第四章 Inter 預測編碼

在影像壓縮的領域，畫面是由背景（Back Scene）加上景物（Object Scene）所構成。動態影像是由連續的畫面所構成，連續的畫面之間背景不變或者極爲近似，而景物則以規則性的方式（例如相同的速度與方向）移動。如此的特徵有利於動態影像的壓縮，利用畫面背景近似以及畫面景物的規則性移動等特質以提高壓縮率的方式，我們叫做 Inter 預測編碼（Inter Prediction Coding）。Inter 預測編碼的工作順序，見圖 4-1。

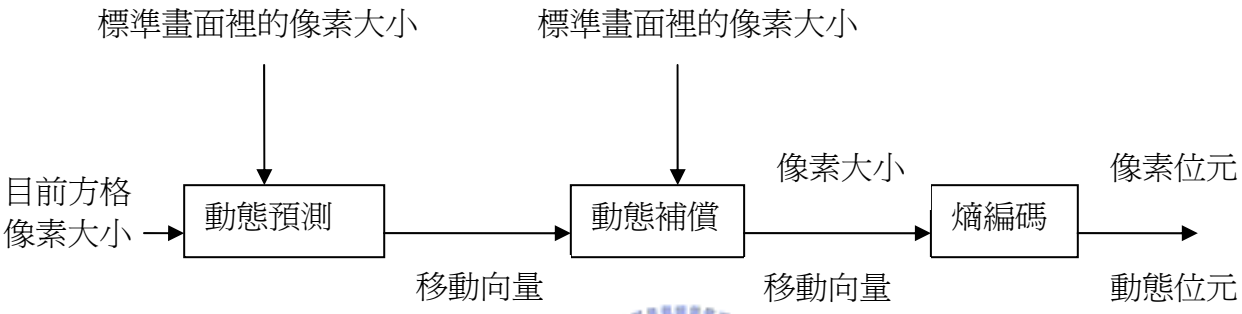

圖 4-1 Inter 預測編碼方塊圖

Inter 預測後產生描述方格動態的移動向量與像素大小，編碼後分別產生動態位元與像素位元。使用大方格預測產生的動態位元數少，而像素位元數多；小方格預測則得到相反的結果。H.263，H.264，MPEG4 的語法允許每個區塊以不同的方格大小進行動態預測，並且在像素位元數與動態位元數之間權衡，務必使得編碼結果產生最少的位元總數。使用不同的方格大小進行動態預測就是變化方格大小動態預測（Variable Block Size Motion Estimation，簡稱 VBS）；使用相同的方格大小進行動態預測就是固定方格大小動態預測（Fixed Block Size Motion Estimation，簡稱 FBS）。

本章介紹並且模擬 H.263++ Annex F〔29〕，H.264〔30〕，以及 MPEG4〔31〕的動態預測。模擬分三個主題：

- 大方格與小方格動態預測的差別：
  觀察不同方格大小的動態預測對動態位元與像素位元數量的影響。
- FBS 與 VBS 的差別：
  觀察 VBS 比起 FBS 的壓縮增益。
- H.263++ Annex F，H.264，以及 MPEG4 的差別：
  不同動態預測方式，不同轉換方式，不同熵編碼方式的整體比較。

模擬結果顯示 VBS 產生比較少的位元總數。三個標準中 H.264 提供最多種不同方格大小的 VBS，壓縮率也最高；卻耗費最多的計算機運算次數。

4.1 變化與固定方格大小動態預測

每個方格進行動態預測（Motion Estimation），是將區塊切割成特定的方格，並且依此方格大小裡的畫面內容，從比對畫面（Reference Frame）中找尋與其最相似的部分。例如 8x8 的動態預測，就是將區塊裡大小 16x16 的明度方格切割成四個 8x8 的方格，然後四個方格獨自找尋與其最相似的部分；而 4x4 的動態預測，則是將 16x16 的明度方格切割成 16 個大小 4x4 的方格，然後十六個方格獨自找尋與其最相似的部分。

視訊壓縮標準定義動態預測的方格大小。當視訊壓縮標準定義畫面中所有的區塊必須以相同的方格大小進行動態預測，這種動態預測我們叫做固定方格大小動態預測（Fixed Block Size Motion Estimation，簡稱 FBS）。相對的，如果視訊壓縮標準定義區塊可以被切割成不同的方格大小來進行動態預測，完成動態預測後從不同的方格大小中擇其一來進行動態補償與後續的編碼，這種動態預測方式我們叫做變化方格大小動態預測（Variable Block Size Motion Estimation，簡稱 VBS）。

H.263，H.264，MPEG4 使用不同的方格切割方式，將大小 16x16 的明度方格切割成不同的方格大小以進行動態預測。H.263 是 FBS，畫面裡所有方格都以 16x16 的方格大小進行動態預測。H.263++裡的 Annex F（Advanced Prediction Mode）則定義了兩種方格大小的語法，允許預測編碼器以 8x8 及 16x16 兩種方格切割方式進行動態預測。因此 H.263++ Annex F 屬於 VBS。H.264 也是 VBS，H.264 的樹狀目錄切割方式語法定義七種不同的切割方式，使得方格經由切割後產生大小 4x4，4x8，8x4，8x8，8x16，16x8，16x16 的方格以進行動態預測。

4.1.1 大方格與小方格的權衡

從 H.263++ Annex F 與 H.264 所定義的語法我們可看出，不同的切割方式會產生不同的方格大小。而不同的切割方式也產生了不同的方格數量。同一個區塊以不同方格大小進行動態預測會產生不同結果。比較起大方格動態預測，小方格動態預測能從比對畫面中找到相似性高的畫面區域，因此經過動態補償後，像素差值小，像素位元數目少。相對的，小方格數量比大方格多，移動向量也多，小方格的動態位元數目比大方格多。

由此我們可知，使用不同大小方格進行動態預測，實際上是像素位元數目與動態位元數目之間的權衡。壓縮器以不同的方格大小進行動態預測，經由動態補償以及後續的編碼，找出像素位元數目以及移動向量位元總數最少的方格大小。變化方格大小動態預測的優點就是提供彈性做法，讓編碼的結果得到最少的位元總數。

4.1.2 計算機運算次數與壓縮率的權衡

變化方格大小動態預測雖然減少編碼後的位元總數，卻增加動態預測時的運算量。以下我們比較 H.263、H.263++ Annex F、H.264（H.263 是 FBS，Annex F 和 H.264 是 VBS）的動態預測，在一個相同大小的搜尋區域下的計算機運算量。

一、**H.263**（**FBS**）：

16x16 = 256

每個 H.263 區塊計算 **256** 次

二、**H.263++ Annex F**（**VBS**）：

16x16 切割方式：16x16 = **256**

8x8 切割方式：4 個方格 x 8x8 = **256** **256** + **256** = **512**

每個啓動 Annex F 的 H.263++區塊計算 **512** 次

三、**H.264**（**VBS**）：

16x16 切割方式：16x16 = **256**

16x8 切割方式：2 個方格 x 16x8 = **256**

8x16 切割方式：2 個方格 x 8x16 = **256**

8x8 切割方式：4 個方格 x 8x8 = **256**

8x4 切割方式：8 個方格 x 8x4 = **256**

4x8 切割方式：8 個方格 x 4x8 = **256**

4x4 切割方式：16 個方格 x 4x4 = **256**

256 + 256 + 256 + 256 + 256 + 256 + 256 = **1792**

每個 H.264 區塊計算 **1792** 次

很明顯的，VBS 的運算量比 FBS 大。而切割方式愈多，產生的方格大小種類愈多，計算機運算量也愈大。

4.2 H.263++ AnnexF 與 MPEG4 變化方格大小預測編碼

H.263++裡的 Annex F（Advanced Prediction Mode）〔29〕定義了變化方格大小動態預測以及移動向量的差分編碼方式（Differential Coding）。當編碼器啓動 Annex F 作爲編碼的模式，每個方格可以選擇進行 16x16 或 8x8 方格大小的動態預測，因此 Annex F 是 VBS。MPEG4 也是 VBS 動態預測，並且定義了 16x16 與 8x8 兩種方格大小。

4.2.1 Intra 或 Inter 預測判斷方式

P 畫面中的區塊可進行 Intra 預測或 Inter 預測，TMN8〔17〕定義了 Annex F 啓動時 P 畫面裡區塊的 Intra 或 Inter 預測判斷方式。首先分別以 8x8 方格以及 16x16 方格進行動態預測。完成後分別計算兩者的絕對值總合差（Sum of Absolute Difference），*SAD*8 與 *SAD*16 。絕對值總合差是目前方格與比對區域的像素差的絕對值總合。接下來的判斷分三個步驟，如下：

步驟一：Intra 或 Inter 預測的判斷

步驟一判斷目前區塊要進行 Intra 還是 Inter 預測，公式如下：

$$W < \min\{SAD8, SAD16\} - 500 \qquad (4.1)$$

W 是目前目前區塊裡明度方格的絕對值總合差，其算法是區塊裡 256 個明度像素與區塊的明度方格平均值相減取絕對值，再相加取總合。*SAD*8 與 *SAD*16 取其小者，減 500，再與 *W* 相比，若大於此值，就進行 Intra 預測，進入步驟二；反之，進行 Inter 預測，進入步驟三。

