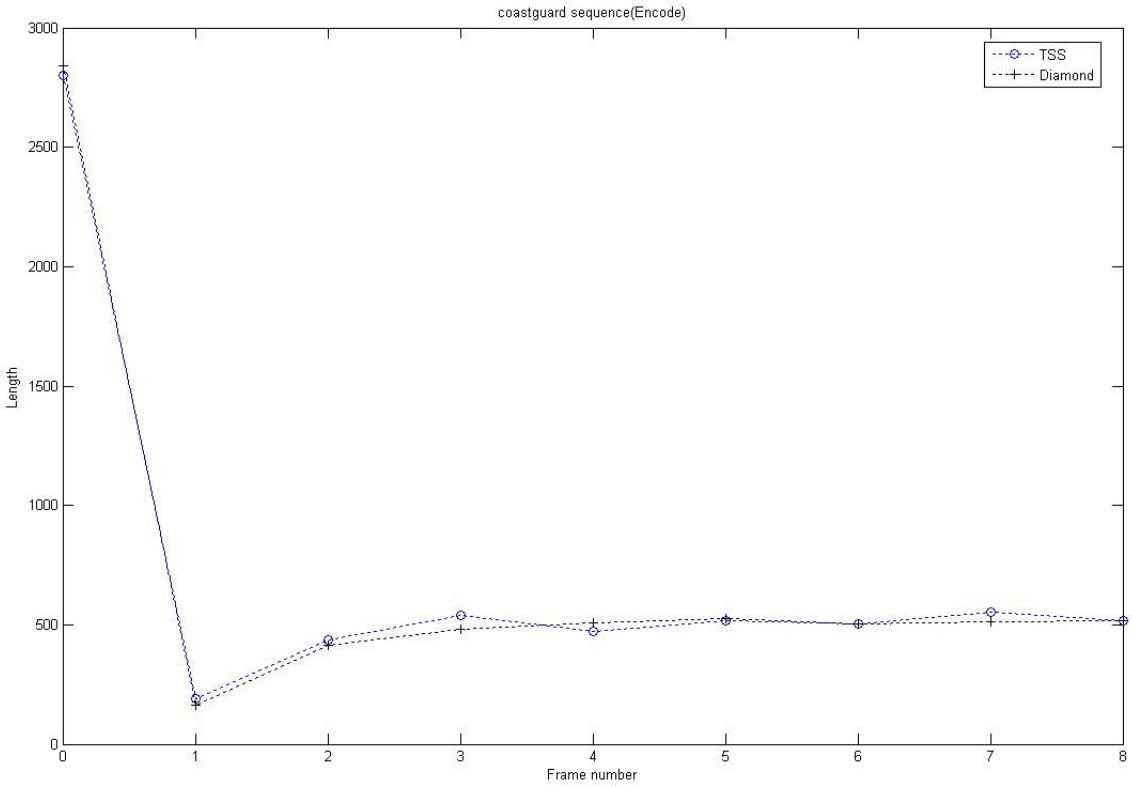

Figure 5.14: (coastguard) 壓縮長度之比較

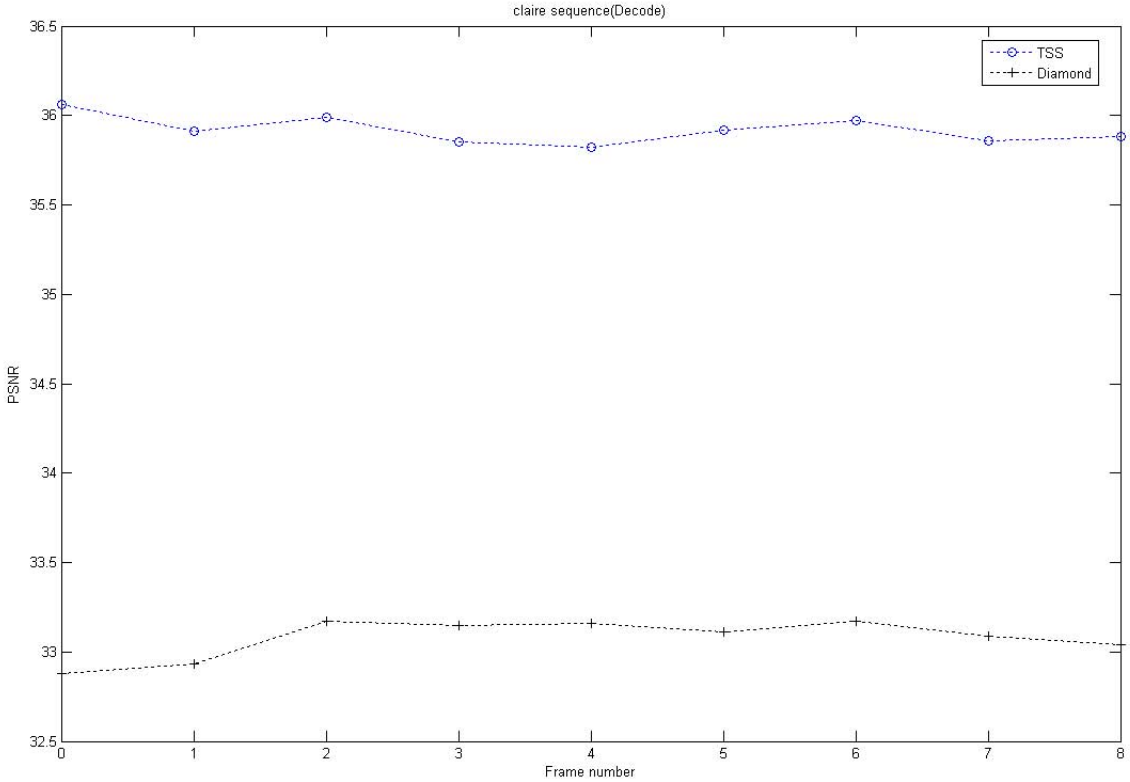

Figure 5.15: (claire)PSNR之比較

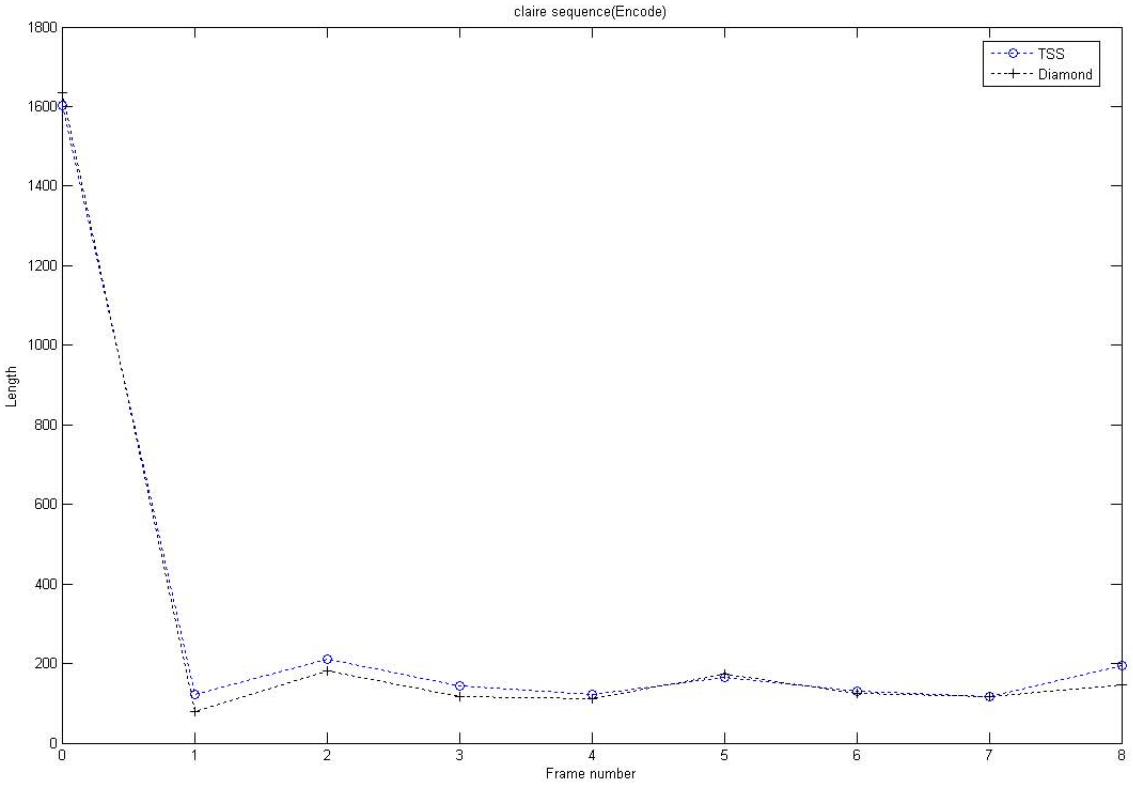

Figure 5.16: (claire) 壓縮長度之比較

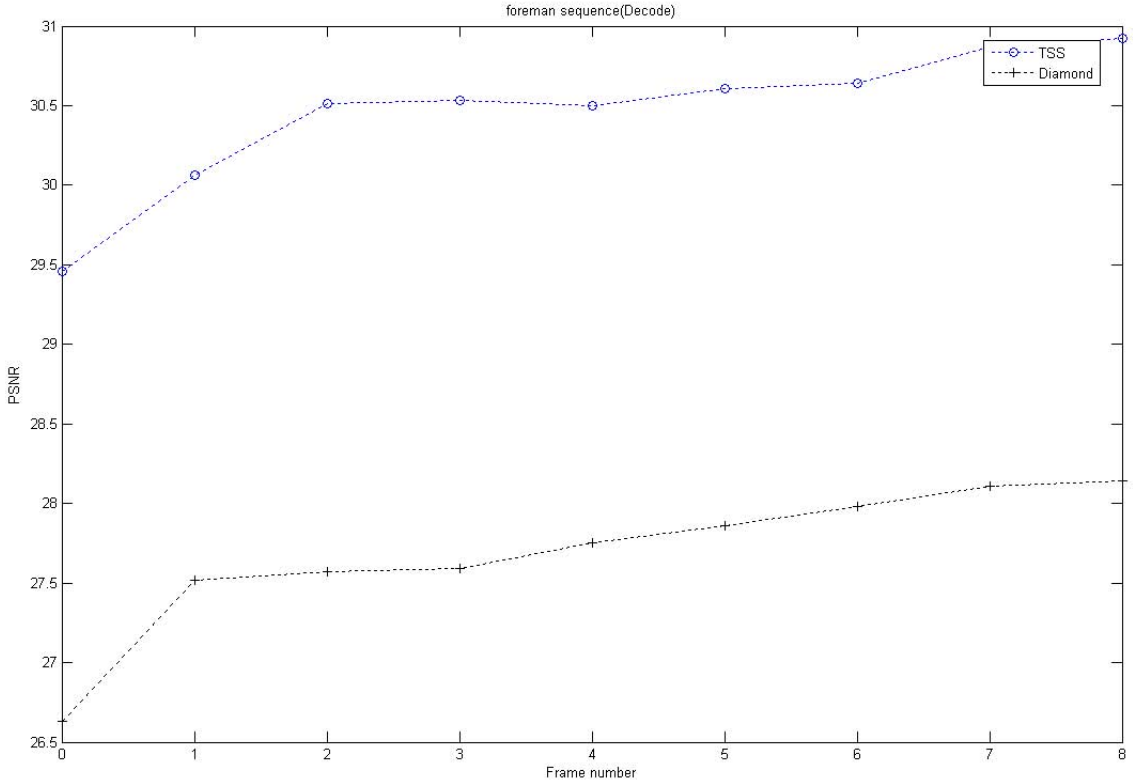

Figure 5.17: (foreman)PSNR之比較

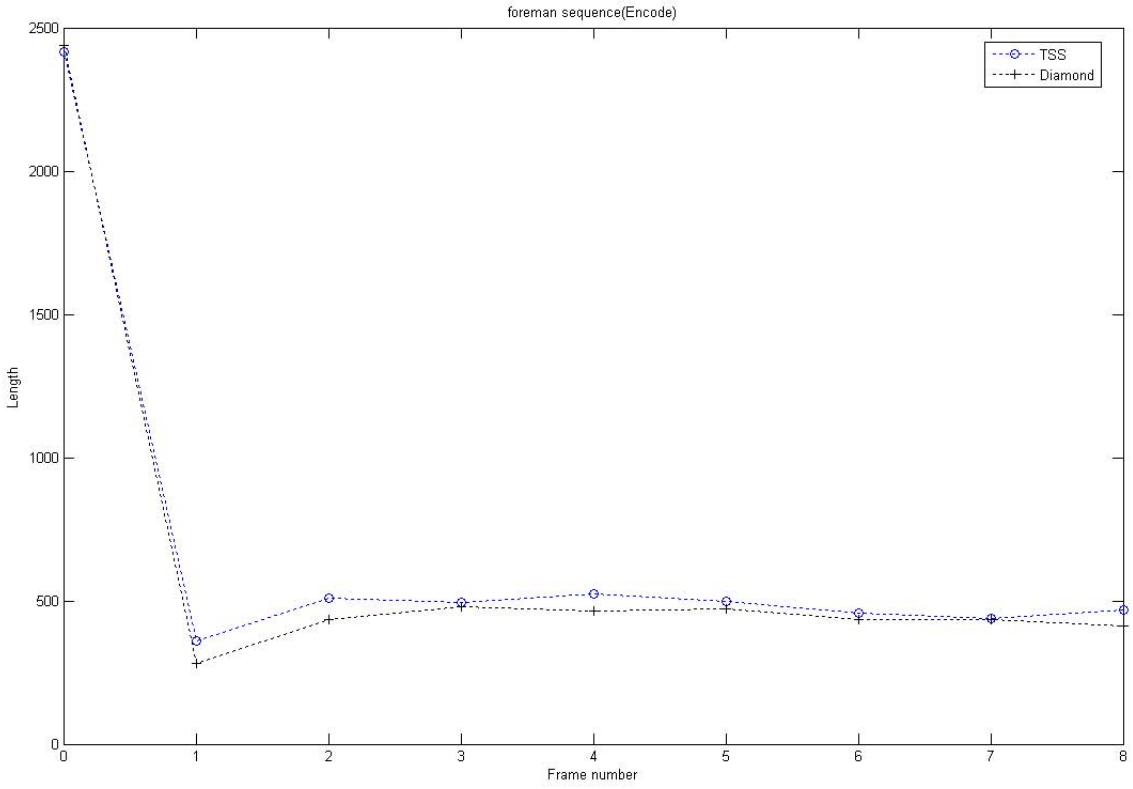

Figure 5.18: (foreman) 壓縮長度之比較

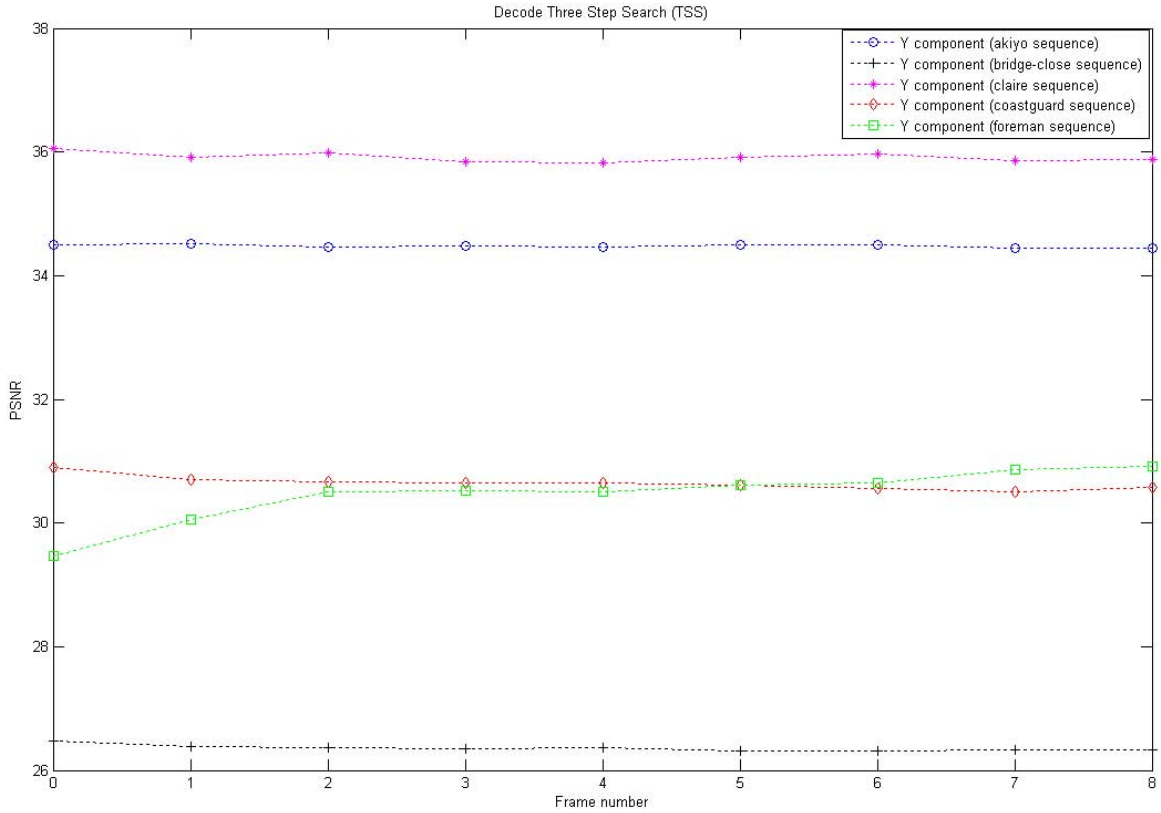

