國立交通大學

電機學院與資訊學院 資訊學程

碩士論文

嵌入浮水印於可調式視訊

Embedding Watermarks in Scalable Video

研究生：王季培

指導教授：傅心家　教授

中華民國九十五年七月

嵌入浮水印於可調式視訊

Embedding Watermarks in Scalable Video

研 究 生：王季培　　Student：Chi-Pei Wang

指導教授：傅心家　　Advisor：Prof. Hsin-Chia Fu

國 立 交 通 大 學

電機學院與資訊學院專班 資訊學程

碩 士 論 文

A Thesis

Submitted to Degree Program of Electrical Engineering and Computer Science

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

嵌入浮水印於可調式視訊

研究生：王季培　　指導教授：傅心家 教授

國立交通大學 電機學院與資訊學院 資訊學程（研究所）碩士班

摘　　要

這篇論文是以可調式視訊編碼 (scalable video coding) 的演算法為基礎，在可調式視訊中嵌入浮水印，首先介紹可調式視訊編碼的演算法，可調式視訊編碼包含時間維度 (Temporal domain) 的轉換、空間維度 (Spatial domain) 的轉換、壓縮編碼 (Entropy coding)、和移動評估 (Motion Estimation)。移動評估是用來選擇移動向量 (Motion Vector) 的編碼模式。而我嵌入浮水印的演算法，是將浮水印嵌入在空間維度轉換後 L frame 的中頻，採用區塊的方法來嵌入，並加上安全密碼 (Security Key) 和量化區間係數 (quantization coefficient)，來加強浮水印的強固性。我們也針對不同的解析度、傳輸速率和量化區間係數做實驗，由實驗中可以看出來在嵌入浮水印演算法中，不同的解析度對浮水印的偵測會產生出相同的結果，而不同的量化區間係數對視覺感受浮水印的偵測會有不同的結果，量化區間係數愈大 BCR (Bit Correct Rate) 就愈高，但在 PSNR (Peak Signal to Noise Ratio) 上就愈差。我也使用影格丟棄 (frame drop)、影格平均 (frame average) 和移除平行線 (remove line) 三種破壞方式，企圖破壞浮水印，利用本論文所提的演算法還是可以偵測出所嵌入的浮水印。

Embedding WaterMarks in scalable video

Student: Chi-Pei Wang Advisor: Prof. Hsin-Ghia Fu

Degree Program of Electrical Engineering and Computer Science

National Chiao Tung University

ABSTRACT

This paper is based on scalable video coding algorithm to embedding watermark in scalable video, first we will introduce on scalable video coding algorithm. Scalable video coding algorithm include temporal domain transformation, spatial domain transformation, entropy coding, and motion estimation. Motion estimation is used to select motion vector coding mode. In my embedding watermark algorithm, watermark will embed in L frame's middle frequency after spatial domain transformation. The concept is to use block to embed watermark and append security key and quantization coefficient to make watermark robust. We also use different resolutions, rates and quantization coefficients to do some experiments. In this experiment, we can see my algorithm to detect watermark in different resolution has same result. But different quantization coefficients will get different results in detecting watermark, so quantization coefficient increase the BCR (Bit Correct Rate) will increase, but PSNR will decrease. I also try to destroy watermark use three methods frames drop, frames average and remove line, and use this paper algorithm we can detect embedded watermark successful.

誌謝

在就讀在職專班的二年中，首先我要感謝我的指導教授傅心家教授，對我在多媒體視訊研究領域的指導，陳岳宏學長對浮水印技術所提供的支援和解惑，還有陳淵祥同學努力不懈的態度，激勵著我不斷向前，最後我要謝謝交大的老師和在職專班資訊組 93 級的全體同學，帶給我充實的二年。

目錄

圖目錄

一、緒論

1.1 研究動機

現在是一個資訊傳播快速的時代，資訊的快速傳播所依靠的是網路快速的發展，由於如此也就衍生出許多不同的需求，3G 手機、PDA 和其他無線的應用，而這些應用都和網路通訊有關，所以如何能在這些移動設備上看到順暢的影片，可調式視訊 (scalable video) [1-4] 這個課題就非常重要。而可調式視訊的目的就是能在不同的網路頻寬上均可達到我們所希望的視訊品質，但在伺服器端卻只需提供一個檔案供所有的設備要求。即無論在用戶端所使用的設備為何，網路頻寬有多大，對伺服器都只要求同一個檔案，不但可以提供不同設備的需求，還可以節省檔案伺服器對空間上的負擔。

在可調式視訊中有分為兩大陣營，分別為 H.264 和三維小波轉換，由於三維小波轉換是由微軟所提，所以將三維小波轉換做為可調式視訊的源碼基礎。加上在網路上對著作財產權的沒視，就以三維小波轉換為基礎的可調式視訊來嵌入浮水印，使得在網路上所交流的視訊檔案都可以受到著作財產權的保護。

1.2 相關研究

在之前針對視訊的數位浮水印所做的研究中，有些是先將原本的影片分割成 video shot [9, 10]，以 video shot 為嵌入浮水印的基本單位，不過這類的演算法都有一個相同的缺點，就是在做浮水印偵測時，都須要透過原本的影片和嵌入浮水印的影片做比較。還有另外針對浮水印的研究 [11]，使用一個圖片來當做嵌入的浮水印，為了要達到可調式的目的，在小波轉換的過程中，將要嵌入的圖片調整成要嵌入區塊的大小，在一次的嵌入中，同樣的圖片會嵌入的不同位置，且圖片

的解析度不同，在偵測浮水印是也是要在不同的解析度下，用不同的圖片來偵測，這樣在嵌入和偵測浮水印時，效率會打折扣。

而在可調式編碼中，能夠快速且正確的偵測出浮水印也是一個重點，基於以上需要原影片和嵌入浮水印效率這兩個問題，提出嵌入浮水印的演算法，不需原本的影片就可以偵測出浮水印，而且實做容易。由於是選擇在可調式視訊嵌入浮水印，基於可調式視訊的特性，不同的網路頻寬提供不同解析度的檔案，所以若將浮水印嵌入在小波轉換高頻的部份，在網路頻寬比較小的情形下，高頻的部份會被忽略，以至於浮水印就會被丟棄。所以在本論文中選擇將浮水印嵌入在可調式視訊經過小波轉換後的中頻位置，不但可以支援可調式視訊的特性也可以使得在低頻影像得以完整不受破壞。

本篇論文的第二章是介紹可調式視訊編碼的演算法，第三章則是嵌入浮水印的演算法，第四章是實驗結果，第五章是結論與未來展望。

二、可調式視訊編碼的演算法

在可調式視訊編碼 (scalable video coding) 我們使用的是以 wavelet 來達到可調式編碼的演算法 [1-5]，主要概念就是二維小波轉換再加上一維的時間維度 (Temporal domain) 的轉換，而通常時間維度的轉換都會在二維小波轉換之前，並且在時間維度的轉換中增加了移動補償 (MC, Motion Compensation) 的部份。可調式視訊編碼在空間頻率維度 (space-frequency domain) 提供了多階 (Multi-scale) 影像的表示，在傳輸速度 (rate)、品質 (quality) 和解析度 (Resolution) 上提供了可調式的能力。因為在可調式視訊編碼中，移動補償是整個編碼過程中最花時間的，所以為了降低在較低的位元速率 (Bit rate) 預測移動的負擔 (overhead)，就定義了 micro-block 層移動模式的預測。

2.1 時間維度轉換

在時間維度的轉換中，分為兩種方法，一種是傳統的方法，將鄰近的兩個影格中的點 (pixel) 之間做移動評估 (Motion Estimation)，進而產生移動向量 (Motion Vector)，將移動向量串連起來成為移動線程 (Motion thread)，由於這種方法是以一個點和下個點的關係產生，這會造成無法精準的表現出影格的移動。另一種方法是使用 lifting-based motion-threading 的方式，原本只有針對點來做移動評估而變為可以針對 1/2–pel 和 1/4 –pel 解析度來做移動評估，使得產生出來的移動向量更為精準。這裡所提的 lifting-based 小波轉換是使用 5/3 正交小波轉換來實作。

2.1.1 移動評估

Lifting-based 小波轉換中的移動評估是依照影格中的點 (pixel)，產生出低通 (Low pass) 小波係數 {L0, …, L3} 和高通 (High pass) 小波係數 {H0, …, H2}。由

圖 2.1 可以得到 lifting steps 如下：

$$\begin{cases} H_i = x_{2i+1} + a(x_{2i} + x_{2i+2}) \\ L_i = x_{2i} + b(H_{i-1} + H_i) \end{cases} \qquad (2.1)$$
$$a = -1/2, \quad b = 1/4$$

在 Lifting-based 的演算法中，每一次的步驟只會更新一半的點，而且每一個 lifting base 的步驟都是可逆的，見圖 2.1。

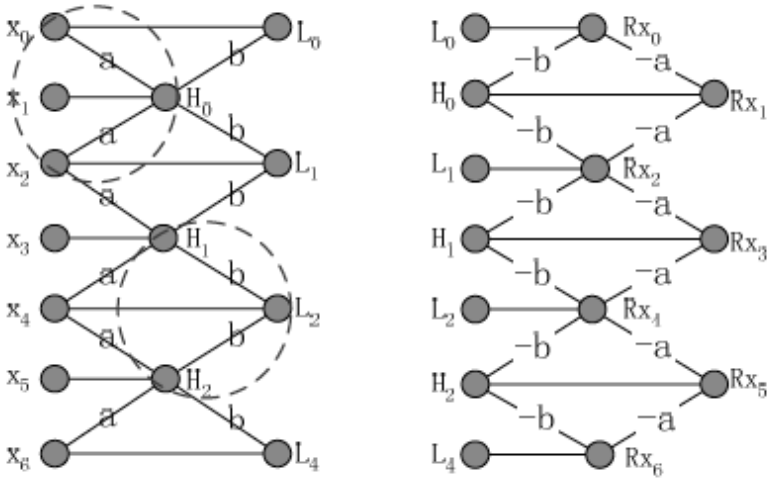


圖 2.1 用 5/3 正交小波轉換實作 Lifting-base 小波轉換。左邊的是 lifting based 中的正向轉換，右邊的是反向轉換

圖 2.2 可以看出，我們先更新奇數的影格，然後才更新偶數的影格，其中所有的影格中的區塊都代表著一個點，而由奇數影格對鄰近的一對偶數影格做移動評估，產生出對應的移動向量，單位就是影格中的點。在每個 Lifting step 中的點，無論是在奇數或偶數的影格中，都位在移動線程的路徑上，經過正規化後公式如下：

$$\begin{cases} P_{H_i} = P_{F_{2i+1}} + a(MT(F_{2i}) + MT(F_{2i+2})) \\ P_{L_i} = P_{F_{2i}} + b(MT(P_{H_{i-1}}) + MT(P_{H_i})) \end{cases} \qquad (2.2)$$
$$a = -1/2, \quad b = 1/4$$