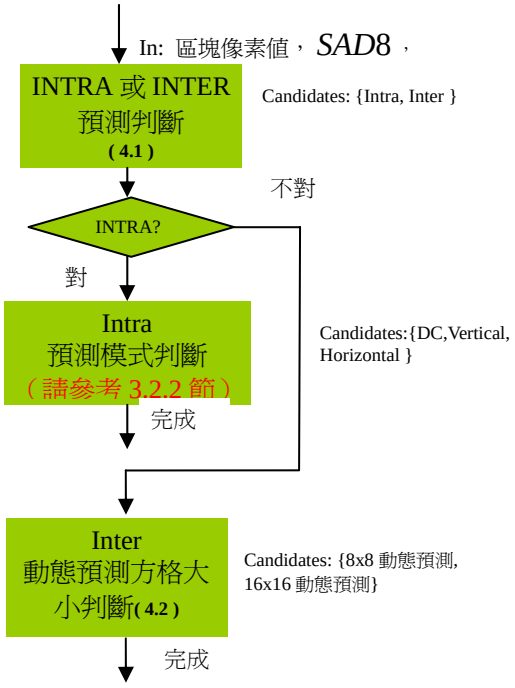

圖 4-2 P 畫面裡的區塊採用 Inter 或 Intra 預測的判斷流程（TMN8）

步驟二：Intra 預測模式的選擇

如果畫面啓動 Annex I，便進行 Intra 預測模式的選擇；反之，直接進行編碼，不進行預測。Intra 預測模式的選擇請參考 3.2.2 節。

步驟三：Inter 預測模式的選擇

步驟三判斷目前區塊要進行 8x8 還是 16x16 動態預測，公式如**(4.2)**。

$$SAD8 < SAD16 - 200 \qquad \textbf{(4.2)}$$

若以上的公式成立，使用 8x8 動態預測；反之，使用 16x16 動態預測。

4.2.2 Annex F 編碼方式

4.2.2.1 Annex F 畫面標頭

Annex F 是 H.263 的附屬功能，編碼器需要在畫面標頭以特別的位元告知解碼器 Annex F 是否啓動。當 Annex F 啓動，畫面層級的標頭內容（Frame Layer Header）會顯示此畫面爲 Annex F 編碼模式，標頭內容如表 4-1。〔25〕

表 4-1 Annex F 畫面標頭

欄位名稱	號碼	位元數
Picture Start Code（PSC）	000000000xxx1	22
TR	x	8
PTYPE	xxxxx**111**	8
PLUSPTYPE	xxx**1**x	**5**
UFEP	x	**3**
OPPTYPE	x	**18**
MPPTYPE	x	**9**
CPCFC	x	8
ETR	x	2
UUI	x	變化長度碼
SSS	x	變化長度碼
ELNUM	x	變化長度碼
PQUANT	x	5

當編碼器選擇啓動 Annex F 作爲編碼的模式，欄位 **PTYPE** 的後三個位元是”111”，而 **PLUSPTYPE** 的第四個位元是 1。當編碼器選擇啓動 Annex F 作爲編碼的模式，畫面層級的標頭內容會多出 **PLUSPTYPE**，**UFEP**，**OPPTYPE** 以及 **MPPTYPE** 四個欄位。因此當編碼器選擇啓動 Annex F 作爲編碼的模式，**每張畫面會多出 35 個位元。**

4.2.2.2 Annex F 區塊標頭

區塊層級標頭裡的 MCBPC 欄位標明動態預測的方格大小。當區塊以 16x16 方格大小進行動態預測，產生一個移動向量，MCBPC 名稱是 INTER 或 INTER + Q。MCBPC 採用變化長度編碼，依據 CBPC（Chrominance Block Pattern Code，色度編碼）與量化參數變化與否，有不同的二進位代號，見表 4-2。〔26〕

表 4-2 16x16 動態預測時的 MCBPC 二進位代號

比起前一區塊的量化參數，目前區塊的量化參數是否改變?	CBPC	二進位代號（Codeword）	代號長度
否 （INTER）	00	1	1
	01	0011	4
	10	0010	4
	11	000101	6
是（INTER + Q）	00	011	3
	01	0000111	7
	10	0000110	7
	11	000000101	9

當區塊以 8x8 方格大小進行動態預測，總共產生四個移動向量，MCBPC 名稱是 INTER4V 或 INTER4V + Q。此時的 MCBPC 二進位代號與其長度請見表 4-3。〔26〕

表 4-3 8x8 動態預測時的 MCBPC 二進位代號

比起前一區塊的量化參數，目前區塊的量化參數是否改變?	CBPC	二進位代號（Codeword）	代號長度
否 （INTER4V）	00	010	3
	01	0000101	7
	10	0000100	7
	11	00000101	8
是（INTER4V + Q）	00	00000000010	11
	01	0000000001100	13
	10	0000000001110	13
	11	0000000001111	13

由表 4-2 與表 4-3，比較起 16x16 動態預測，8x8 動態預測在區塊標頭多出 2 到 12 個位元。

MPEG4 以欄位 MCBPC 告知解碼器目前方格是否使用 8x8 動態預測。當 MB type 號碼是 2(Inter 4V)，表示目前方格使用 8x8 動態預測。根據 CBPC 的不同，MCBPC 有不同的代號，見表 4-4：

表 4-4 MPEG4 中 MCBPC

MB type Index	CBPC	二進位代號（Codeword）	代號長度
2 (Inter 4V)	00	010	3
	01	0000101	7
	10	0000100	7
	11	00000101	8

我們可看出 MPEG4 與 H.263++ Annex F 的語法以及欄位號碼完全相同，只是 MPEG4 沒有提供 Inter 4V + Q。也就是說，當 MPEG4 的 Inter Prediction Coder 選擇四個移動向量的動態預測，就不能改變量化值，不能進行方格層級的速率控制（Rate Control）。

4.3 H.264 變化方格大小預測編碼

H.264 將區塊切割成七種方格大小以進行動態預測〔30〕。H.264 依據這七種方格大小進行動態預測與補償，從七種方格中找出最少位元總數的編碼方式，並依此方格大小所計算出的移動向量，像素差進行編碼。

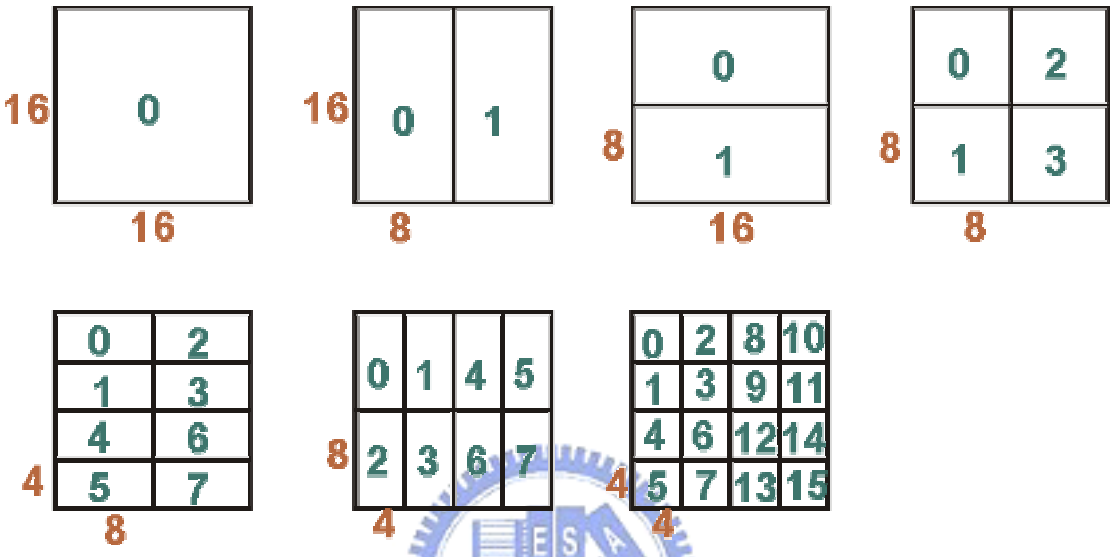

圖 4-3 H.264 的區塊切割後的七種方格大小

七種切割方式將區塊切割成 16x16，16x8，8x16，8x8，8x4，4x8，4x4 等七種方格大小，如圖 4-3 所示。圖中橙色數字代表方格經由樹狀目錄切割語法切割後的方格大小，綠色數字則代表切割後各方格的編號。預測編碼以此編號爲順序，進行動態預測以及後續的編碼工作。

4.3.1 H.264 Intra 或 Inter 預測判斷方式

P 畫面裡區塊的 Intra 或 Inter 預測判斷方式，請參考 3.1.2.2 節。編譯器選擇最少拉氏總合的預測模式作爲區塊的預測模式。

4.3.2 H.264 區塊語法

H.264 區塊層級的標頭，由 **MB TYPE** 與 **SUB MB TYPE** 欄位顯示區塊切割後的方格大小。根據不同的 **MB TYPE** 與 **SUB MB TYPE**，H.264 再由欄位 **ref_idx [mbPartIdx]**以及函數 **mvd[mbPartIdx]**顯示每個方格內的移動向量〔14〕〔15〕。見圖 4-4。

