複雜度可控制之 HEVC 視訊編碼器

謝昇宏、李俊良、王周珍、董棋偉
義守大學電子工程學系

摘要—本論文提出一個複雜度可控制之 HEVC 視訊編碼器，能在不同的編碼時間要求下完成視訊編碼，以期能應用在消費性視訊產品的即時編碼與傳輸。論文所提方法主要是在時間軸上進行最佳 CTU 樹狀結構的預測，並控制每張畫面的編碼時間來進行複雜度控制，來降低 HEVC 編碼複雜度，達到在預設的時間內完成編碼。我們首先利用前一張已編碼完成的畫面，並儲存該畫面的最佳 CTU 樹狀結構，作為待編畫面之 CTU 樹狀結構，再利用已編碼的視訊時間來預測下張畫面的複雜度(FU 或 FC)，進行 HEVC 編碼複雜度控制。從實驗結果可以發現，在犧牲些許的位元率和視訊品質下，論文所提方法在時間限制內，能成功完成視訊編碼。

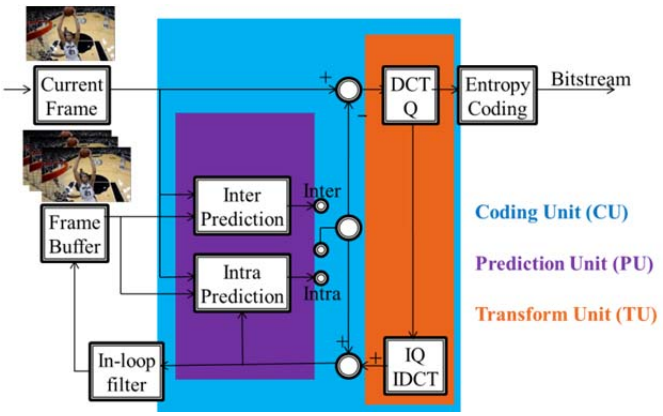
一、 簡介

高效率視訊編碼(high efficiency video coding: HEVC)標準是由 ITU-VCEG 和 ISO MPEG 所組成之視訊編碼共同協作團隊(Joint Collaborative Team on Video Coding，JCT-VC)共同制定的最新國際視訊編碼標準[1]，和先前的視訊標準比較起來，不管是品質或是壓縮率都有相當程度的改善。為了提高編碼效能，達到高壓縮和高品質的目標，HEVC 除了可變區塊大小(variable block size)、畫面間預測(inter prediction)、去方塊(deblocking)濾波器、整數離散餘弦轉換(integer discrete cosine transform: IDCT)等等與 H.264 相同的工具外，HEVC 並將眾多工具歸納出三個基本編碼單位，分別為編碼單位(coding unit: CU)、預測單位(prediction unit: PU)和轉換單位(transform unit: TU)。

圖一為 HEVC 的編碼架構圖 HEVC 基本上與 H.264 相似，採用區塊混和編碼方式，但主要架構上被分出三個基本的單位，對於新的視訊編碼標準 HEVC 而言，其中在 CU 的部分佔了全部編碼端整體運算時間的 95%，其中包含了 PU 的預測時間佔了約 60%，這部分主要運算是運動估測(motion estimation: ME)，而 TU 的部分則佔了約 34%。因此 CU 的部分是 HEVC 編碼運算中最重要且是最花時間的一個模組。為了能達到時間上的控制，我們勢必要降低編碼時間，所以論文著重在 CU 編碼的高運算量，以達到編碼時間控制的目的，本論文提出了一個改良型複雜度控制演算法，利用編碼時間的複雜度判斷，提早去預測整體的編碼時間，利用畫面的調控改變編碼速度，已達成我們的目標編碼時間，並結合最佳 CTU(coding tree unit)樹狀結構，達到加速編碼與提升畫面品質的目的。

[1] 本研究由義守大學校內專題計畫補助，計畫編號 ISU 102-04-03。

[2] 通訊作者: 王周珍 E-mail: chchwang@isu.edu.tw



圖一：HEVC 視訊編碼架構圖

二、 複雜度的分析

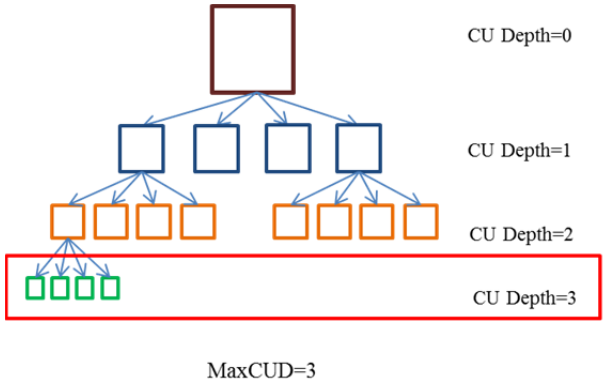
影像編碼的複雜度在 HEVC 之中提升了非常多，在 HEVC 之中的 CU 大小從 8×8 到 64×64，而且在每一層級的 CU 又包含 PU 和 TU 的部分，PU 有多達 7 種畫框間的預測、2 種畫框內的預測，在 PU 模式下，還要在往下計算 3 層的 TU，再加上編碼工具的增加，影像解析度的提高等等，導致 HEVC 的計算複雜度比起 H.264 要來的高上許多，我們利用實際以影像去做分析，從中可以發現到，CU 包含了 PU 及 TU 占了約全部編碼時間的 95%，其餘 5%的部分是一些讀取檔、內嵌式迴路濾波器等等，而在 CU 整體 95%的時間之中，裡面的 PU 預測時間部份佔了約 60%，TU 的轉換時間部分約佔了 34%，其餘 6%的部分是一些其他編碼工具的時間，由此分析可以得知，整個編碼的複雜度幾乎都在 CU 的部分，因此，如果能再決定 CU 的部份來降低它的編碼複雜度，那就有很高的機率降低它整個編碼的複雜度。