F 是在移動線程上的偶數影格的點，P 為要更新的奇數或偶數的影格，MT(.) 為相鄰兩個影格所對應點的移動線程。

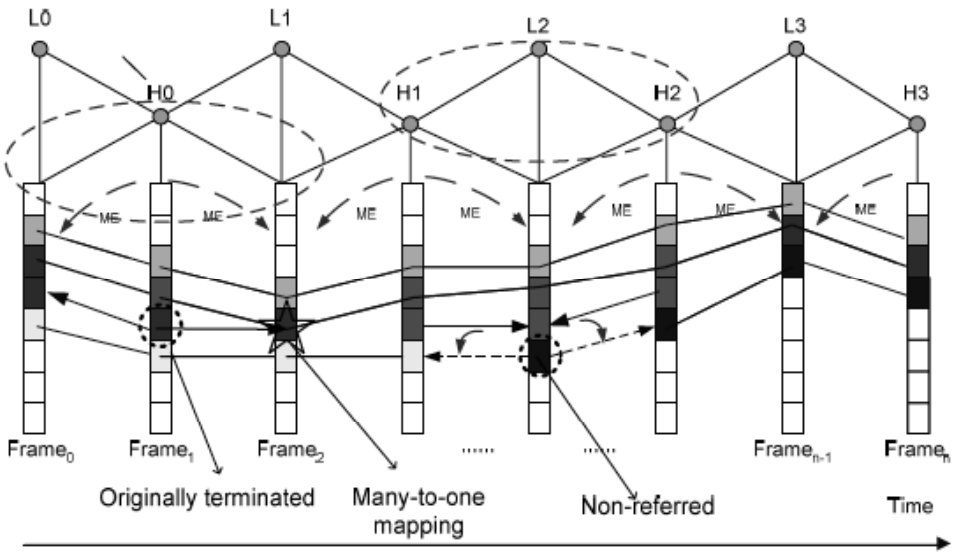


圖 2.2 Lifting-based 移動線程雙向移動搜尋

在圖 2.2 中可以看到由於 Lifting-based MT (Motion Thread) 的演算法對於移動評估提供了雙向的評估，進而解決了傳統方法所產生的終結點 (Terminated) 、多對一對應 (Many-to-one mapping) 和無參照 (Non-reference) 的問題。針對終結點這種情形，可依照左右對應的點來產生新的移動向量。而多對一對應則會以第一個對應到的移動向量為主。至於無參照的問題，則是以鄰近移動線程中的移動向量做為移動向量的方式。如此每個點都可以確保有前後方向的連結，也可確定 lifting 的方法是可逆的。而傳統的方法遇到了這三種問題時，會造成在編碼時的效率降低。

在圖 2.3 表示出 lifting 中的雙向模式的行為，黑色圈圈表示是沒有經過壓縮的點，灰色的圈圈表示是 1/2 解析度的點，白色的圈圈表示是 1/4 解析度的點。實線的部份是區塊做移動評估後所產生，虛線部份是實線的反向。以 1/2 解析度舉例來說，如果 X_2 在 F_{2n+1} 中移動評估要對應到 F_{2n} 中介於 X_1 和 X_2 的點，那麼在下一個 lifting 階段更新 F_{2n} 的 X_2 時，就會將移動評估對應到 F_{2n+1} 中 X_2 和 X_3 之間，但為反向。1/4 解析度的步驟和 1/2 解析度相同。

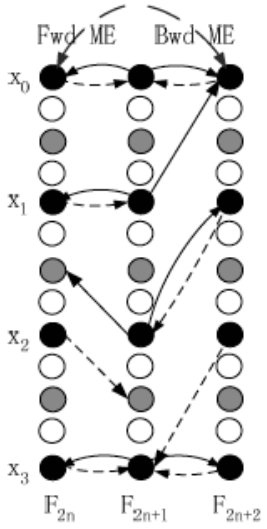


圖 2.3 多重解析度下的 lifting 運算

2.1.2 多層移動線程 (Multi-layer Motion Thread)

在可調式視訊編碼中經過時間維度轉換後，會產生多層的結構，而每一層都可以達到可調式視訊的目的。影格的數目從最高層往低層是以 2 的冪次方遞減，而在較高層的移動向量是由較低層來還原，所以在較高層的移動向量是較精準，而較低層的移動向量是比較不準的。圖 2.4 可以看到經過時間維度轉換後所產生的四層結構。

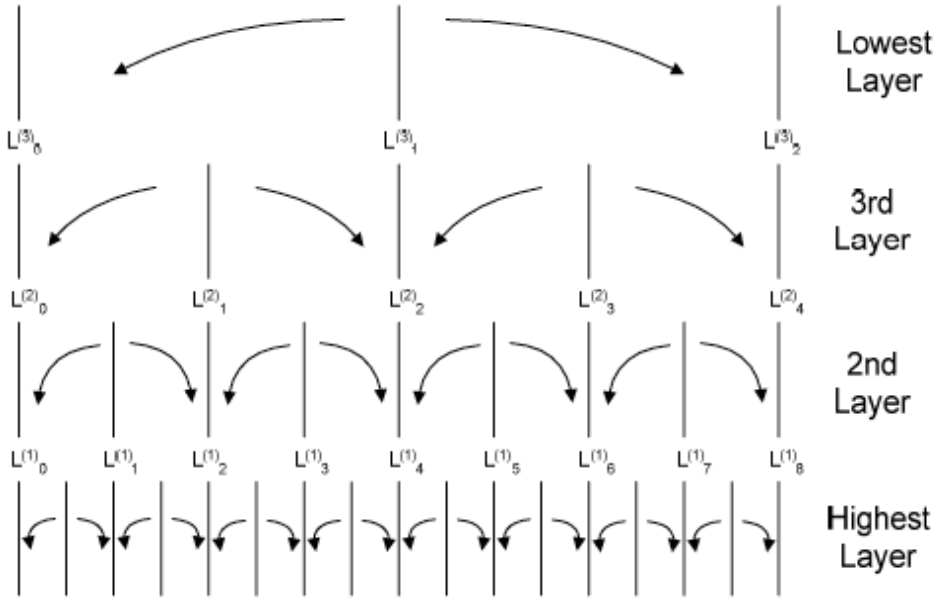


圖 2.4 時間維度的四層小波轉換結構

在做移動評估時通常都是由奇數影格對相鄰偶數影格做雙向的移動評估，圖 2.5 是在一個固定移動速率下，由 $frame_{2n+1}$ 的 (x, y) 向前向後做移動評估，而移動向量為 (dx, dy)，則在 $frame_{2n}$ 和 $frame_{2n+2}$ 中分別對應到 (x+dx, y+dy) 和 (x-dx, y-dy)，我們可以看到在前後兩個移動向量有相同的絕對值，所以在編碼時則只需對一個移動向量編碼，另一個只是反向，而這種情形就可將移動評估的結果表示為雙向模式。如此使用記錄模式的方式就可以完整記綠移動評估產生的移動向量。

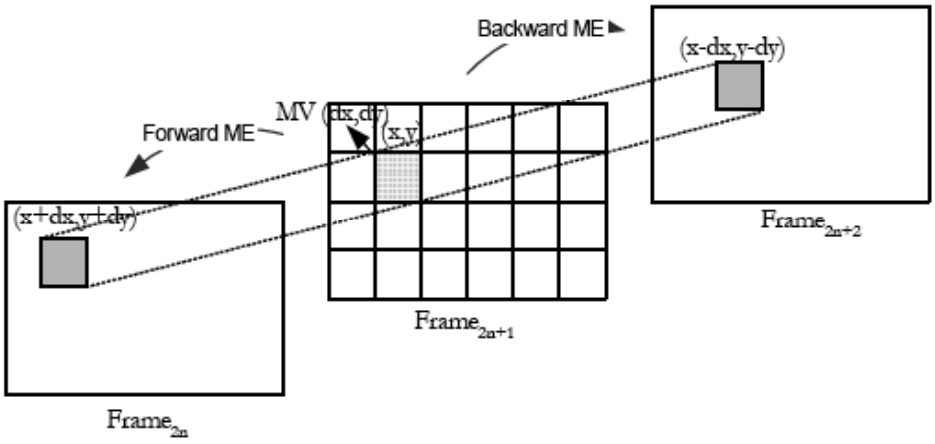


圖 2.5 在相鄰影格的移動向量和移動評估的關係

在移動評估中有分兩個方向，一個方向是從上層到下層，這一個方向在做下層解碼時，會由上層解碼下來，所以對於下層的解碼就會比較花時間，且下層會受到上層的影響。另一個方向是從下層到高層，在做上層解碼時會較花時間，且受到下層影響。圖 2.6 中我們用第二種方法來看，在較下層中有一個向前的移動評估，而在較上層則有前後兩個移動評估，因為之前有將移動向量做編碼且儲存移動評估的模式，所以在從下層到上層時，就可以很輕鬆依照下層的資訊來還原上層的移動評估。

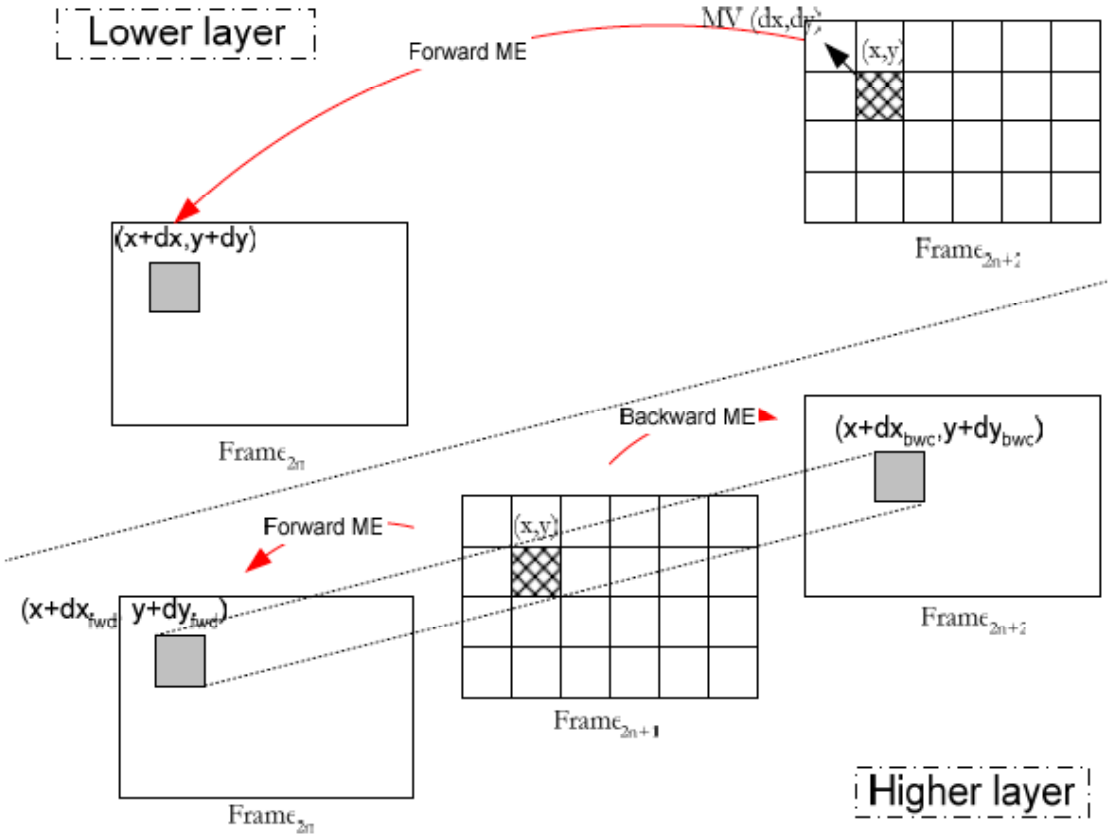


圖 2.6 不同時間層移動評估的關係

2.1.3 移動評估的模式種類

根據前面的分析，提供八種移動評估模式的選擇，圖 2.7 是移動評估流程圖，而選擇移動評估的模式則依照 R-D (Rate Distortion) 最佳化的方法來選擇，前後的移動向量會依照所選到的模式來做更新，下一個 MB (Micro-Block) 將依照這次所更新的模式來決定，移動線程也會依照更新的模式來重組。