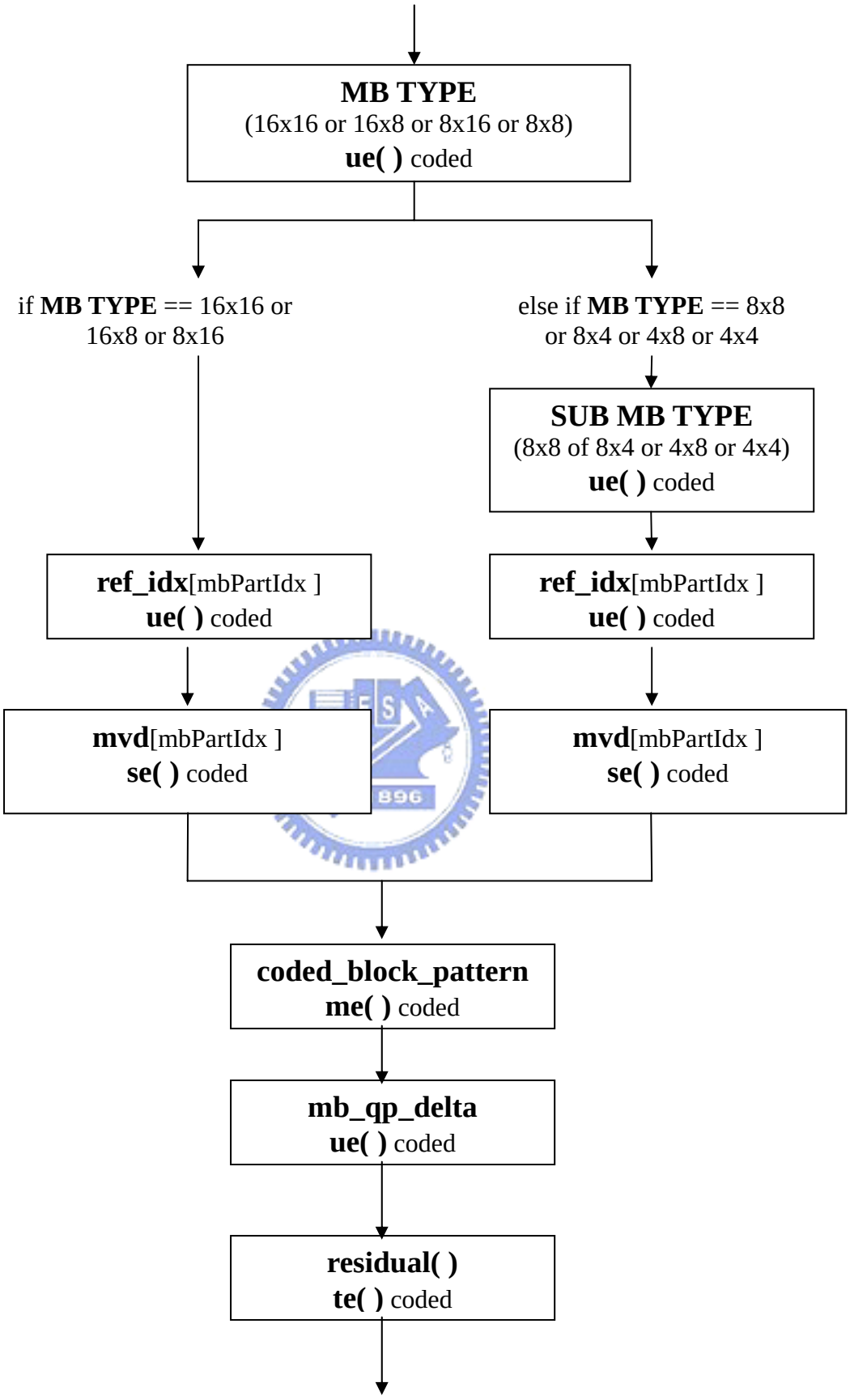

圖 4-4 H.264 裡 Inter 區塊語法

圖 4-4 每一個欄位根據索引編號以及 **ue( )**、**se( )**、**me( )**得到 Exp – Golomb Codeword。**ue( )**、**se( )** 、**me( )**以及 Exp – Golomb Codeword 的定義請參考 2.2.3.2 節。現在分別介紹圖 4-4 中每一個欄位的語意。

MB TYPE

MB TYPE 顯示區塊的型態、區塊的預測方式以及區塊裡明度與色度方格的型態。P 畫面裡的 Inter 區塊，**MB TYPE** 索引編號從 0 到 3；這四個索引號碼分別代表區塊被切割成 16x16，16x8，8x16，8x8 大小的方格進行動態預測。如果區塊以小於 8x8 的大小進行動態預測，**MB TYPE** 索引編號等於 3，然後再用 **SUB MB TYPE** 欄位描述方格的大小。有關 **SUB MB TYPE** 的語意請見下面的敘述。表 4-5 顯示 **MB TYPE** 名稱、相對應的索引編號、Exp – Golomb Codeword 以及位元長度。當切割後的方格大小是 16x16，16x8，或 8x16，**MB TYPE** 名稱分別爲 **P_L0_16x16**， **P_L0_L0_16x8**，以及 **P_L0_L0_8x16**。

表 4-5 P 畫面裡 Inter 區塊的 MB TYPE 索引與 Exp –Golomb Codeword

索引編號	MB TYPE	切割後的方格大小 (X 方向)	切割後的方格大小 (Y 方向)	Exp – Golomb Codeword	位元長度
0	**P_L0_16x16**	16	16	1	1
1	**P_L0_L0_16x8**	16	8	010	3
2	**P_L0_L0_8x16**	8	16	011	3
3	**P_8x8**	8	8	00100	5
4	**P_8x8ref()**	8	8	00101	5

P 畫面裡的 Intra 區塊，**MB TYPE** 索引編號從 5 到 29；各索引編號的語意與 I 畫面裡的 Intra 區塊相同，只是索引編號相差 5。見表 4-6。

表 4-6 P 畫面裡 Intra 區塊的 MB TYPE 索引與 Exp –Golomb Codeword

索引編號	MB TYPE	預測模式（Intra4x4 或 Intra16x16）	預測模式	Coded Block Pattern Chroma	Coded Block Pattern Luma	Exp – Golomb Codeword	位元長度
5	**I_4x4**	Intra_4x4	na	na	na	00110	5
6	**I_16x16_0_0_0**	Intra_16x16	0	0	0	00111	5
7	**I_16x16_1_0_0**	Intra_16x16	1	0	0	0001000	7
8	**I_16x16_2_0_0**	Intra_16x16	2	0	0	0001001	7
9	**I_16x16_3_0_0**	Intra_16x16	3	0	0	0001010	7
10	**I_16x16_0_1_0**	Intra_16x16	0	1	0	0001011	7
11	**I_16x16_1_1_0**	Intra_16x16	1	1	0	0001100	7
12	**I_16x16_2_1_0**	Intra_16x16	2	1	0	0001101	7
13	**I_16x16_3_1_0**	Intra_16x16	3	1	0	0001110	7
14	**I_16x16_0_2_0**	Intra_16x16	0	2	0	0001111	7
15	**I_16x16_1_2_0**	Intra_16x16	1	2	0	000010000	9

16	**I_16x16_2_2_0**	Intra_16x16	2	2	0	000010001	9
17	**I_16x16_3_2_0**	Intra_16x16	3	2	0	000010010	9
18	**I_16x16_0_0_1**	Intra_16x16	0	0	15	000010011	9
19	**I_16x16_1_0_1**	Intra_16x16	1	0	15	000010100	9
20	**I_16x16_2_0_1**	Intra_16x16	2	0	15	000010101	9
21	**I_16x16_3_0_1**	Intra_16x16	3	0	15	000010110	9
22	**I_16x16_0_1_1**	Intra_16x16	0	1	15	000010111	9
23	**I_16x16_1_1_1**	Intra_16x16	1	1	15	000011000	9
24	**I_16x16_2_1_1**	Intra_16x16	2	1	15	000011001	9
25	**I_16x16_3_1_1**	Intra_16x16	3	1	15	000011010	9
26	**I_16x16_0_2_1**	Intra_16x16	0	2	15	000011011	9
27	**I_16x16_1_2_1**	Intra_16x16	1	2	15	000011100	9
28	**I_16x16_2_2_1**	Intra_16x16	2	2	15	000011101	9
29	**I_16x16_3_2_1**	Intra_16x16	3	2	15	000011110	9

SUB MB TYPE

當區塊以 8x8、8x4、4x8、或 4x4 的方格大小進行動態預測，**MB TYPE** 名稱是 P_8x8，索引編號是 3。然後再用欄位 **SUB MB TYPE** 描述切割方式，方格大小 8x8，8x4，4x8，4x4，**SUB MB TYPE** 名稱分別是 **P_L0_8x8** ，**P_L0_8x4**，**P_L0_4x8**，以及 **P_L0_4x4**。**SUB MB TYPE** 名稱、索引編號、**Exp – Golomb Codeword** 與位元長度如表 4-7。

表 4-7 8x8、8x4、4x8、或 4x4 動態預測的 SUB MB TYPE

索引編號	**SUB MB TYPE**	**切割後的方格大小 (X 方向)**	**切割後的方格大小 (Y 方向)**	**Exp – Golomb Codeword**	**位元長度**
0	**P_L0_8x8**	8	8	1	1
1	**P_L0_L0_8x4**	8	4	010	3
2	**P_L0_L0_4x8**	4	8	011	3
3	**P_4x4**	4	4	00100	5

ref_idx[mbPartIdx]與 **mvd[mbPartIdx]**

當區塊是 Inter 區塊，便利用函數 **ref_idx[mbPartIdx]**與 **mvd[mbPartIdx]**顯示方格比對的畫面編號與移動向量。函數中的 **mbPartIdx** 表示方格編號，方格編號的最大值從 0 到 15。**ref_idx[mbPartIdx]**表示比對的畫面編號。**mvd[mbPartIdx]** 使用 DPCM 移動向量編碼，再根據函數 se( )得到 Exp – Golomb Codeword。