Figure 5.19: TSS之比較

由上圖五個測試序列的比較看來, 只有一個測試序列的整體 PSNR 是低於 30dB 的, 所以三步驟演算法其實並不是對所有序列都可以使視訊品質大幅提昇。如果測試序列是屬於畫面背景顏色單調且只有畫面人物之表情有改變的話, 則重建的視訊品質會很差。不過由以上可以看到使用三步驟演算法後, 我們的視訊品質大部分都至少可以增加至少 3dB 以上, 也就是說我們還是可以利用增加少量的計算複雜度來獲得較高的視訊品質。

而在一般在處理搜尋演算法複雜渡過高這方面的問題時, 是直接自行創新一個搜尋演算法演算法, 希望藉由全新的演算法使得視訊品質可以提升且計算複雜度降低。但是本文實際的作法是將原本開放式源碼 (T264) 之中所提供的快速搜尋演算法替換掉, 替換成複雜度稍微提升的快速搜尋演算法來獲得大幅度視訊品質上的提升。這裡所指的快速搜尋演算法是前人研究學者已經發表過的, 所以可以直接跳過開發新演算法這個步驟, 直接利用 DSP 平台上的加速能力使得計算複雜度降低。而實際也可以看到在 DSP 上執行的時間並沒有差距太大, 但是視訊品質 PSNR 上的表現卻有至少 3dB 以上的差距, 確實有得到預期的效果。之後便可以利用 DSP 平台上所提供的最佳化功能, 使得 H.264/AVC 可以更快更有效率的執行於 DSP 平台上。如此在眞正執行於行動視訊之上時, 才可以獲得較好的效能與較低的功率消耗。

Chapter 6

結論與未來展望

其實 H.264/AVC 還有很多需要解決的地方, 因爲在當初制訂這個標準的委員會, 一開始的訴求是在盡量不考慮複雜度的情況下, 盡力使 H.264/AVC 不管在視訊品質還是在壓縮效能上, 都能遠遠超越之前視訊標準規格或者是相容於它們。

所以導致在眞正應用時雖然視訊品質大幅提昇了, 但是複雜度也變得超乎想像的高。導致於在實現時必須仰賴於硬體的上的加速功能, 才使得演算法可以眞正的實現在行動視訊、數位電視廣播、數位監視系統、數位相機、數位攝影機與 STB(Set-Top-Box) 等等。但是追根究底上還是必須從演算法上實質的去改善本身的計算複雜度, 同時這點也是自己與其他研究 H.264/AVC 的研究學者, 所需要繼續努力的地方與追求的目標。

附件

本論文附件的內容, 主要是要介紹由德州儀器公司所自行研發的整合型開發環境 Code Composer Studio (CCS), 並且試著去利用此套軟體開發新的 DSP 演算法。

一開使用 CCS 前, 必須先選定一個 DSPs[1]作爲 CCS 日後要開發的平台。如 Figure 7.1 爲設定 CCS 平台的對話框, 中間欄位顯示可提供程式設計者或開發者程式者開發的環境, 而 CCS 目前皆可支援板卡與模擬環境 (Simulation)。左邊欄位則是顯示目前電腦所選取的開發環境。