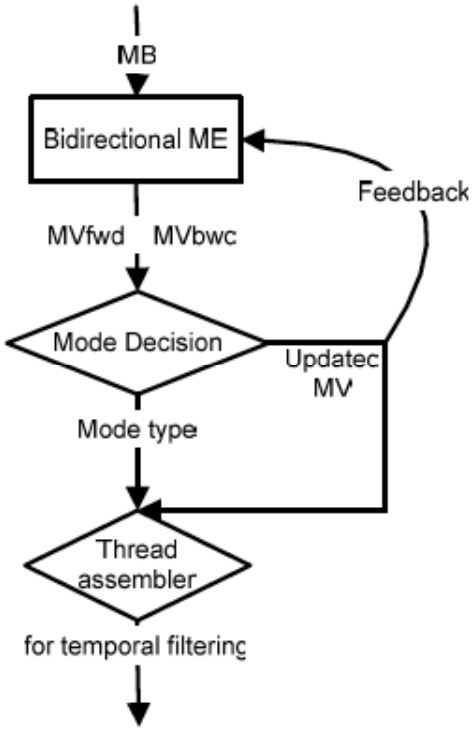


圖 2.7 移動評估的流程圖

移動評估的模式有八種可供選擇，如圖 2.8。

DirL:

在這種模式下需傳送任何移動向量，且前後的移動向量的絕對值都是等於所預測移動向量的一半，方向相反。

FT_BDL(BT_FDL):

在這種模式中只需傳送一個移動向量，若傳送向前的移動向量，則向後的移動向量可依式 (2.3) 計算

$$MV_{bwd} = MV_{fwd} \pm MV_{prevlayer} \tag{2.3}$$

若傳送向後的移動向量，則向前的移動向量可依式 (2.4) 計算

$$MV_{fwd} = MV_{bwd} \mp MV_{prevlayer} \tag{2.4}$$

FwdDir(BwdDir):

在這種模式中只需傳送一個移動向量，且前後的移動向量絕對值相同方向相反。

Fwd(Bwd):

在兩個模式 Fwd 和 Bwd 下，只存在向前或向後的移動向量，至於另一個方向的移動向量就不存在，而移動線程也就在此停止。

Bid:

這個模式是最傳統的方式，前後移動向量都會傳送。

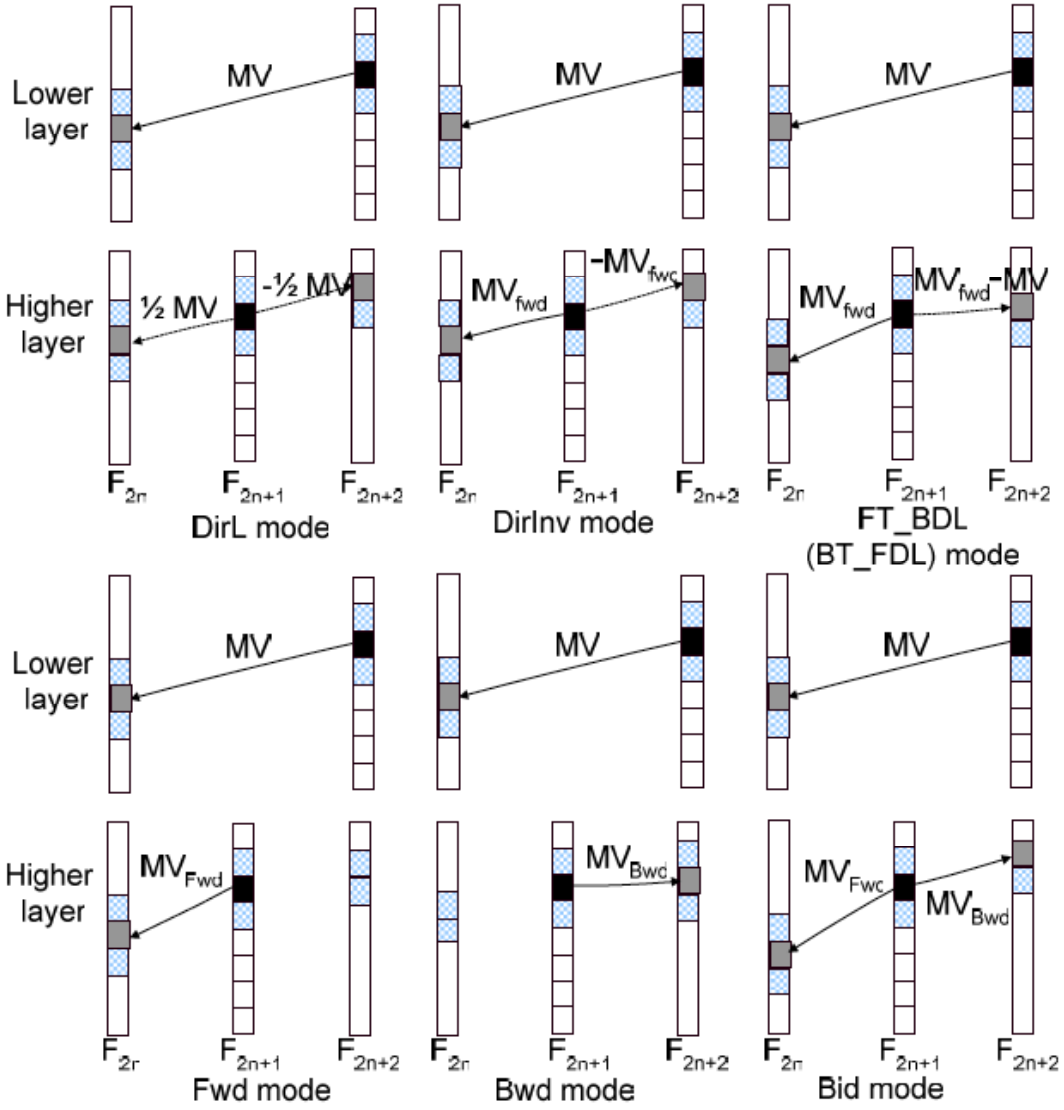


圖 2.8 在移動評估中可供選擇的八種模式

在決定移動評估時所用到的 R-D 最佳化方式的定義如下：

$$Cost = \eta \cdot SAD + \lambda \cdot Bits_{motion} \qquad (2.5)$$

SAD (Sum of Absolute Different) 是現在的 micro-block 和所對應移動評估的差異，包含前後移動向量。例如在 Bid 模式下，SAD 就是和前後移動向量所指的 micro-block 平均的差異，而 η 在 Fwd 和 Bwd 兩種模式下設定為 2，其他六種模式則設定為 1。$Bits_{motion}$ 則是移動向量經過編碼後的位元長度，以 Bid 模式來看就包含著兩個方向移動向量編碼後的位元長度，其他的模式就只包含著一個方向移動向量編碼後的位元長度。λ 則是 R-D 最佳化的權重，一般來說在低位元速率 (bit-rate) 時，R-D 的斜率會比高位元速率來得大，所以 λ 的值在低位元速率時較大。然而在沒有遞迴的小波編碼架構下，λ 的值就可設定為常數，提供給所有的位元速率，在實做編碼系統時，我們將 λ 設為 16。

2.2 空間維度轉換

空間維度的二維小波轉換是使用 9/7 濾波堆做為編碼基礎 [6]，針對每個經過時間維度轉換後的影格做空間轉換，產生 LL, LH, HL 和 HH，如圖 2.9。

LL	LH
HL	HH

圖 2.9 二維小波轉換示意圖

2.3 壓縮編碼

使用 J. Xu, S. Li, Y.-Q. Zhang, and Z. Xiong 所提出的 3D ESCOT (Embedded Sub-band Coding with Optimized Truncation) 方法來做壓縮編碼 [7, 8]，基本上 3D ESCOT 將經過時間和空間維度轉換過的影格，分割成一個個的 GOP (Group Of Pictures)，在編碼和解碼的過程中針對每一個 GOP 做處理。

2.4 圖例說明

在可調式視訊編碼時做一次時間維度的轉換如圖 2.10。在經過時間維度轉換後產生 L frames 和 H frames，再將 L 和 H frames 經過小波轉換，最後經過 entropy coding 產生出可調式視訊串流。