DPCM 移動向量編碼

Inter 區塊的方格數量多，移動向量多，會佔去數量龐大的動態位元數，因此採用 DPCM 移動向量編碼。DPCM 移動向量編碼與鄰近 Inter 區塊裡的方格的移動向量相減以減少動態位元數。〔32〕

圖 4-5 顯示目前區塊 E、左側區塊 A、上側區塊 B、右上側區塊 C，四個區塊之

間的幾何位置。當四個區塊都切割成 16x16 的方格進行動態預測，動態向量編碼方式是區塊 E 動態向量減掉區塊 A 區塊 B 區塊 C 的動態向量的中間值。

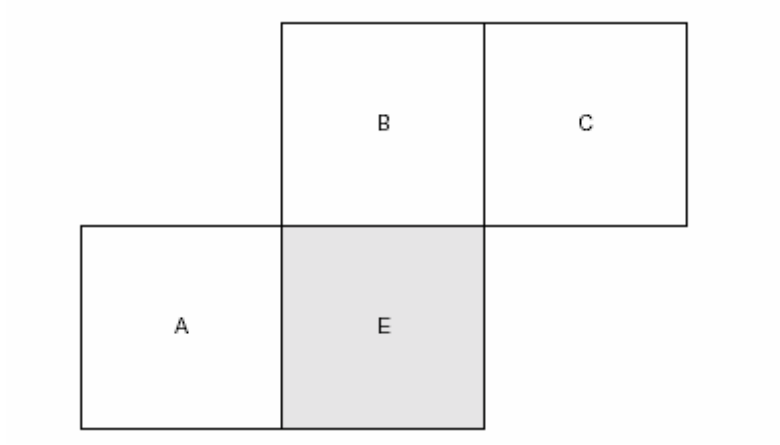

圖 4-5 目前區塊 E 與區塊 A、區塊 B、區塊 C 之間的位置

圖 4-6 顯示，當左側區塊、上側區塊、右上側區塊以 16x16 以外的方格大小進行動態預測時的移動向量編碼方式。此時左側區塊的右上方格定義爲方格 A、上側區塊的左下方格定義爲方格 B、右上側區塊的左下方格定義爲方格 C。移動向量編碼方式是區塊 E 移動向量減掉方格 A 方格 B 方格 C 的移動向量的中間值。

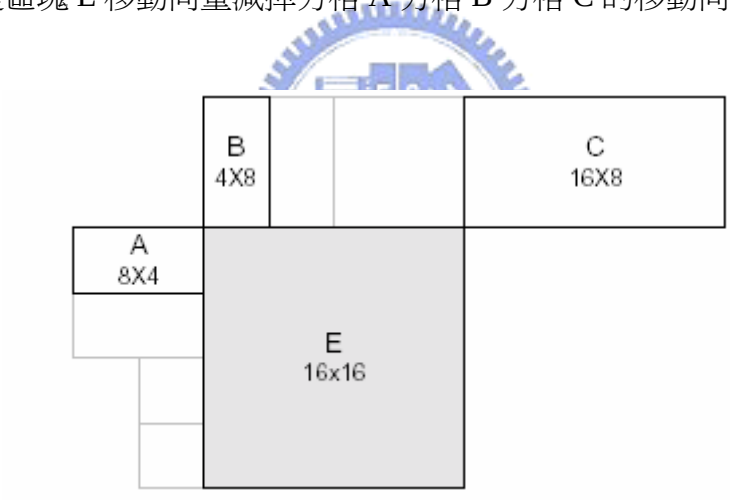

圖 4-6 區塊 A、區塊 B、區塊 C 的定義

以上皆討論區塊 E 以 16x16 進行動態預測時的移動向量編碼方式。現在討論其他切割方式時的移動向量編碼方式。當區塊 E 切割成 8x8，區塊 E 的 4 個動態向量減掉方格 A 方格 B 方格 C 的動態向量的中間值。方格 A 方格 B 方格 C 的動態向量定義請參考圖 4-5（方格大小 16x16）或圖 4-6（16x16 以外的方格大小）。

當區塊 E 切割成 16x8，區塊 E 的上側 16x8 方格動態向量減掉方格 B 的動態向量；區塊 E 的下側 16x8 方格動態向量減掉方格 A 的動態向量。方格 A 方格 B 的動態向量參考圖 4-5（16x16）或圖 4-6（16x16 以外的方格大小）。當區塊 E 切割成 8x16，區塊 E 的左側 8x16 方格動態向量減掉方格 A 的動態向量；區塊 E 的右側 8x16 方格動態向量減掉方格 C 的動態向量。方格 A 方格 C 的動態向量參考圖 4-5（16x16）或圖 4-6（16x16 以外的方格大小）。

4.4 模擬

模擬重點放在三項比較：

- 大方格與小方格動態預測下，動態位元數目與像素位元數目的不同。
  （4.4.1 節）

- FBS 與 VBS。
  （4.4.2 節與 4.4.3 節）

- 比較 MPEG4，H.263++ Annex F，與 H.264 的 Inter 預測整體表現。
  （4.4.4 節）

4.4.1 大方格與小方格動態預測

本節的模擬目的，在於觀察 FBS 下，大方格動態預測與小方格動態預測對動態位元數目與像素位元數目的影響。我們使用 H.263++ Annex F，TMN8〔17〕，以 16 x 16 FBS 爲大方格，8 x 8 FBS 爲小方格。測試序列使用十張 QCIF 的 Coastguard。（圖 4-7）

圖 4-7 Coastguard 樣本畫面

圖 4-8 顯示第一到第十張 Coastguard 的編碼結果。橫軸是畫面編號，縱軸是其編碼的位元數。紅線使用 8 x 8 FBS，藍線使用 16 x 16 FBS。圖 4-9 更進一步顯示兩種方格大小下，移動向量位元數目（Motion Bit）與像素位元數目（Residue Bit）。