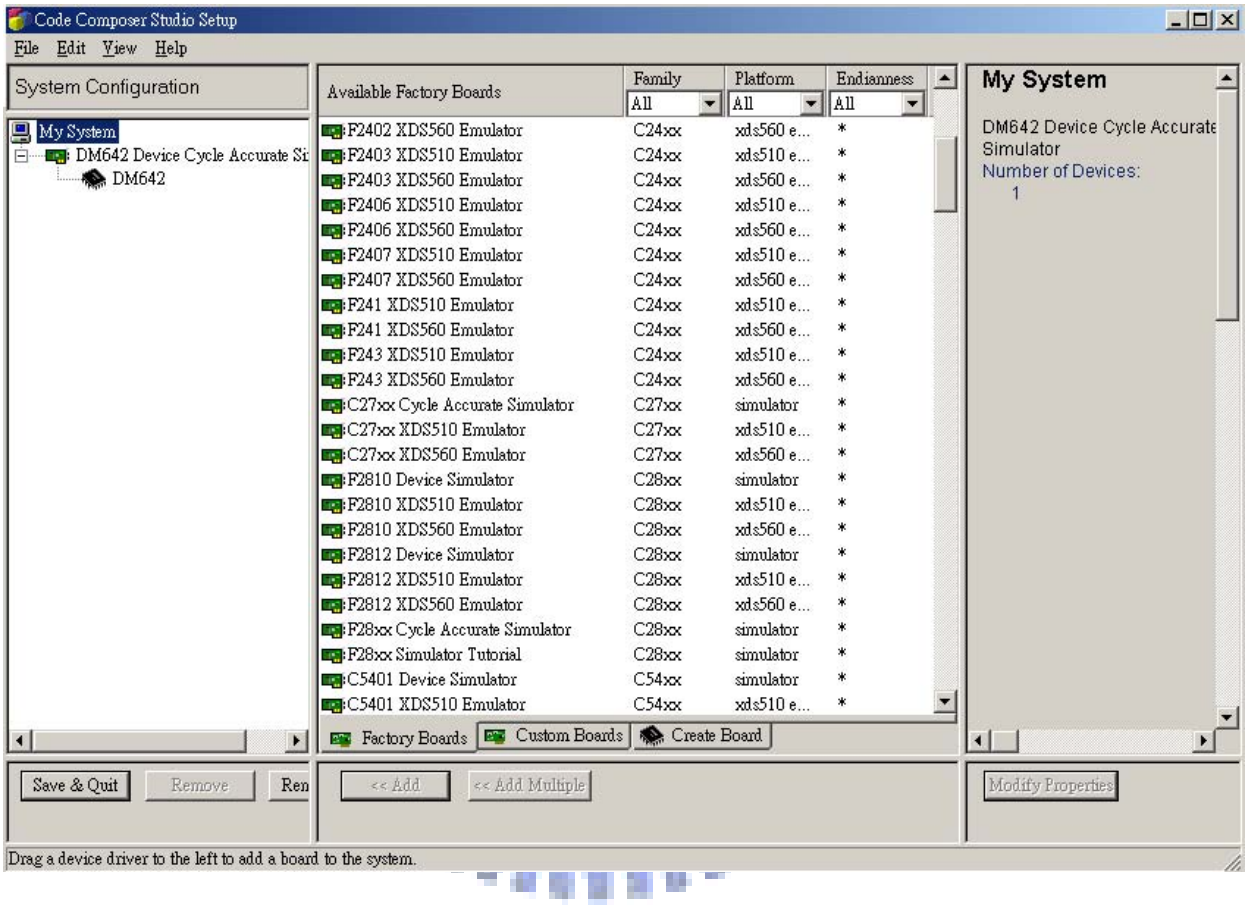

Figure 6.1: CCS平台對話框

[1]這裡所指的 DSP 是 Digital Signal Processors, 即數位訊號處理器單晶片。

仿眞器 (Emulator) 主要的用途在軟體一開始開發的階段, 爲了讓設計者可以在 run-time 的過程中除錯方便。透過 JTAG 作即時的模擬, 且仿眞器可跟 CCS 的除錯器介面配合, 進而產生即時控制與開發板的圖形分析。但是如果手邊沒有仿眞器, 可將環境設定成單機模擬環境“Simulator”。當設定好開發平台後便可開始設計第一個 DSP 程式。設計 DSP 程式的目的爲了是要能成功地載入到 DSPs 上去運作, 而 CCS 開發環境的視窗與一般在使用其他開發環境 (如 Visual C++BVisual Studio 與 Visual Basic) 非常類似, 使得初學者在此方面並不會感到如此陌生。Figure 7.2 即是 CCS 開發環境的視窗。 之後將一些所