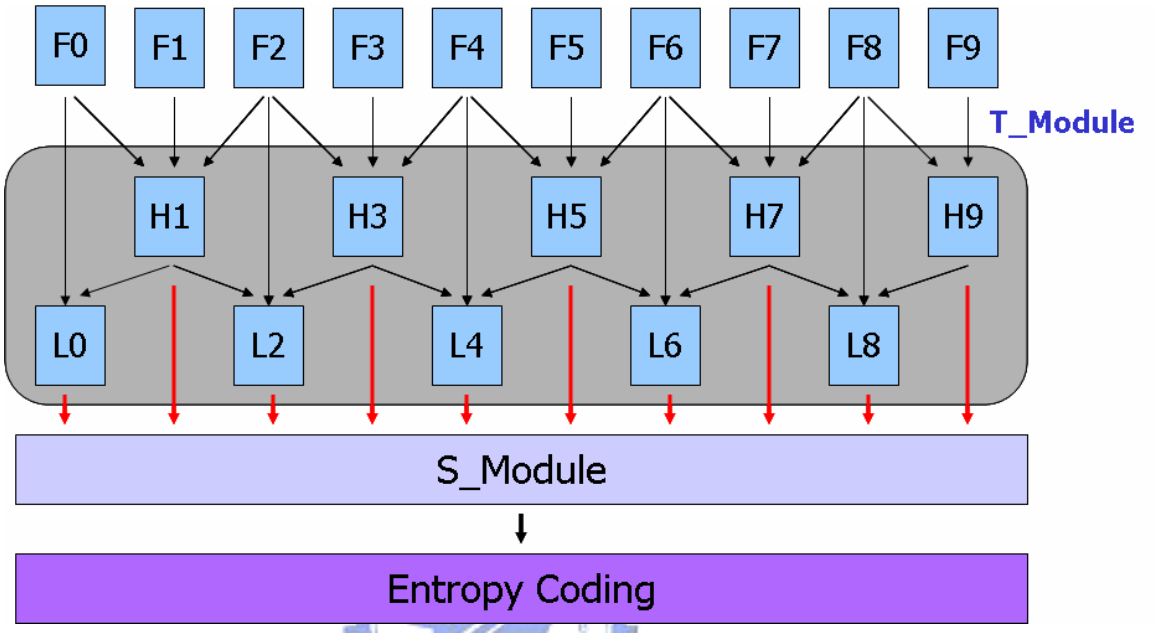


圖 2.10 一次時間維度轉換

在可調式視訊編碼時做二次時間維度的轉換如圖 2.11。在經過時間維度轉換後產生 L frames 和 H frames，再將 L 和 H frames 做第二次時間維度轉換時，只對所產生出來的 L frames 做處理，所以在圖 2.11 中可以看到，之前所產生的 L frames 會被標記成 LL 或 LH (LL 表示之前 L frames 被標記成 L frame，LH 表示之前 L frames 被標記成 H frame)，所以在第二次時間維度轉換時就針對 LL 和 LH 來做轉換，再經過小波轉換，最後經過 entropy coding 產生出可調式視訊串流。

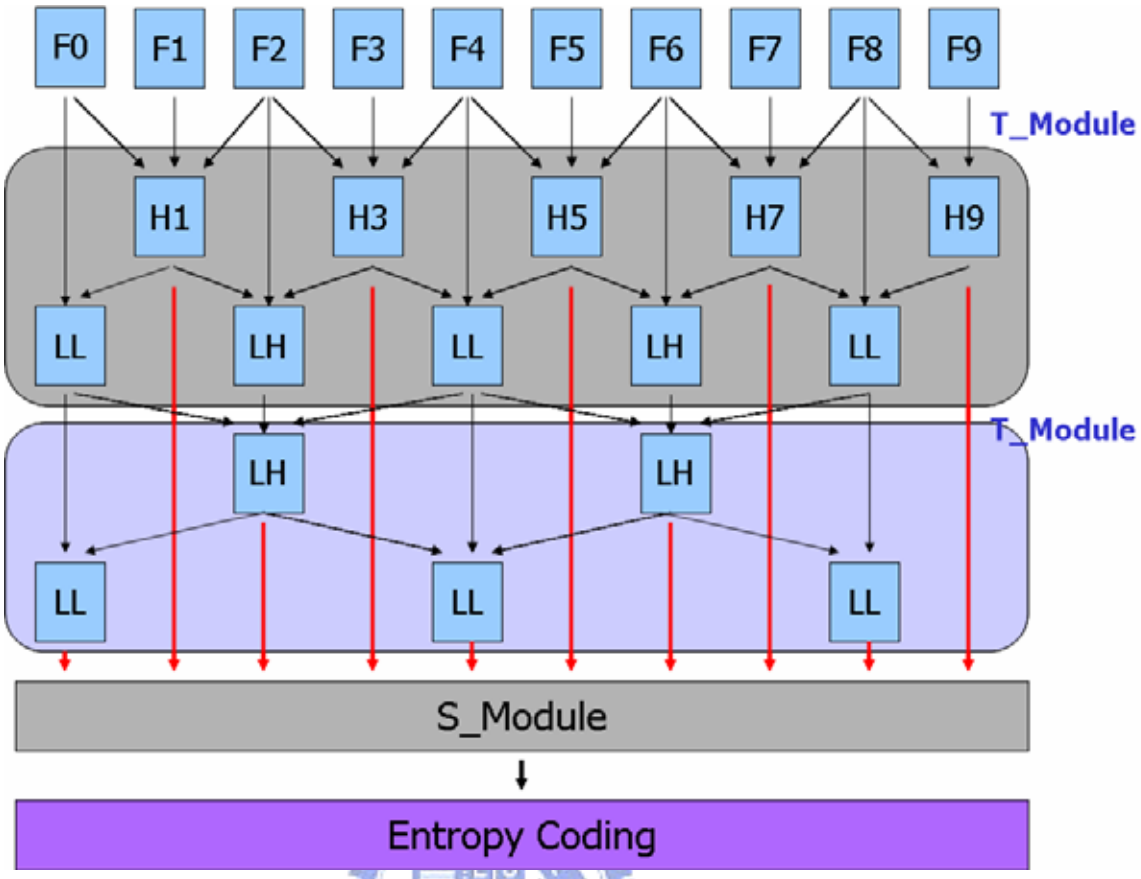


圖 2.11 二次時間維度轉換

三、嵌入浮水印演算法

在可調式視訊演算法中，有三個主要的模組 T Module、S Module、Entropy Coding，我的方法是將浮水印嵌入在 L frame 的中頻，而嵌入的時間點是在 S Module 和 Entropy Coding 之間。在原視訊經過時間濾波器和小波轉換後再加入浮水印，如此浮水印就會被嵌入在時間和空間的維度裡。

在三維小波轉換中 T Module 和 S Module 的順序是可以改變的，目前所使用的方式是 T-S-E（T Module -> S Module -> Entropy Coding），可以在可調式視訊的設定檔中做設定 S-T-S-E（Pre S Module -> T Module -> Post S Module -> Entropy Coding）。在這裡我是使用第一種方法來嵌入浮水印，在經過時間維度和空間維度的轉換後才嵌入浮水印，所以就很容易能在可調式視訊編碼的特性下做到可調式 (scalable) 浮水印的偵測，雖然經過時間維度的編碼後，會產生出每秒影格數量不同的檔案，但是卻不會影響我們對浮水印的偵測，因為實際上時間維度的轉換是在偵測浮水印之後，所以在偵測浮水印時並不會受到不同的影格數量的影響。而經過空間維度的轉換會產生出不同解析度的檔案，而我所使用偵測浮水印的方法，遇到不同解析度的解碼要求時，使用不同的解析度當成參數，由於用相同的判斷方式來偵測不同解析度下的浮水印，如此就使得在可調式視訊的條件下，偵測浮水印就變得單純。

3.1 嵌入方法

本論文所使用的可調式視訊演算法中的時間模組，會將原本的影片轉換成 L frames 和 H frames，而 L frame 是影像主要的部份，H frame 則是兩個相鄰影格的差異。所以我將浮水印嵌入在小波轉換（空間轉換）後 L frames 的中頻（所謂中頻是指在經過二次小波轉換後的低頻中高頻的部份，圖 3.1 可以看到實際上嵌

入浮水印的位置），利用區塊的概念和浮水印結合，來做嵌入的動作，其目的是為了讓嵌入的浮水印個數和視覺觀感上得到一個平衡，嵌入浮水印的數量愈多，浮水印的強固性就愈好，但在視覺觀感上就愈差，這裡是以 PSNR 做為視覺觀感好壞的判斷。由於低頻的部份在視訊中是主要的影像，而高頻的部份比較容易被丟棄，所以在中頻嵌入浮水印是較好的選擇，不但可以將影片在視覺上的差異降到最低，也會增加浮水印在經過可調視訊編碼後的強固性。所使用的浮水印是由 0 與 1 所組成的 bit stream。

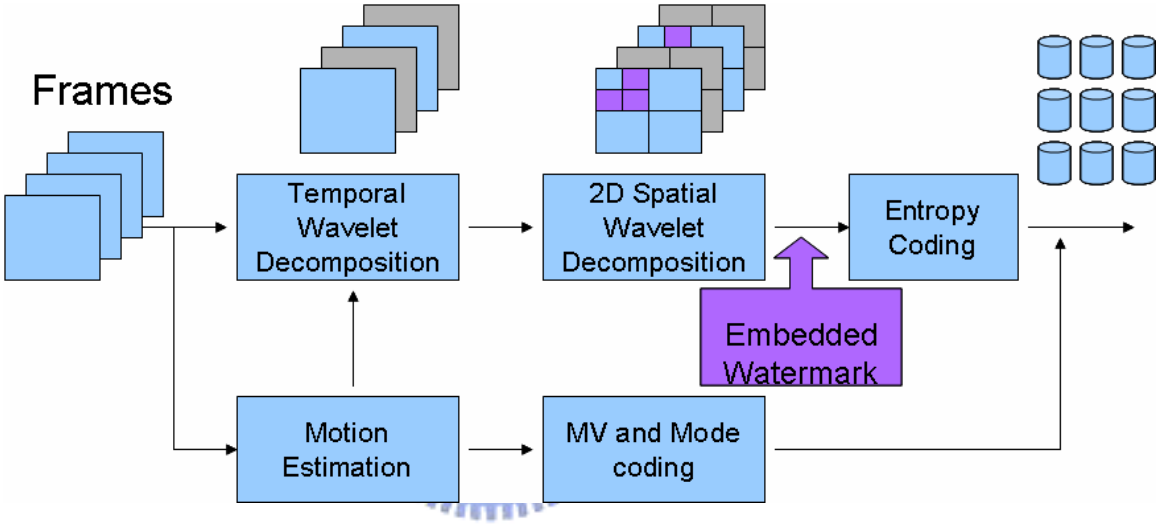


圖 3.1 嵌入浮水印系統

嵌入浮水印時將 L frame 的中頻部份，切割成 $n \times n$ 的區塊（在本論文中 n 為 8，所以每個區塊為 8x8），總共切割成 m 塊（以實驗所用影片 352 x 288 來看，則總共切割成 297 個區塊），產生一個浮水印 $\boldsymbol{W_i}$ 長度為 m（和區塊的個數相同長度為 297 個），再利用亂數產生器產生一組 $n \times n$ 安全密碼 (security key) $\boldsymbol{R_j}$，將所產生的安全密碼和原本的每一個浮水印做互斥運算 (XOR)，進而產生出新的浮水印 $\tilde{W}_j$，再依照新的浮水印的值來做嵌入

$$\tilde{W}_j = W_i \oplus R_j \tag{3.1}$$

先將區塊中的小波係數 $\boldsymbol{x_j, j=1\ldots} n \times n$，除上一個量化區間係數 α，

$$r_j = x_j / \alpha \tag{3.2}$$

根據算出的 r_j 和新的浮水印 $\tilde{W}_j$，來決定小波係數要如何調整，如果 $\tilde{W}_j == 1$，r_j 依式 (3.3) 做調整，

$$\begin{aligned} &if \quad r_j \quad is \quad odd, \begin{cases} r_j = r_j + 2, & r_j > 0 \\ r_j = r_j - 2, & r_j < 0 \end{cases} \\ &if \quad r_j \quad is \quad even, \begin{cases} r_j = r_j + 1, & r_j > 0 \\ r_j = r_j - 1, & r_j < 0 \end{cases} \end{aligned} \tag{3.3}$$