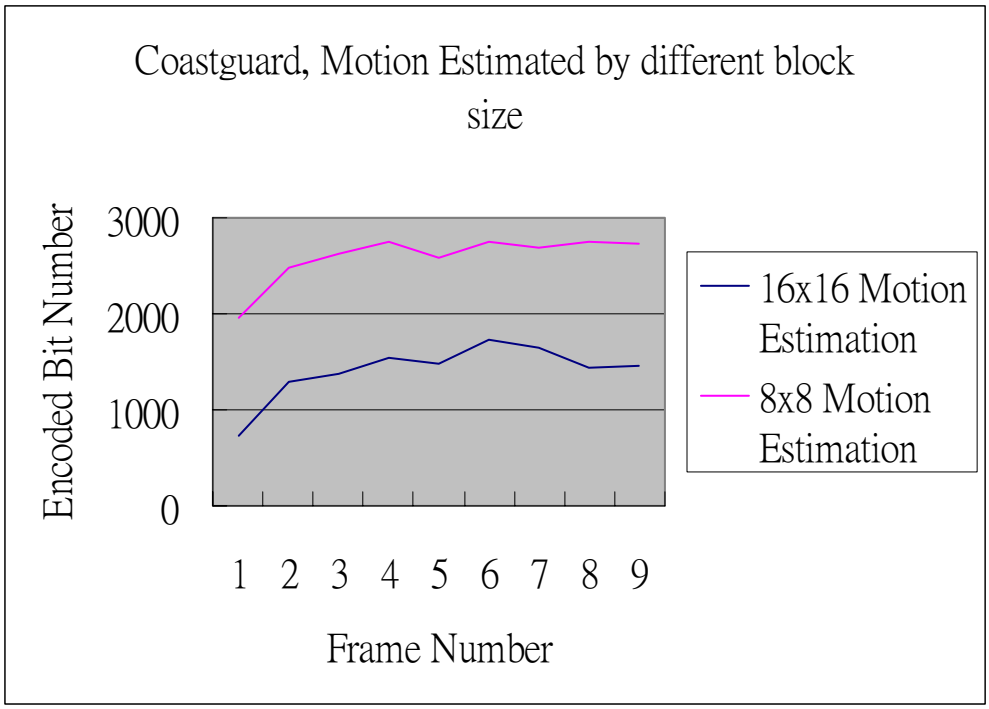

圖 4-8 各畫面位元總數

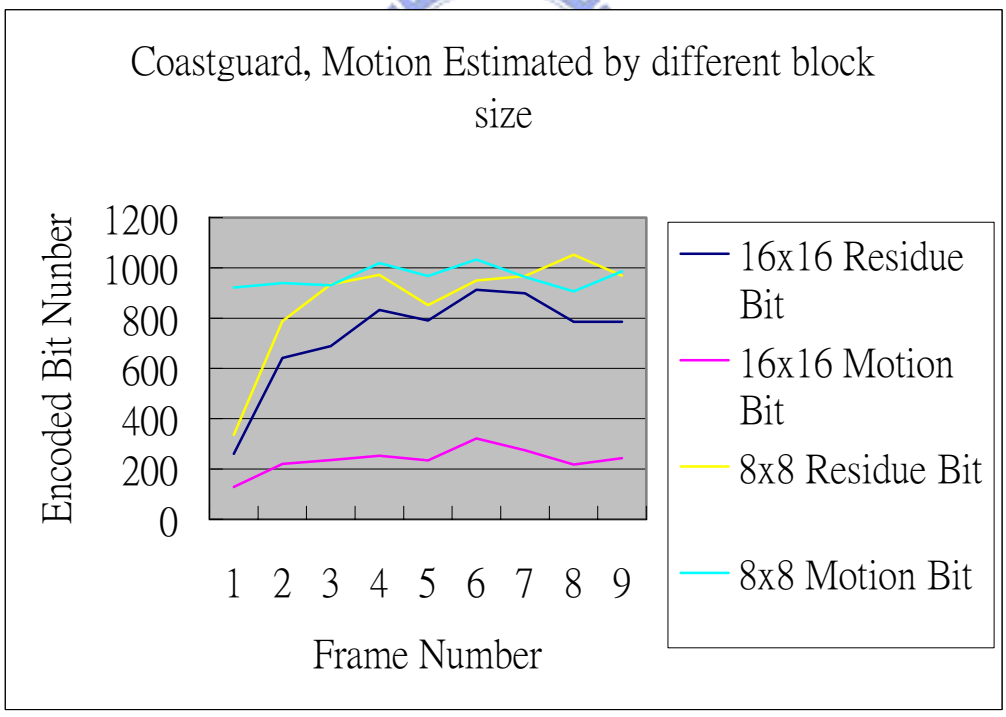

圖 4-9 各畫面動態位元總數與像素位元總數

圖 4-8 可看出大方格的位元總數比小方格少。圖 4-9 可解釋其原因。圖 4-9 裡紅線代表每張畫面中，大方格動態預測的動態位元數，從第一張到第十張大方格的動態位元數保持 200 個位元左右。至於藍線則代表每張畫面中，小方格動態預測的動態位元數，每張的動態位元數大約保持 900 個位元左右。大方格的動態位元數遠大於小方格的動態位元數，由此可見。

至於像素位元數，圖 4-9 深藍線代表每張畫面中，大方格動態預測的像素位元數，黃線代表每張畫面中小方格動態預測的像素。由圖中可見，兩個方格大小產生的像素位元差不如動態位元差大。由此我們可以簡略的作出一個結論：不同方格大小動態預測產生不同的動態位元數；動態位元數主導了壓縮的結果。

我們也可從圖 4-9 看到一個有趣的現象：兩條表示動態位元數的線，變動均不大，兩條表示像素位元數的線，起伏較大，而且走勢與圖 4-8 中第一張到第十張畫面的位元總數雷同。

4.4.2 H.264 變化與固定方格大小動態預測

本節以 H.264 JM7.1〔18〕比較 VBS 與 FBS 動態預測。我們選擇三種固定方格大小作爲 FBS：16 x 16， 8 x 8，與 4 x 4。VBS 動態預測則使用拉氏最佳化方法從七種方格大小中（4.2.2 節）找尋一種進行比對。

測試序列使用 Akiyo，Coastguard，Container 以及 Foreman，畫面樣本見圖 3-13。每個測試序列取一百張連續的 4：2：0 QCIF 畫面，並且以量化參數 10 到 50 調整各視訊壓縮標準在不同的壓縮比下的畫質表現。