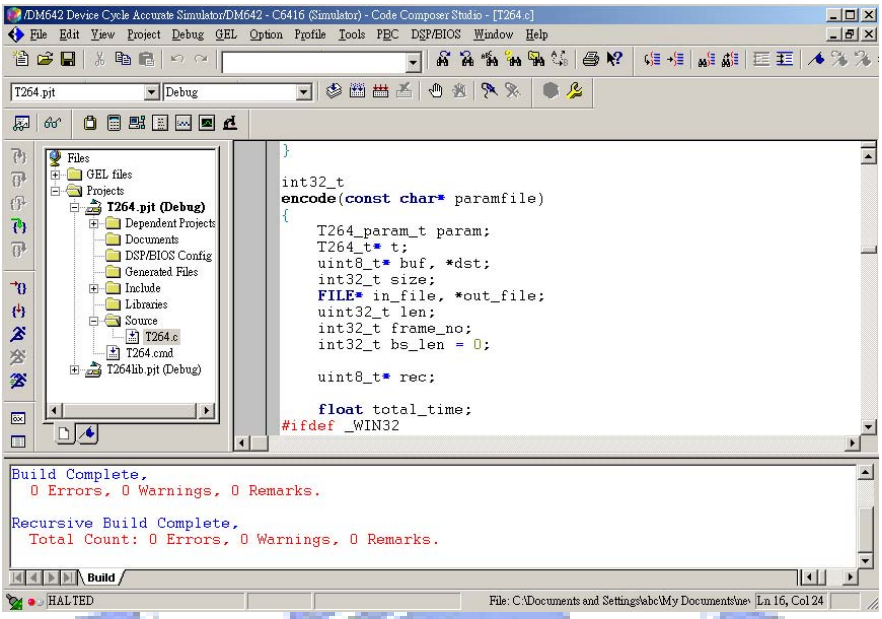

Figure 6.2: CCS環境視窗

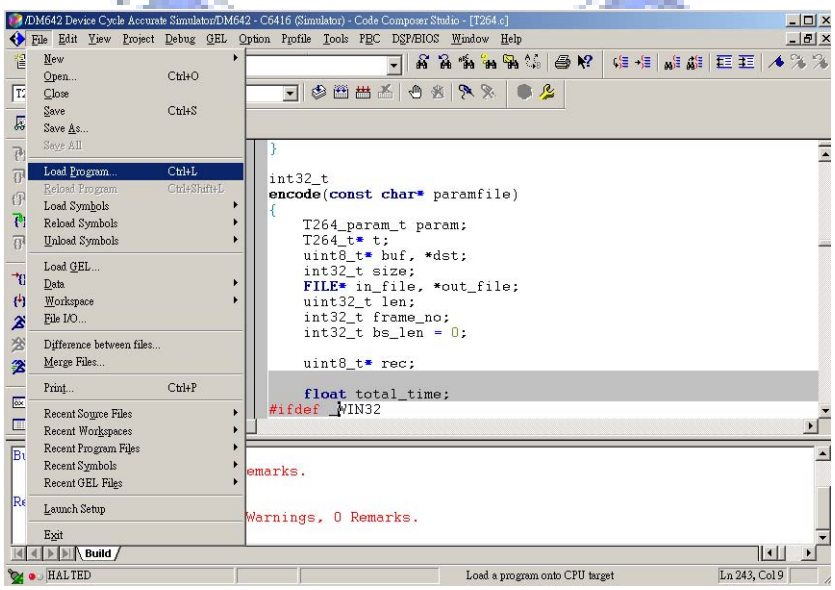

Figure 6.3: CCS Load Program

需要的程式[2]載入到左邊的“Project”中, 將整個 Project 一起編譯, 如果正確無誤便會產生一個 “*.out” 檔, 將此檔案載入到開發環境中, 如 Figure 7.3 所示, 也就是將程式載入到 DSPs 中去執行。載入之後便可以看到視窗中顯示程式碼第一步所要執行的地方, 如 Figure 7.4 所示, 如果希望載入執行檔後可直接跳到主程式去執行的話, 可以在工具列上選擇

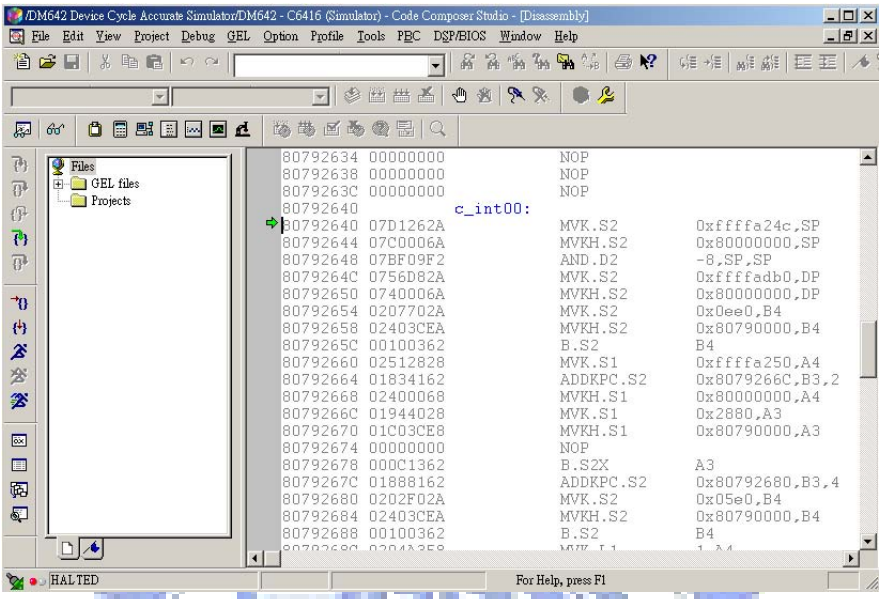

Figure 6.4: CCS載入程式檔畫面

Debug→Go Main 之後, 執行的指標便可直接移到主程式上。如果需要能夠檢查目前變數的數值與 CPU 暫存器的數值的話, 這時就需要一些工具的幫助, 如 Figure 7.5 所示。