如果 $\tilde{W}_j == 0$，r_j 依式 (3.4) 做調整，

$$\begin{aligned} &if \quad r_j \quad is \quad even, \begin{cases} r_j = r_j + 2, & r_j > 0 \\ r_j = r_j - 2, & r_j < 0 \end{cases} \\ &if \quad r_j \quad is \quad odd, \begin{cases} r_j = r_j + 1, & r_j > 0 \\ r_j = r_j - 1, & r_j < 0 \end{cases} \end{aligned} \tag{3.4}$$

將 r_j 乘上量化區間α，再加上 α/2 取代原本的小波係數（最後加上的α/2 使我們在做奇偶判斷時更準確），經過式 (3.3)、(3.4) 和 (3.5) 將小波係調整後回存。

$$x_j = \begin{cases} r_j \times \alpha + \lfloor \alpha/2 \rfloor, & r_j < 0 \\ r_j \times \alpha - \lfloor \alpha/2 \rfloor, & r_j > 0 \end{cases} \tag{3.5}$$

3.1.1 實例說明

以下用例子說明嵌入浮水印的演算法。

0	2	0	0	0	2	0	1
8	0	0	0	0	0	0	1
0	0	5	43	0	0	1	0
0	1	0	6	1	1	1	0
5	0	2	0	0	0	1	1
8	1	4	4	0	10	0	7
0	0	0	0	6	0	17	0
5	2	6	11	11	0	4	1

圖 3.2 小波係數列表

以圖 3.2 來說，我們將 L frame 的中頻部份分成 2x2 的三個區塊，我們使用

的浮水印為 101（浮水印的位元個數和區塊的個數相同），而安全密碼為 1011（由於區塊大小為 2x2，所以安全密碼為 4 個位元），將浮水印和安全密碼做 XOR，產生新的浮水印為 [0,1,0,0]、[1,0,1,1]、[0,1,0,0]，所以將這三個區塊分別嵌入 [0,1,0,0]、[1,0,1,1]、[0,1,0,0]。第一個區塊中的浮水印為 [0,1,0,0]，依浮水印的值來嵌入，量化區間 α 為 5，產生出新的值後，取代原來的小波係數。如圖 3.3。

$$\widetilde{W}_0 = 0 => x_0 \Rightarrow 0/5 = 0 \Rightarrow x_0 = (0+2)\times 5 - \lfloor 5/2 \rfloor = 8$$
$$\widetilde{W}_1 = 1 => x_1 \Rightarrow 0/5 = 0 \Rightarrow x_1 = (0+1)\times 5 - \lfloor 5/2 \rfloor = 3$$
$$\widetilde{W}_2 = 0 => x_2 \Rightarrow 0/5 = 0 \Rightarrow x_2 = (0+2)\times 5 - \lfloor 5/2 \rfloor = 8$$
$$\widetilde{W}_3 = 0 => x_3 \Rightarrow 0/5 = 0 \Rightarrow x_3 = (0+2)\times 5 - \lfloor 5/2 \rfloor = 8$$

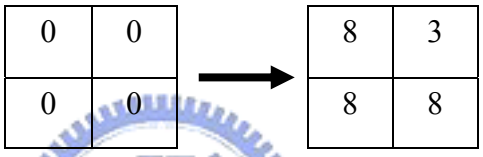


圖 3.3 嵌入浮水印後區塊中的小波係數

經過三個區塊的嵌入後，小波係數的改變如圖 3.4

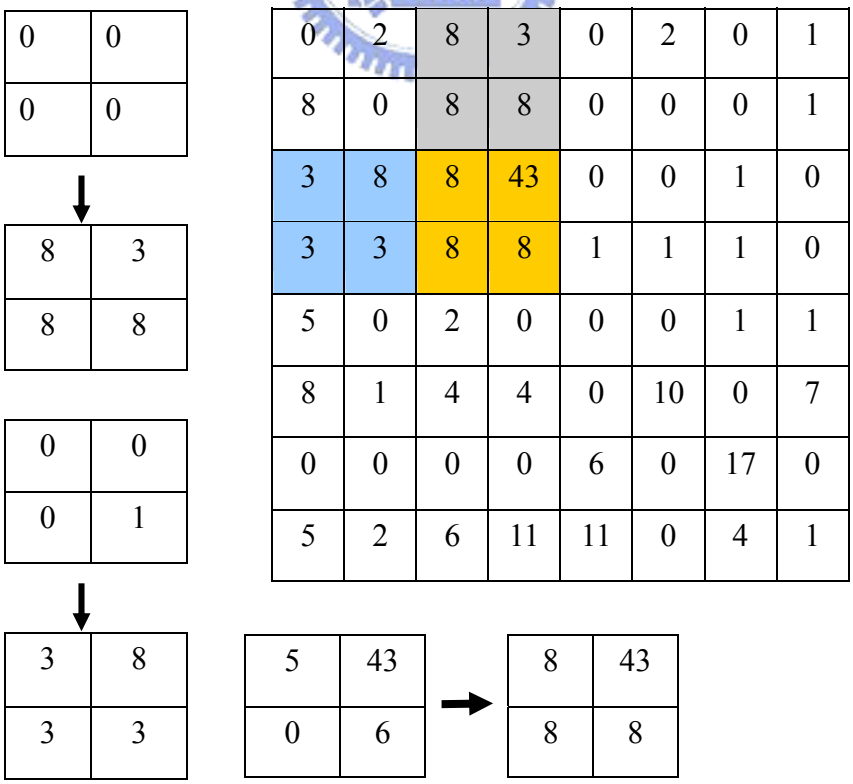


圖 3.4 影格經過小波轉換後係數的改變

3.2 偵測方法

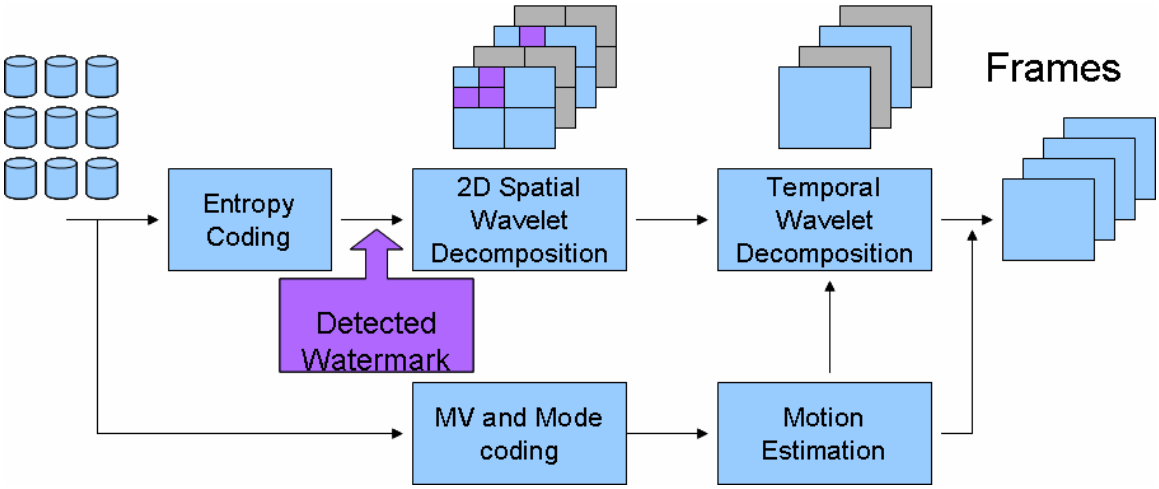


圖 3.5 偵測浮水印系統

偵測浮水印方面，在這篇論文中所使用的方法並不需要原來的影格，經過 entropy coding 後，在 S Module 之前就先取出 L frame，並在 L frame 的中頻部份切割成和嵌入浮水印時相同的 $n \times n$（8x8）個區塊總共 m（297）個，先將區塊中的小波係數 x_j 先除上量化區間係數 α，如式(3.2)。再依照 r_j 係數的奇偶特性，決定偵測出的浮水印 $\tilde{W}_j$，

$$\tilde{W}_j = \begin{cases} 1, & if \quad r_j \quad is \quad even \\ 0, & if \quad r_j \quad is \quad odd \end{cases} \tag{3.6}$$

將原始浮水印 W_i 和安全密碼 R_j 做互斥運算，再和所偵測出的浮水印 $\tilde{W}_j$ 比對，計算在區塊中的位元正確率 (B_BCR, Block Bit Correct Rate)。

$$B\_BCR++, \quad if \quad \tilde{W}_j == W_i \oplus R_j \tag{3.7}$$

如果區塊位元正確率的值大於 threshold 0.5 則累積計算每個影格中區塊所含浮水印的 BCR，如此直到做完所有的區塊（在這裡提到的 threshold 是用來取出在區塊中所嵌入的浮水印），就可以得到在這個影格中所偵測出浮水印的 BCR，可依 BCR 的值看出是否有嵌入浮水印。

3.2.1 實例說明

我們以第一個區塊做說明，區塊內的值分別為 8, 3, 8, 8，如圖 3.6

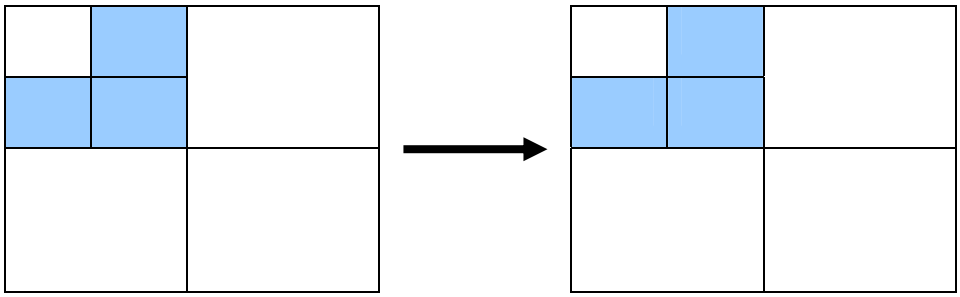
8	3
8	8

圖 3.6 嵌入浮水印的區塊

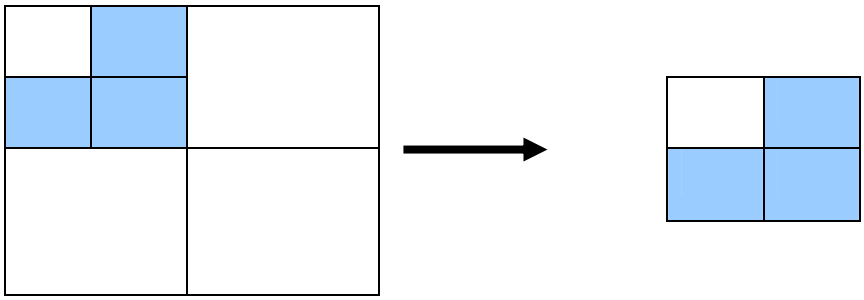
利用偵測浮水印演算法，再經過下列計算，我們將所有的區塊都判斷出相對的浮水印，完成一個影格浮水印的偵測，如此繼續做下去，直到影片中所有影格的 L frames 都已偵測完畢，結束我們偵測浮水印的演算法。