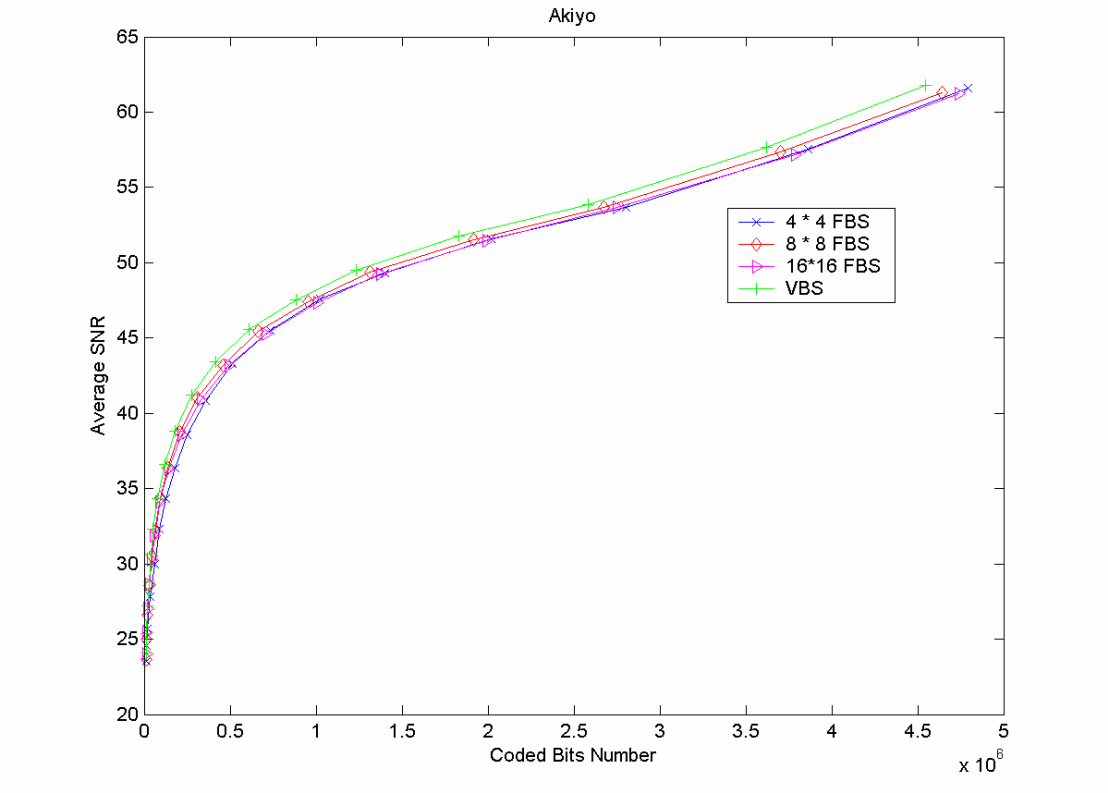

圖 4-10 Akiyo，H.264 FBS 與 VBS

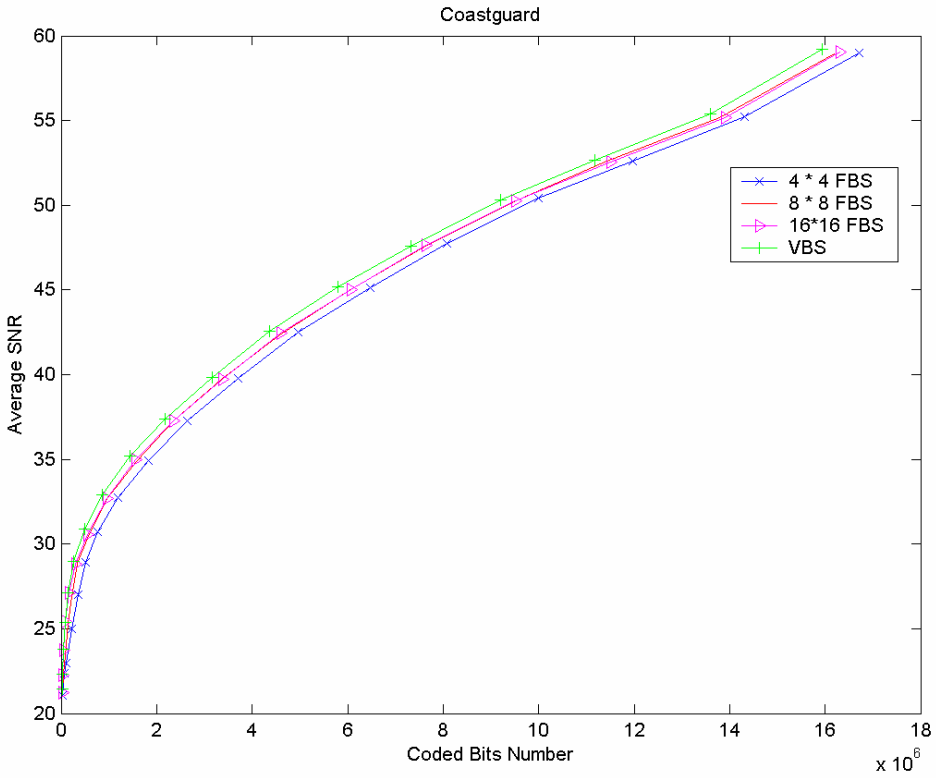

圖 4-11 Coastguard，H.264 FBS 與 VBS

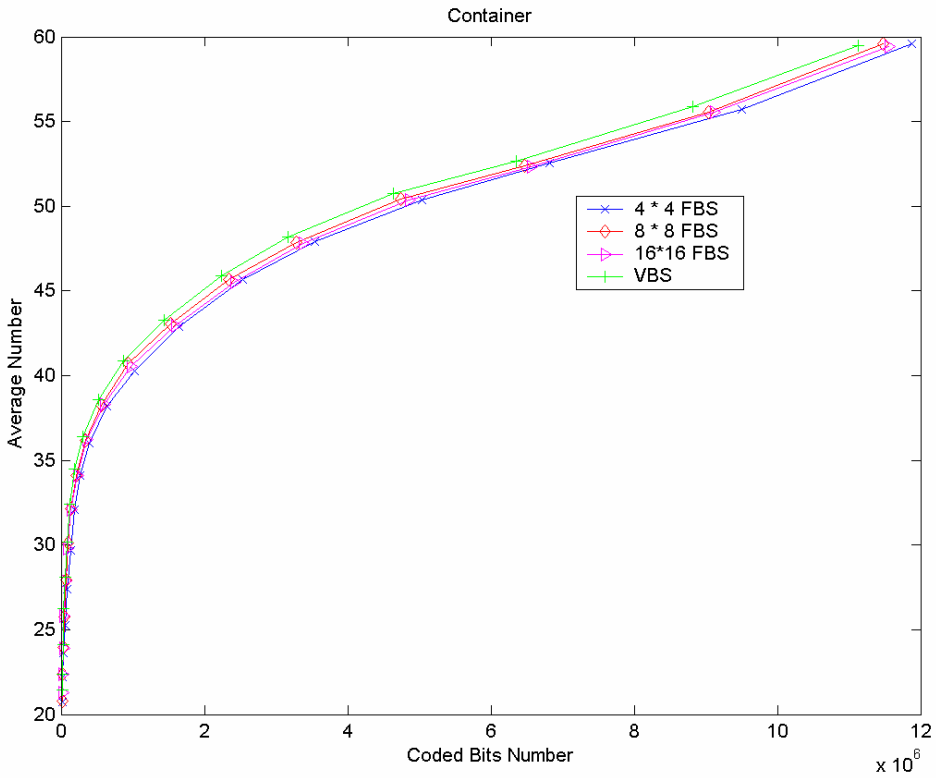

圖 4-12 Container，H.264 FBS 與 VBS

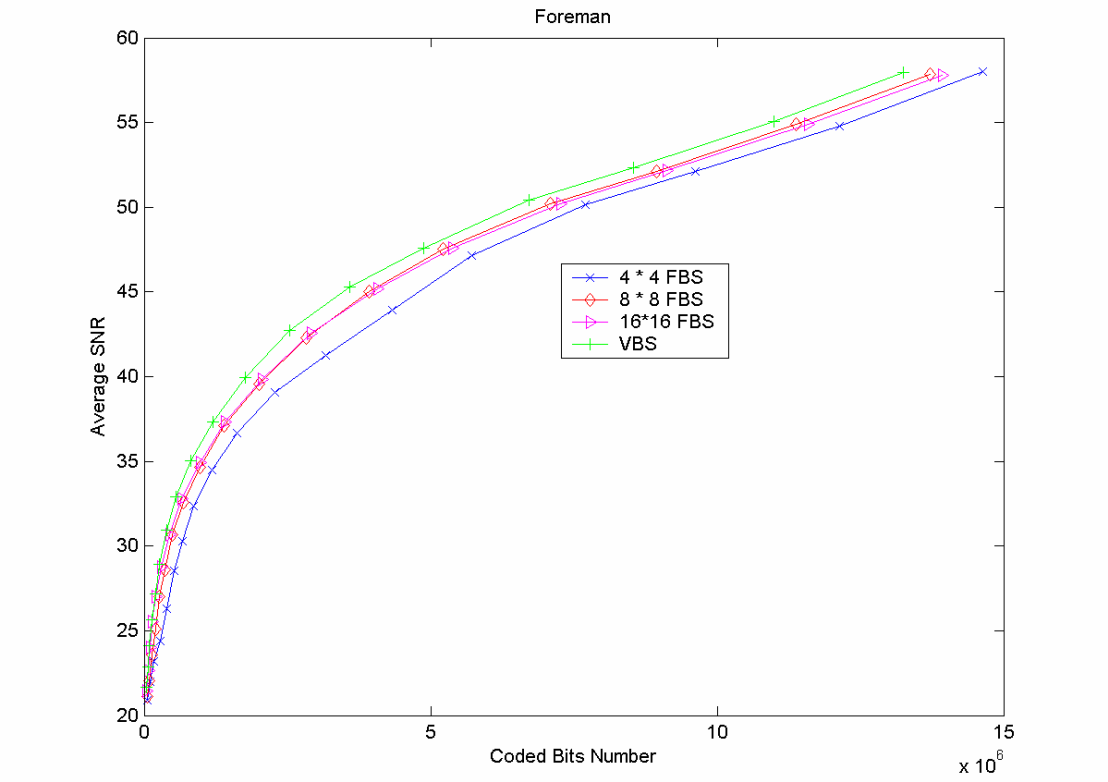

圖 4-13 Foreman，H.264 FBS 與 VBS

模擬結果分析

- **VBS 與三個 FBS 之間的比較：**

模擬結果顯示 VBS 比三種 FBS 都好（同樣位元數下，平均訊噪比較高），但差別不大。除了 Foreman，其他三個測試序列得到的模擬結果，VBS 與 FBS 的平均訊噪比差距都小於 1dB，肉眼無法分辨這樣的差距。從 4.1 節的分析我們知道 H.264 VBS 比任何一種 H.264 FBS 都高出七倍的計算機運算量，如此的代價卻只得來 1dB 的進展，我們不得不說如此的代價太高昂了。

16x16 切割方式：16x16 = **256**
16x8 切割方式：兩個方格 x 16x8 = **256**
8x16 切割方式：兩個方格 x 8x16 = **256**
8x8 切割方式：四個方格 x 8x8 = **256**
8x4 切割方式：八個方格 x 8x4 = **256**
4x8 切割方式：八個方格 x 4x8 = **256**
4x4 切割方式：十六個方格 x 4x4 = **256**

16x16，8x8 與 4x4 FBS 每個方格計算 **256** 次
VBS 每個方格計算 **1792** 次，是 FBS 的七倍

- **三種 FBS 之間的比較：**

三種 FBS 以小方格 4 x 4 壓縮表現最差，但與 8 x 8 或 16 x 16 差距很小。4.4.1 節的結果告訴我們 H.263 的小方格動態預測動態位元數目太大，導致整體表現不佳。H.264 使用更精確的移動向量差分編碼，大大減少了動態位元數目，因此拉低了小方格與大方格的差距。（4.4.1 節中，小方格的動態位元數目是大方格的 4.5 倍）