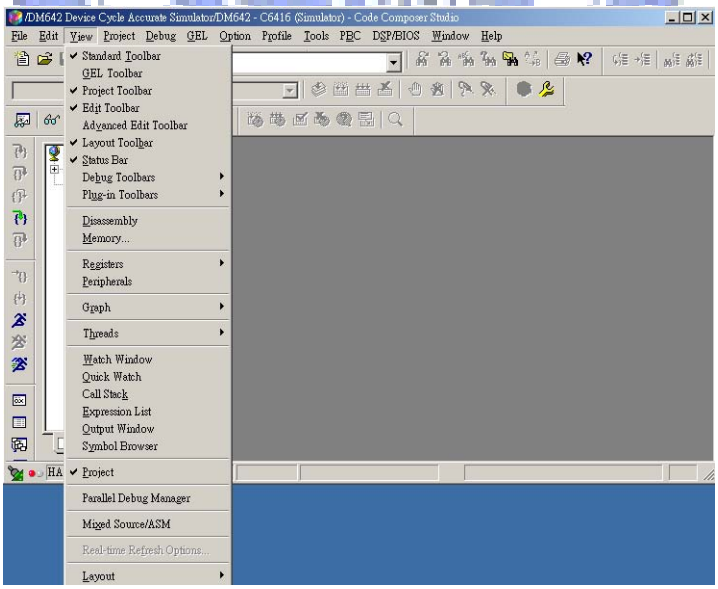

Figure 6.5: CCS工具列

[2]原始檔: *.c、*.asm 與*.sa, 命令檔:*.cmd, 標頭檔:*.h, 函示庫:*.lib

- 選項中的“Disassembly”: 將 C 語言轉成組合語言。
- 選項中的“Memory”: 設定記憶體的的位址後, 可觀察該記憶體上的數值。
- 選項中的“Registers”: 觀察目前 CPU 暫存器所存的值。
- 選項中的“Watch Window”: 列出變數的位址與目前數值。
- 選項中的“Quick Watch”: 用來快速計算變數的運算。
- 選項中的“Call Stack”: 觀察目前函示堆疊的狀態。
- 選項中的“Output Window”: 輸出視窗與編譯的結果。
- 選項中的“Symbol Browser”: 觀察 symbol 的位址、大小與範圍。
- 選項中的“Graph”: 可畫出資料的圖形。

如果要觀察該程式執行的時間, 可在工具列上選擇 Profile→Setup 之後, 就會出現如 Figure 7.6 的視窗。而圖中有很多選項可以選擇, 可依照程式設計者的需求去選取適合的選項

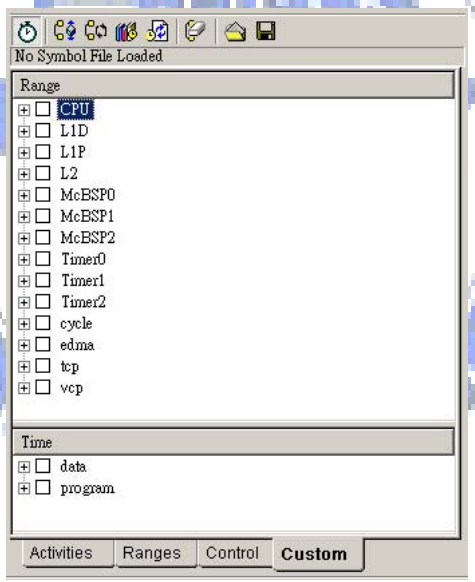

Figure 6.6: CCS Profile 設定視窗

去觀察。如果是想要觀察個別程式、函示或迴圈所需要花費的時間就必須點選該視窗下方的“Ranges”選項。如 Figure 7.7 所示。接著還可設定編譯器最佳化的等級, 可依照程式設計者的需求自行去選取, 如 Figure 7.8 所示。但是如果等級越高相對也會付出記憶體變多的的代價, 不過也相對的讓效能變快, 完全取決於程式設計者的考量。經由上述的介紹, 程式設計者便可利用 CCS 自行開發新的 DSP 演算法, 之後再載入至 DSPs 中去執行即可完成整個 DSP 開發的過程。