$$\tilde{x}_0 = 8 \Rightarrow 8/5 = 1 \Rightarrow \tilde{W}_0 = 0$$
$$\tilde{x}_1 = 3 \Rightarrow 3/5 = 0 \Rightarrow \tilde{W}_1 = 1$$
$$\tilde{x}_2 = 8 \Rightarrow 8/5 = 1 \Rightarrow \tilde{W}_2 = 0$$
$$\tilde{x}_3 = 8 \Rightarrow 8/5 = 1 \Rightarrow \tilde{W}_3 = 0$$
$$\tilde{W}_0 == W_0 \oplus R_0 \Rightarrow 0 == 1 \oplus 1 \Rightarrow B\_BCR++$$
$$\tilde{W}_1 == W_0 \oplus R_1 \Rightarrow 1 == 1 \oplus 0 \Rightarrow B\_BCR++$$
$$\tilde{W}_2 == W_0 \oplus R_2 \Rightarrow 0 == 1 \oplus 1 \Rightarrow B\_BCR++$$
$$\tilde{W}_3 == W_0 \oplus R_3 \Rightarrow 0 == 1 \oplus 1 \Rightarrow B\_BCR++$$
$$B\_BCR = B\_BCR/4 = 4/4 = 1 > 0.5$$

上述算式所計算出 B_BCR 大於 0.5 則我們認定這個區塊有我們嵌入的浮水印。在偵測浮水印時，不同解析度影片會使用不同的參數代入，如圖 3.7。若在可調式解碼時要解出和原圖相同解析度的影片，則在偵測的參數就會以原圖解析度當成參數，如圖 3.7(b)。 若在可調解碼時要解出和原圖 1/2 解析度的影片，則在偵測的參數就會以原圖解析度的 1/2 當成參數，如圖 3.7(d)。



(a)Embedded Watermark (b)Detected Watermark



(c)Embedded Watermark (d)Detected Watermark

圖 3.7 不同解析度的影片的偵測

四、實驗結果

在這一章中我們可以看到用本論文所提出的演算法嵌入浮水印前後的影格在視覺上的感受和實驗數據。圖 4.1 是嵌入浮水印前的影格。

圖 4.1 嵌入浮水印前的影格

在本次的實驗中，我們使用不同的量化區間來嵌入浮水印，圖 4.2(a) 和圖 4.3(a) 分別是以量化區間以 5 和 7 來嵌入，而圖 4.2(b) 和圖 4.3(b) 則是所嵌入的浮水印。以視覺上來看，在圖 4.2(a) 和圖 4.3(a) 其實我們並不會很容易的看出有嵌入浮水印。

(a) 嵌入浮水印的影格　　(b) 嵌入的浮水印

圖 4.2 α=5 時嵌入浮水印的影格和浮水印

(a) 嵌入浮水印的影格　　(b) 嵌入的浮水印

圖 4.3 α=7 時嵌入浮水印的影格和浮水印

4.1 偵測浮水印的實驗數據

實驗的設定參數：

檔案：forman_352x288_300.yuv

解析度：352x288

影格數量：300

每秒影格數量：30 frames/sec

程式設定：

Random key: 12345

Block_size: 8 x 8

量化區間 α: 5

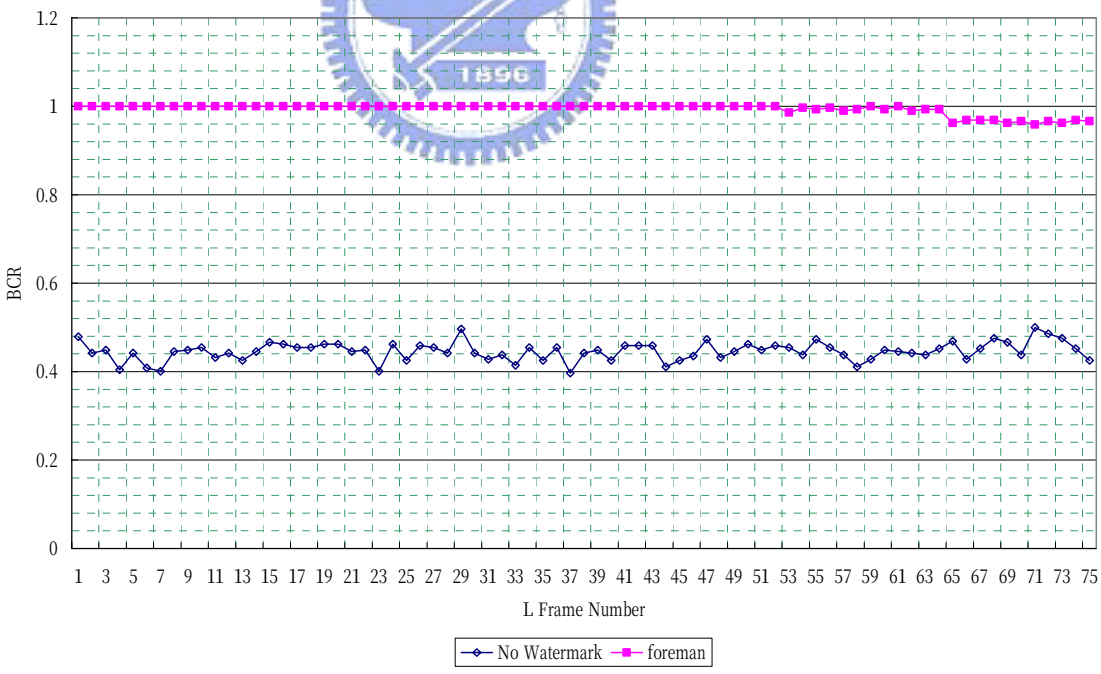
取出浮水印的 Threshold: 0.5

圖 4.4 到 圖 4.6 我們針對不同的解析度、位元速率和量化區間做了實驗，實驗的設定是以兩次的時間和兩次的空間轉換來編碼，在經過一次的時間編碼後 (T = 0)，再經過視訊解碼會產生和原來影片相同影格數量的影片，經過兩次的時間編碼後 (T = 1)，再經過視訊解碼會產生和原來影片一半影格數量的影片。 在經過

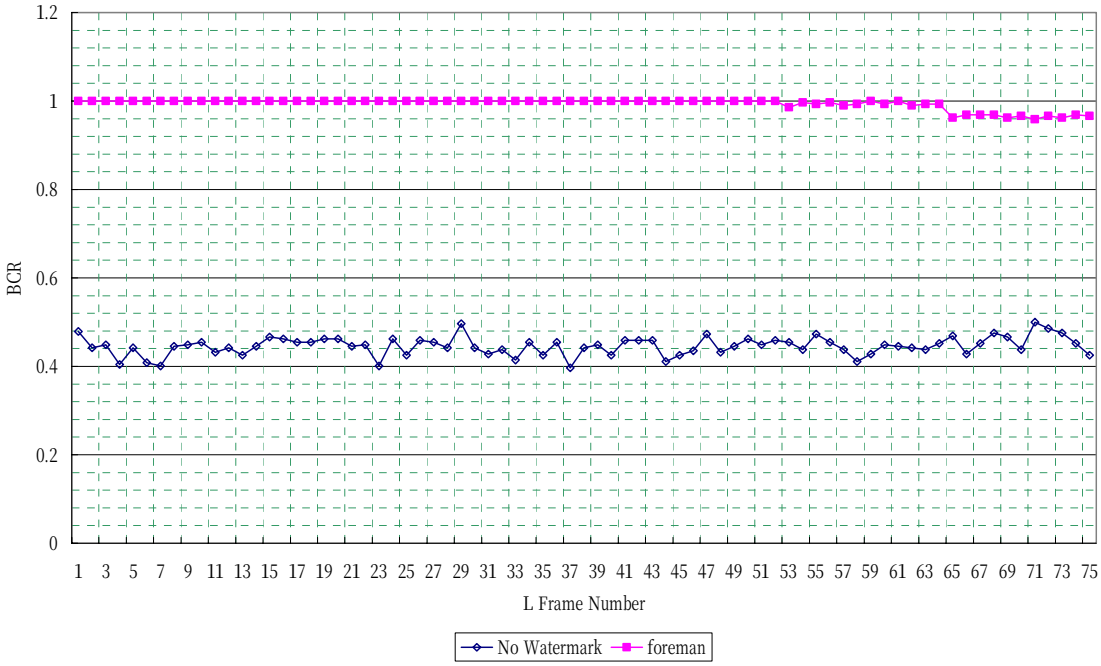
一次的空間編碼後 (S = 0)，再經過視訊解碼會產生和原來影片相同解析度的影片，經過兩次的空間編碼後 (S = 1)，再經過視訊解碼會產生和原來影片長寬解析度均為一半的影片。

4.1.1 不同解析度的實驗數據

在不同解析度的實驗中，我們使用一次的編碼，在解碼時去解出兩種不同解析度的影片，圖 4.4(a) 的解析度是 176x144，圖 4.4(b) 的解析度是 352x288，兩個影片都是在經過時間維度的兩次轉換，所以在解碼出來的影片每秒的影格數量只有原本的一半 (15 frames/sec)。由於在嵌入浮水印時是以 L frame 做為嵌入的基準，所以在經過兩次時間維度的轉換後，實際嵌入浮水印 L frame 只佔原本影格數量的 ¼。 我們設定相同的影格速率 (1k bits/sec) 來實驗，由圖 4.4 可以看出在我們嵌入浮水印的演算法中，不同的解析度下對浮水印的辨識度並沒有差異。