4.4.3 H.263 變化與固定方格大小動態預測

本節以 H.263++ Annex F 比較 VBS 與 FBS。FBS 使用 8 x 8 與 16 x 16。

模擬比較三種動態預測方式：

1. Annex F on，VBS；
2. Annex F on，8 x 8 FBS；
3. Annex F off，16 x 16 FBS。

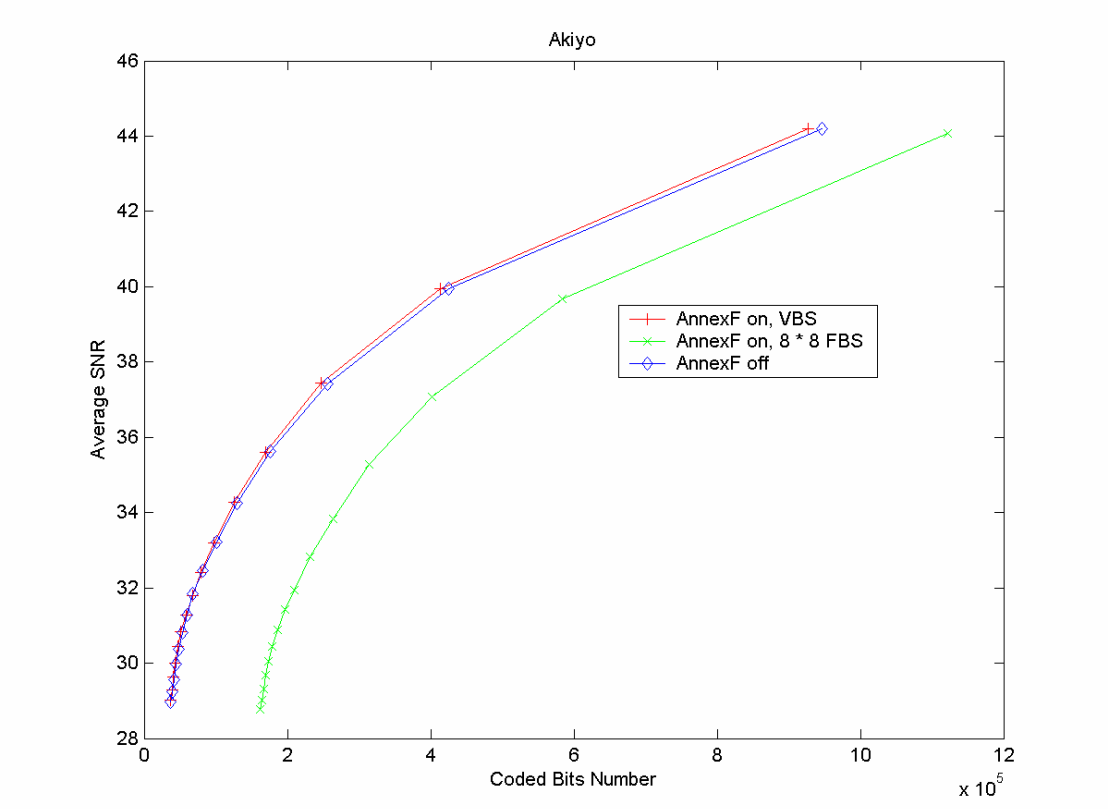

圖 4-14 Akiyo，H.263 FBS 與 VBS

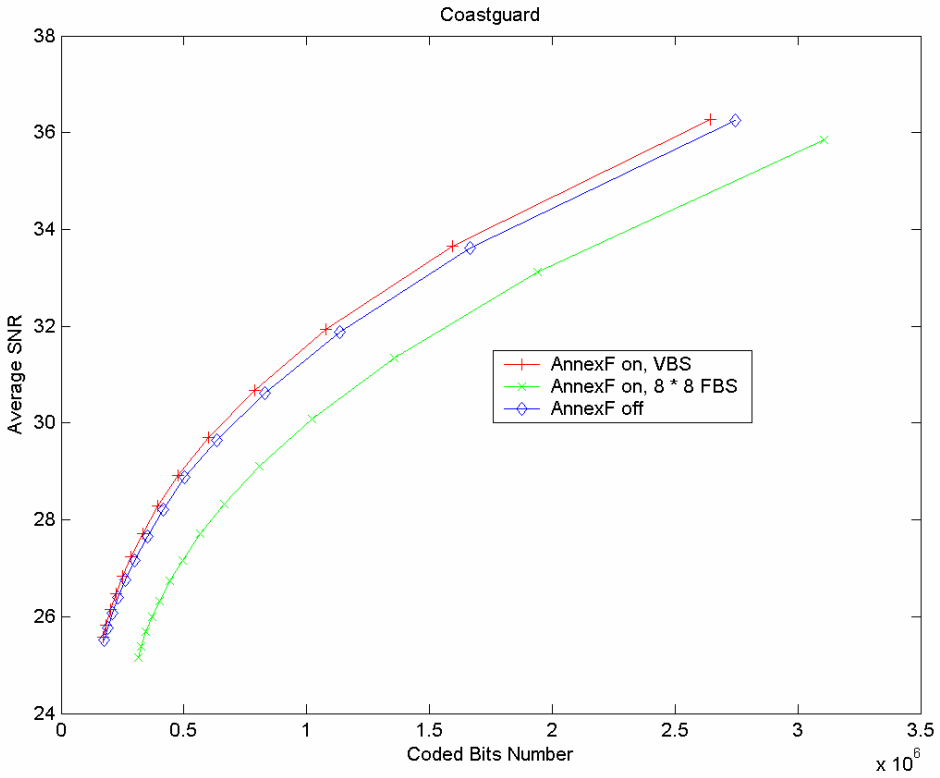

圖 4-15 Coastguard，H.263 FBS 與 VBS

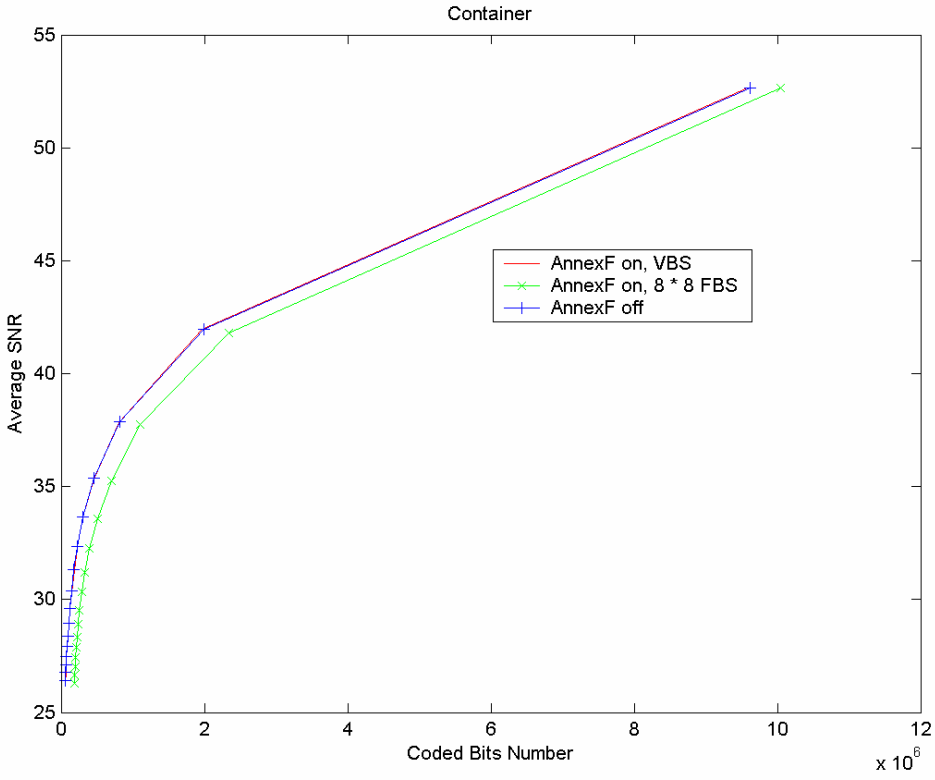

圖 4-16 Container，H.263 FBS 與 VBS

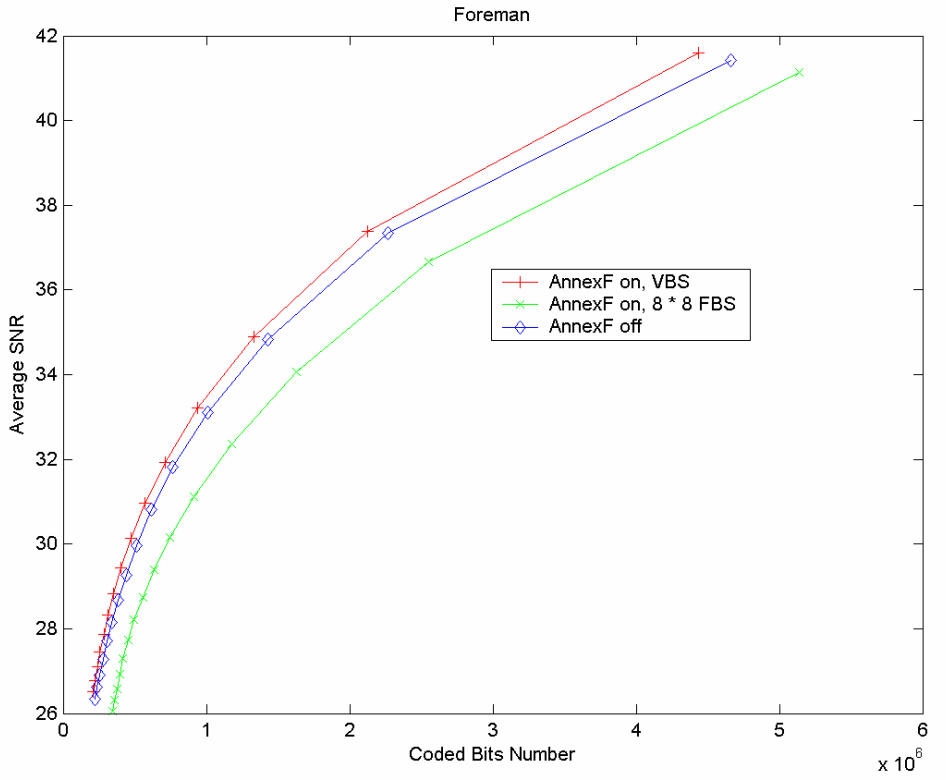

圖 4-17 Foreman，H.263 FBS 與 VBS

模擬結果分析：

4.4.1 節的模擬結果已經告訴我們 8x8 FBS 產生太多動態位元數，四個測試序列的結果中，8x8 FBS 整體壓縮表現最差是可以預期的。模擬結果同時顯示 VBS 優於 16 x 16 FBS，但差別很小。VBS 優於 16 x 16 FBS 原因是預測編碼器可以自由在動態位元數目與像素位元數目之間做選擇。VBS 與 16 x 16 FBS 壓縮後的差別很小，主要因爲 VBS 需要啓動 Annex F，每個畫面標頭因此多出了 35 個位元，有使用 8 x 8 動態預測的方格多出 3 到 13 個位元（見 4.2.1 與 4.2.2 節）。4.4.2 與 4.4.3 兩節的結果都是 VBS 壓縮率優於 FBS，VBS 計算機運算量多於 FBS。

4.4.4 Inter 預測編碼綜合模擬比較

本節以模擬比較 MPEG4 VM5〔28〕，H.263++ Annex F TMN8〔17〕，與 H.264 JM7.1〔18〕的 Inter 預測編碼，轉換編碼與熵編碼的整體表現。模擬所呈現的結果同時顯示了視訊標準的 Inter 預測編碼，轉換編碼與熵編碼優劣。測試序列選用 Akiyo，Coastguard，Container 以及 Foreman，畫面樣本見圖 3-9。每個測試序列取一百張連續的 4：2：0 QCIF 畫面，也就是每張 QCIF 畫面有 176x144 個明度像素，2x88x72 個彩度像素。我們以量化參數調整各視訊壓縮標準在不同的壓縮比下的畫質表現。H.263 量化參數由 10 到 30；H.264 量化參數由 10 到 50；MPEG4 量化參數由 10 到 30。