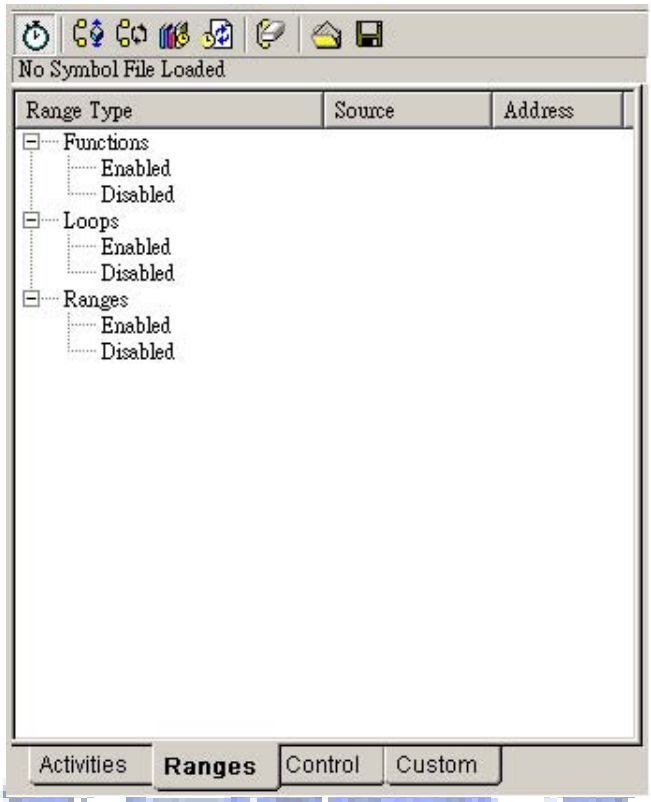

Figure 6.7: CCS Profile 範圍

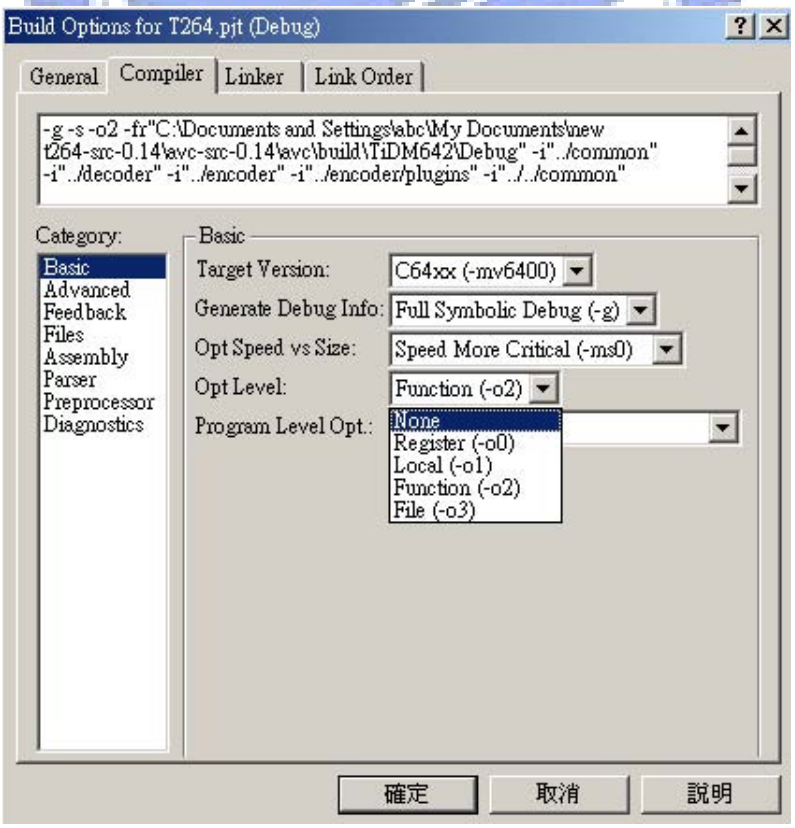

Figure 6.8: CCS 編譯器最佳化設定

Chapter 7

參考文獻

[1] "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496–10 AVC)," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG,JVT-G050r1, Fairfax, VA, 2003.

[2] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, 99 438–442, Aug. 1994.

[3] L. M. Po and W. C. Ma,"A novel four–step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, 99. 313–317, June. 1996.

[4] T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[5] G. Bjontegaard and K. Lillevod, "Context–adaptive VLC (CAVLC) coding of coefficients," Joint Team (JVT) of ISO/IEC MPEG and ITU–T VCEG Document, Fairfax, VA, JVT–C028, May. 2002

[6] I G. R. Richardson., "H.264 and MPEG–4 Video Compression," New York: Wiley, 2003.

[7] S. Golomb, "Run-length encodings," *IEEE Trans. Information Theory.,* vol. 12, no. 3, pp. 399–401, Jul. 1966.

[8] Iain Richardson, "H.264 White Paper," http://www.vcodex.com/h264.html

[9] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers.,* vol. C–32, pp. 90–93, Jan. 1974.

[10] S. Zhu, K. K. Ma, "A new diamond search algorithm for fast block–matching motion estimation," *IEEE Trans. on Image Processing.,* vol. 9, no. 2, Feb. 2000

[11] Vun,Nicholas and Nguyen,T N A. "Development of H.264 encoder for a dsp baesd embedded system," *ISCE 2007.*, pp. 1–4, June. 2007.

[12] H.-J. Wang, Y.-J Huang and H. Li, "H.264/AVC Video encoder implementation based on TI TMS320DM642," *IIH–MSP 2006.*, pp. 503–506, Dec. 2006.

[13] R. C. Gonzalez and R. E. Woods.,"Digital image processing," Prentice-Hall,2001.

[14] 戴顯權, 資料壓縮, 紳藍出版社,2001.

[15] 戴顯權, 多媒體通訊原理. 標準. 與系統, 紳藍出版社,2005.

[16] 繆紹剛, 數位影像處理–運用 MATLAB, 東華書局,2005.

[17] 陳信宏、王逸如, 數位信號處理的新利器 TMS320C6X, 全華科技圖書,2004.

[18] 盧怡仁、蔡偉和, 單晶片於數位信號處理的應用–以 TMS302C6000的開發平台爲例, 文魁資訊,2007.

[19] 陳宏宇,DSP 系統設計, 松崗出版社,2004.