(a)解析度 176x144

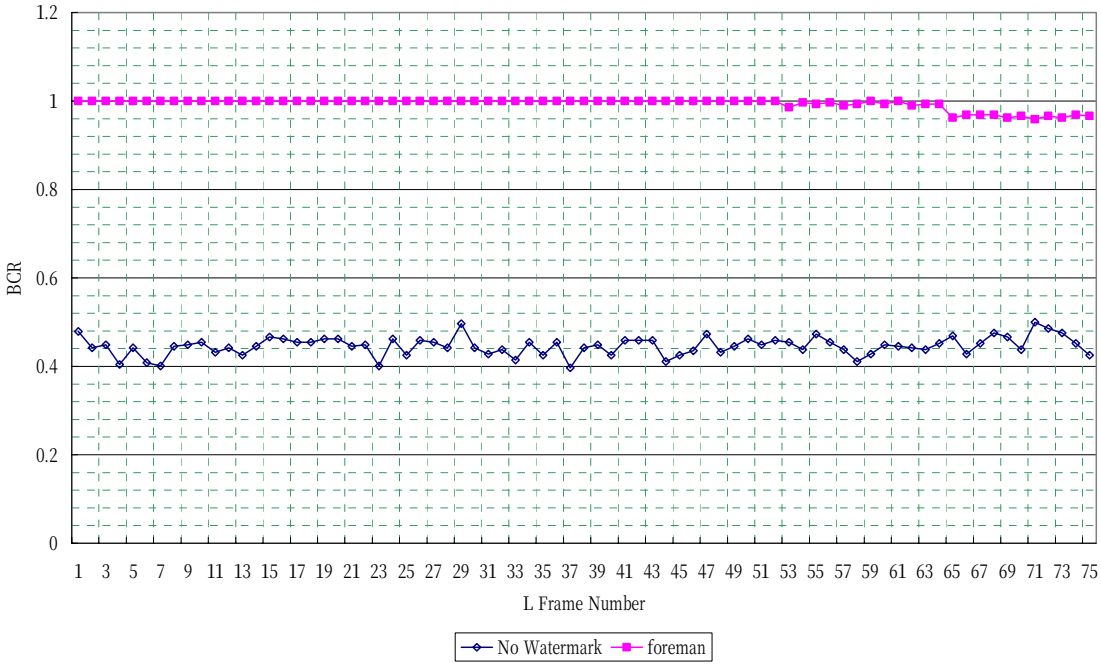


(b)解析度 352x288

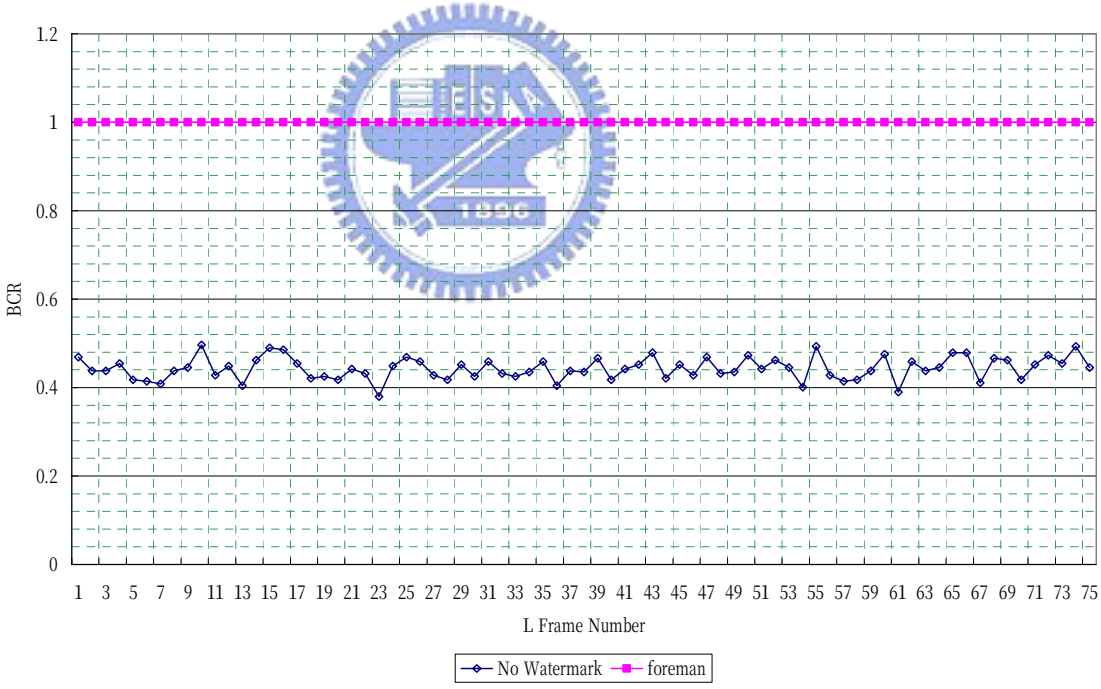
圖 4.4 不同解析度下浮水印的偵測比較

4.1.2 不同影格速率的實驗數據

在這個實驗中，我們使用兩種不同的影格速率來表示在不同的網路頻寬下，使用本論文所提出的演算法做偵測的結果。這兩個實驗的影片都是經過兩次的時間編碼後 (T = 1)，再經過一次的空間編碼後 (S = 0)，視訊解碼會產生和原來影片相同解析度，影格數量為原影片一半。在圖 4.5 我們可以看出在不同的影格速率下的 PSNR 會有不同，愈大的影格速率所包含影格較細微的資訊愈多，浮水印的資訊也就愈多，所以在 PSNR 的值就愈高，視覺感受也就愈好，偵測浮水印的準確率也就愈高。



(a)Bit Rate: 1k　PSNR: 38.6



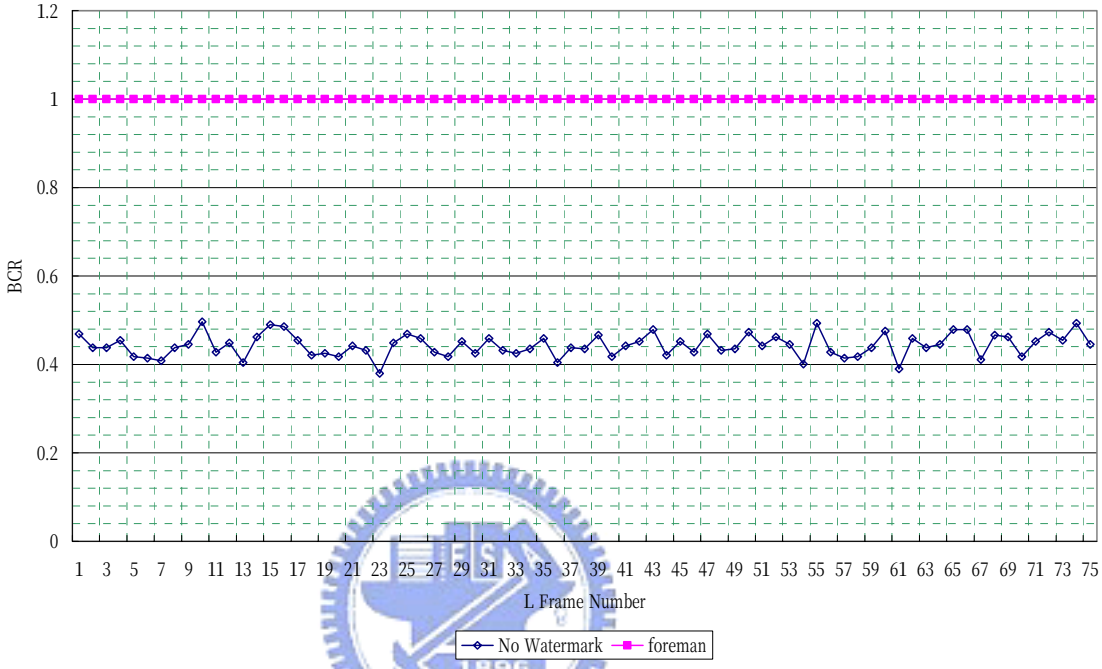
(b)Bit Rate: 4k　PSNR: 40.64

圖 4.5 不同速度下浮水印的偵測比較

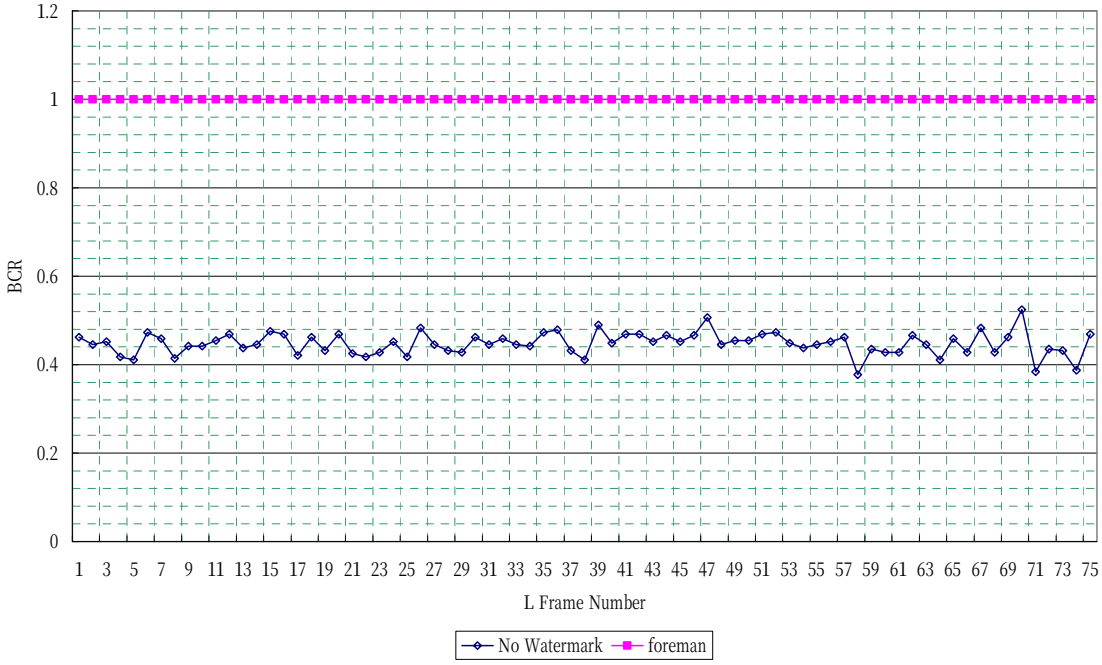
4.1.3 不同量化區間的實驗數據

量化區間在本論文的演算法中，代表著浮水印的強固性，使用不同的量化區間來嵌入浮水印，觀察在增加浮水印的強固性後，對 PSNR 的影響。使用的影片

是經過一次時間和空間的轉換 (S = 0, T = 0)，所產生和原影片完全相同的解析度和影格數量，在影格速率為 4k bits/sec 的下來做實驗。圖 4.6 可以看出量化區間 α 的值對 PSNR 是會有影響，而量化區間的值愈大則 PSNR 的值就愈小，視覺感受也會愈差，由圖 4.2 和圖 4.3 也可以看到相同的結果。