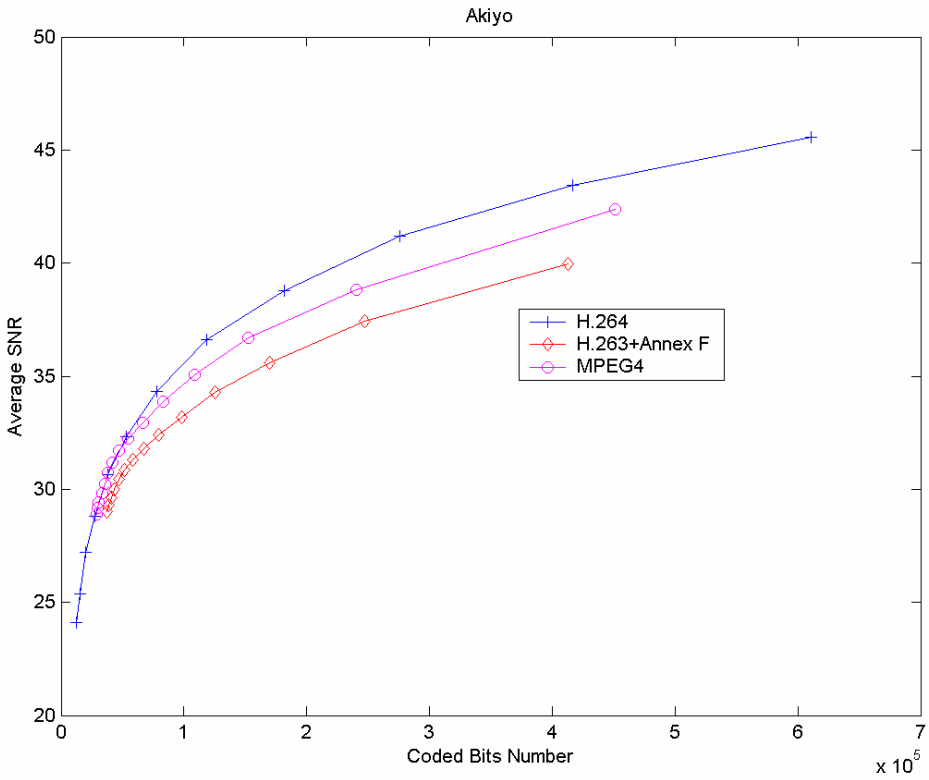

圖 4-18 Akiyo，三種 Inter 預測編碼綜合比較

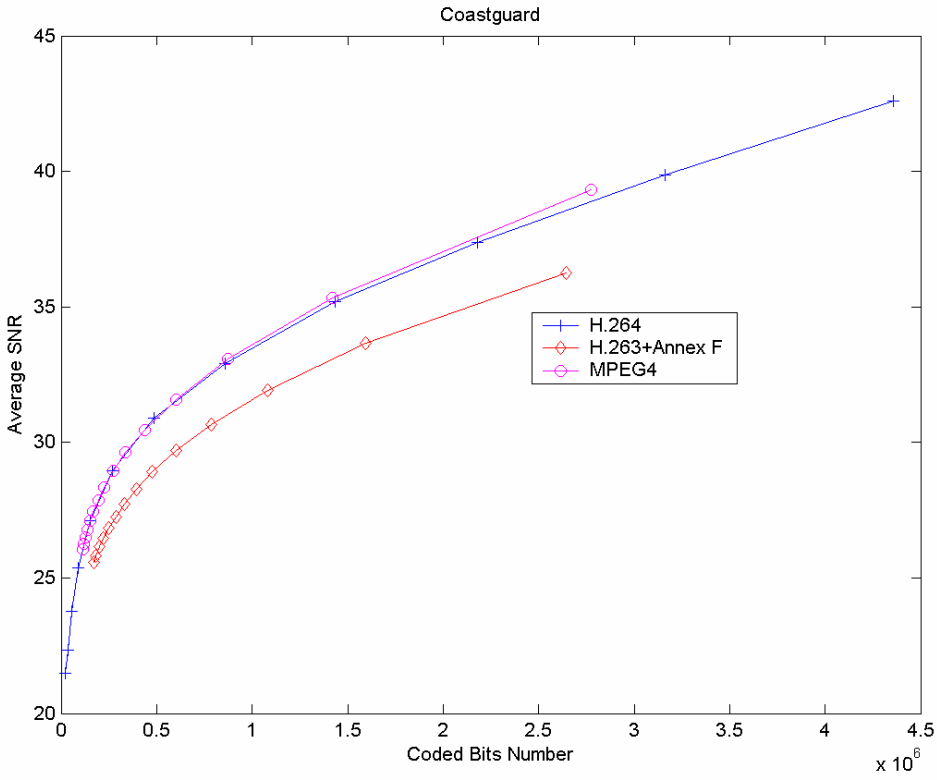

圖 4-19 Coastguard，三種 Inter 預測編碼綜合比較

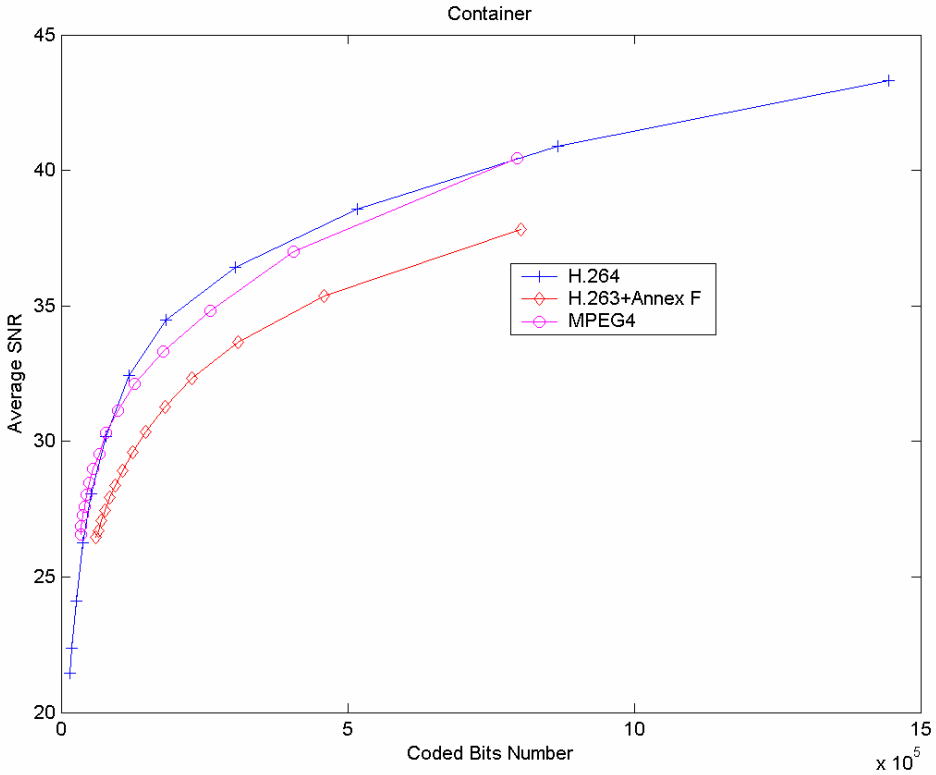

圖 4-20 Container，三種 Inter 預測編碼綜合比較

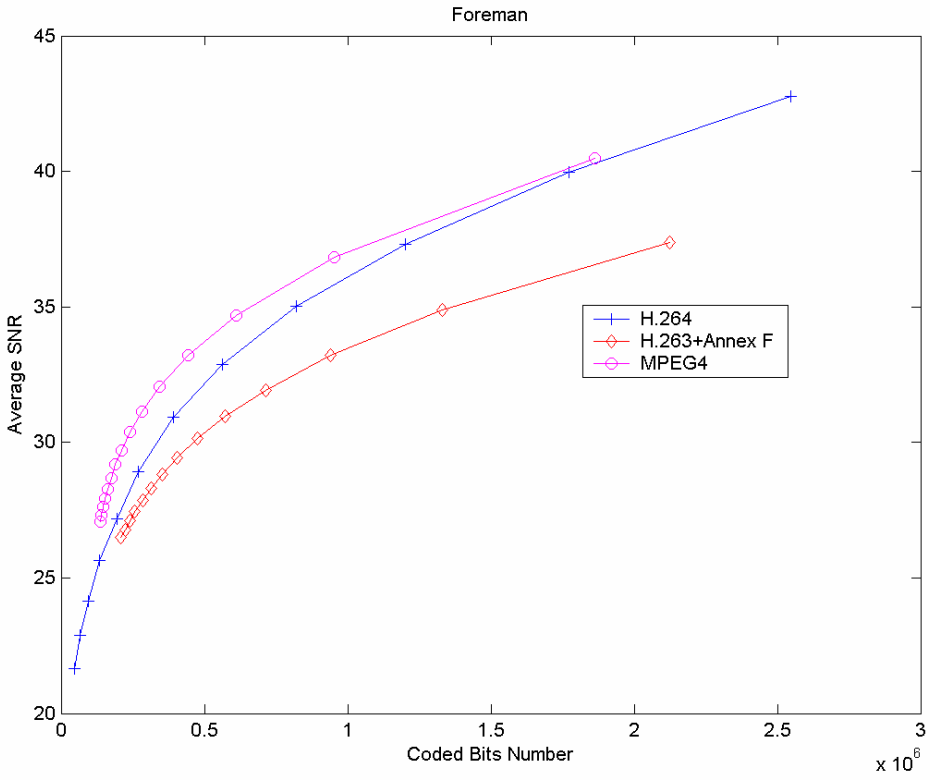

圖 4-21 Foreman，三種 Inter 預測編碼綜合比較

模擬結果分析：

從 Akiyo，Coastguard 與 Container 的模擬結果，H.264 優於 MPEG4；而 Foreman 的模擬結果，MPEG4 卻優於 H.264。四個測試序列的差別如下： Foreman 的表情動作誇張，畫面重疊性不大；而其他三個測試序列都有固定的背景（Akiyo 是攝影棚和單一顏色的服飾）或景象（Coastguard 與 Container 的水面波紋），前景則有固定單純的移動格式（Akiyo 是播報員簡單的面部表情，Coastguard 與 Container 則是船隻以固定速度航行）。也就是說，Akiyo，Container 與 Coastguard 畫面中的鄰近方格間的移動向量極爲近似。近似的移動向量有利於 H.264 這種複雜的移動向量差分編碼；至於畫面重疊性不大的 Foreman，H.264 的發揮空間有限。