(a)α = 5　PSNR: 41.02



(b)α =7　PSNR: 38.16

圖 4.6 不同量化區間下浮水印的偵測比較

4.2 破壞浮水印的實驗數據

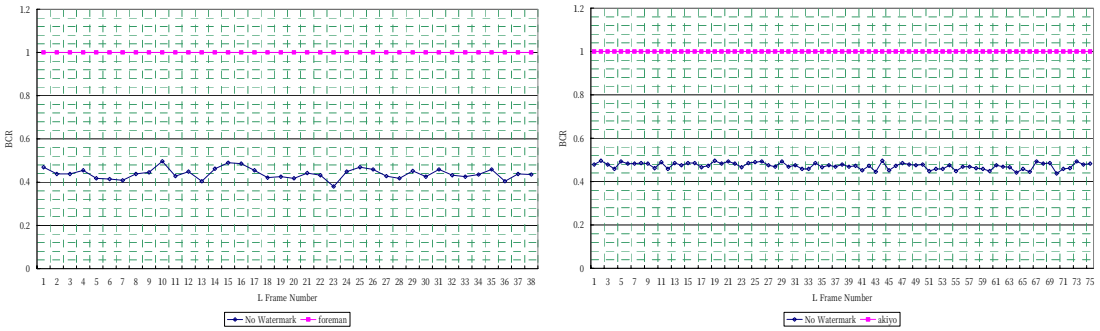
我使用丟棄影格和平均影格和移除平行線三種破壞浮水印的方法，作用在 foreman 和 akiyo 兩個不同的影片如圖 4.7。

(a) foreman (b)akiyo

圖 4.7 破壞浮水印的樣本影片

4.2.1 丟棄影格破壞浮水印

圖 4.8 使用丟棄影格的破壞方式，我們可以看出在不同的影片中，對丟棄影格的破壞有相同的偵測能力。

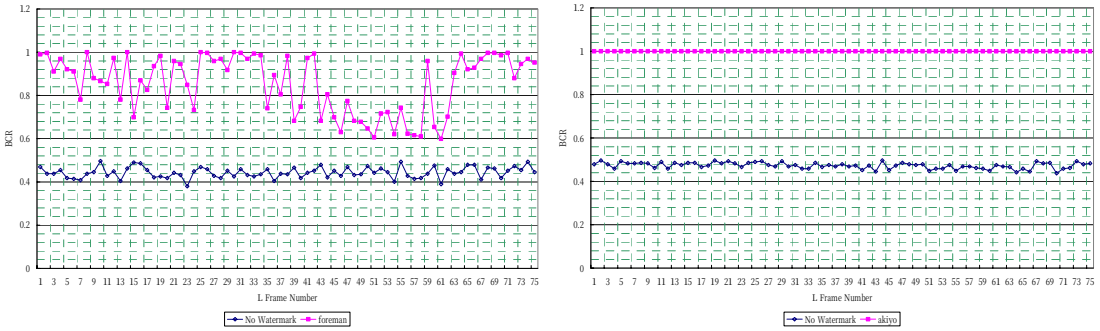


(a) foreman (b) akiyo

圖 4.8 丟棄影格破壞後浮水印的偵測比較

4.2.2 平均影格破壞浮水印

平均影格的破壞方式是將影片中相鄰兩個影格相加除以二，如此企圖將浮水印破壞。圖 4.9 使用平均影格的破壞方式，在這種破壞方式下，由於相鄰的兩個影格差異不大，所以在做過平均影格後，視覺上會感覺影片有一點模糊，所以在做浮水印偵測時所用到的 L frames 和原本的不同，但差別不大。



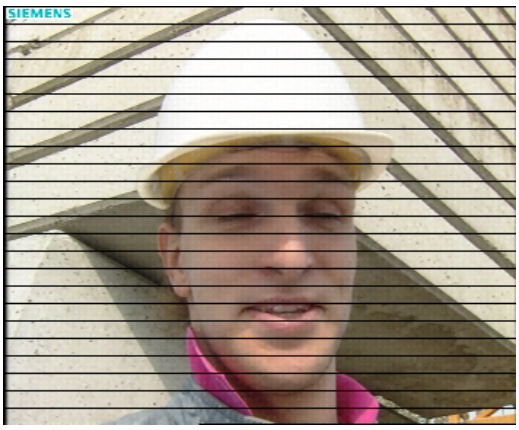
(a) foreman (b) akiyo

圖 4.9 影格平均破壞後浮水印的偵測比較

圖 4.9(a) 在經過破壞後還是有 60% 的可辨識率，但圖 4.9(b) 在經過破壞後仍然有很好的辨識率，其原因在於圖 4.7(b) 整體的移動較少，也就是影片大部份都是屬於靜止，對於破壞會有比較大的抗性。

4.2.3 移除平行線破壞浮水印

圖 4.10 是我們經過移除平行線後的影片，在攻擊中每 12 條平行線就移就一條，所以在測試的影片 foreman 和 akiyo 這兩個影片中我們的攻擊總共會移除 24 條平行線。

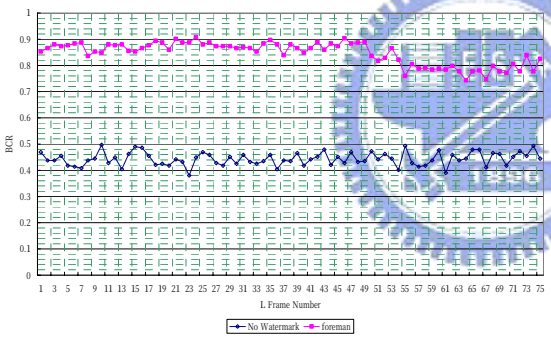


(a) foreman

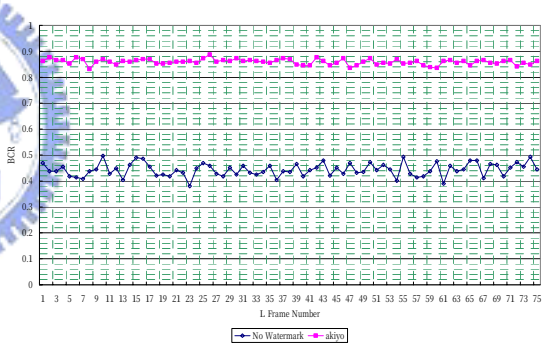
(b) akiyo

圖 4.10 移除平行線後的影片樣本

利用移除平行線的攻擊後，在圖 4.11 可看出偵測浮水印有很好的效果。不過在 akiyo 的影片經過破壞後，我們還是可以看出偵測浮水印時會比較穩定。



(a) foreman



(b) akiyo

圖 4.11 移除平行線破壞後浮水印的偵測比較

在可調式視訊中，有些破壞不一定是人為的，而是在網路上的封包遺失，或是當網路碰撞太多時封包遞送的時間太長，以至於當封包到達時已失去了解碼的時效性。要解決這樣的問題，可以使用服務品質 (QoS) 的方式來確保網路上有足夠的頻寬，如此可降低發生封包遺失的問題，也可以當所要求的影片下載的檔案大小到達一定的比例後，才做解碼的動作，並且使用 TCP 的方式做傳輸，才能確定在封包遺失時會有重送的機制。

五、結論和未來展望

本論文所提出的演算法經過實驗後可以看出，嵌入浮水印時量化區間係數的大小決定了浮水印的強度，而在固定量化區間係數 α 的情形下，位元速率的大小和影片的視覺感受成正比，也就是說若能提供的網路頻寬愈大，視覺感受就愈好。但浮水印的強度並不會因為在不同的解析度下，而有所不同。對於破壞方面來說，由實驗數據可以知道，在影片本身移動較大時，移動向量的偏差也會較大，所以原影片在經過破壞後，在做移動評估時所決定的移動向量偏差就會變大，所以在破壞後偵測出浮水印的 BCR 就會比較低，如圖 4.7(a) 影片中的移動就比圖 4.7(b) 大很多，在經過破壞後的 BCR 圖 4.7(a) 就明顯比圖 4.7(b) 低。在本論文所提出的浮水印演算法，增加量化區間和安全密碼來加強浮水印的強固性，而且實作上也很容易，相信對網路上所流通視訊的智慧財產權的保護，提供了另一種的選擇。

在目前所提出浮水印的演算法中，浮水印的強度和視覺上的感受是互相影響的，浮水印的強度愈強，視覺上的感受就愈差，這是一個兩難的問題，但是對於在移動劇烈的影片對破壞的抗性，是可以深入研究的，而浮水印嵌入的位置是否可以在移動評估之前，如此移動向量對浮水印的影響就會減少，對破壞後的抗性也會增加。

參考文獻

[1] Lin Luo, Feng Wu, Shipeng Li, Zixiang Xiong, Zhenquan Zhuang, “Advanced motion threading for 3D wavelet video coding,” Signal Processing: Image Communication, special issue on subband/wavelet video coding, 2004.

[2] Lin Luo, Feng Wu, Shipeng Li, and Zhenquan Zhuang, “Advanced lifting-based Motion-Threading (MTh) techniques for 3D wavelet video coding,” invited paper, to appear in SPIE/IEEE Visual Communications and Image Processing (VCIP2003), Lugano, Switzerland.

[3] Xiangyang Ji, Jizheng Xu, Debin Zhao, Feng Wu, “Architectures of incorporating MPEG-4 AVC into three-dimensional wavelet video coding,” Picture Coding Symposium, San Francisco, CA, USA, Dec 2004.

[4] L. Luo, J. Li, S. Li, Z. Zhuang, and Y.-Q. Zhang, "Motion compensated lifting wavelet and its application in video coding", IEEE Intl.Conf. on Multimedia and Expo(ICME 2001), Tokyo, Japan, Aug. 2001.

[5] Ruiqin Xiong, Feng Wu, Shipeng Li, Zixiang Xiong, Ya-Qin Zhang, “Exploiting temporal correlation with adaptive block-size motion alignment for 3D wavelet coding”, SPIE Visual Communications and Image Processing (VCIP), vol. 5308, pp 144-155, January 2004.

[6] L. Cheng, D. Liang, and Z. Zhang, "Popular biorthogonal wavelet filters via a lifting scheme and its application in image compression," IEE Proc. Vision, Image and Signal Processing, vol. 150, no. 4, pp. 227--232, August 2003.

[7] J. Xu, S. Li, Y.-Q. Zhang, and Z. Xiong, “A wavelet video coder using 3-D embedded subband coding with optimized truncation (3-D ESCOT),” in Proc. ISMIP'00,

Sydney, Australia, December 2000.

[8] J. Xu, Z. Xiong, S. Li, and Y.-Q. Zhang, “Three-dimensional embedded subband coding with optimized truncation (3D ESCOT)”, Applied and Computational Harmonic Analysis10, pp.290-315, 2001.

[9] Ying Li, Xinbo Gao, Hongbing Ji, “A 3D Wavelet Based Spatial-Temporal Approach for Video Watermarking,” Proceedings of International Conference on Computational Intelligence and Multimedia Applications, pp.260-265, Xi'an, China, Sept.27-30, 2003.

[10] A. Koz and A. Alatan, Video watermarking using temporal sensitivities of Human visual system, 4th COST #276 Workshop, Bordeaux, France, March-April, 2002.

[11] Xiamu Niu and Shenghe Sun, “A New Wavelet-Based Digital Watermarking for Video.” 9th IEEE Digital Signal Processing Workshop, Texas, USA, Oct. 2000.