三、 複雜度可控制之 HEVC 視訊編碼器

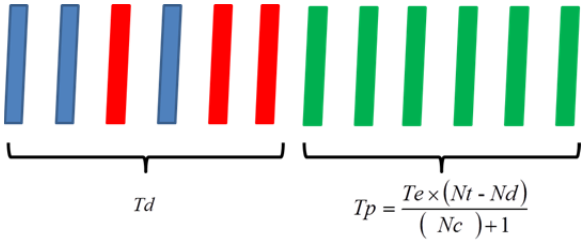
3.1 複雜度控制演算法

Correa 等學者所提出的複雜度控制演算法(complexity control algorithm: CCA)[2]，在 CCA 的方法中，必須定義最大深度的 CU，如圖二所示，圖中表示一個 64×64 的 CTU 經過每一層 CU 的計算之後，最後分割完成所決定的尺寸大小，從中我們定義它往下切到最深的層級為最大深度(maximum CU depth: MaxCUD)，圖中的情況 MaxCUD 為 3，接下來 CCA 將定義一個不受複雜度控制的畫面(unconstrained frames: *FU*)和一個受複雜度控制的畫面(constrained frames: *FC*)，藉由這兩種畫面去調整

Corresponding author:chchwang@isu.edu.tw
DOI:10.6159/IJSE.2014.(4-2).81



圖二：最大深度定義示意圖



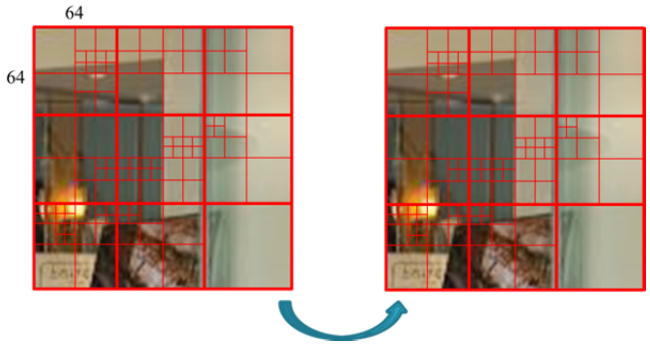
圖三：利用 *Tp* 整體預測時間示意圖

到我們想要的目標編碼時間，而 *FU* 的定義是不做複雜度控制，執行原始的 HEVC，因此編碼的時間最長，但影像品質與位元率為最佳，而 *FC* 的定義為利用前一張畫面所存下來的 MaxCUD，直接計算該值所表示的 CU 深度，因此加快編碼的時間，但由於切乏彈性導致影像品質與位元率效能不佳。

CCA 之 *FU* 與 *FC* 畫面的配置，如圖三所示，藍色表示 *FU*，紅色表示 *FC*，而 *FC* 是利用前一張畫面所存的 MaxCUD 的值，直接計算該值表示的 CU，其中 *Nc* 代表 *FU* 畫面往後面加了 *FC* 的張數，而 *Nc* 要如何去決定出來，就必須依靠一個由 CCA 提出公式與目標編碼時間(target time: *Tt*)去做決定，*Tt* 的定義就是我們想要編碼器在我們設定的時間內，去完成編碼的動作。論文將目標編碼時間設定為實際編碼時間(*Tt*)±10%之間，未編完畫面預測複雜度(*Tp*)定義如下:

$$T_p = \frac{T_e \times (N_t - N_d)}{N_c + 1} \quad (1)$$

其中 *Nt* 代表的是影像畫面總數，*Nd* 是目前為止處理的畫面總數，*Nc* 是 *FC* 的連續張數，*Te* 是最後 *FU* 畫面的編碼複雜度加上 *FC* 畫面裡 *Nc* 數量的畫面編碼複雜度，*Tp* 是代表剩餘未編完的畫面預測複雜度，而 CCA 是利用已經編碼完成的畫面的時間，稱為 *Td*，之後加上預測尚未編碼的畫面預測時間 *Tp*，就可以形成一個預估整體編碼完成的預測時間，如圖三所示。我們利用 *Tt* 進行畫面編碼複雜度判斷，當 $Tp \geq 1.1Tt$ 表示編碼時間不足，所以必須增加 *Nc*，也就是加入 *FC* 降低複雜度，相反的，當 $Tp \leq 0.9Tt$ 則表示編碼時間充足，可以降低 *Nc*，利用 *FU* 進行編碼，以維持影像品質，當 $0.9Tt > Tp > 0.9Tt$ 代表符合編碼時間的要求，所以不需要變動 *Nc*。

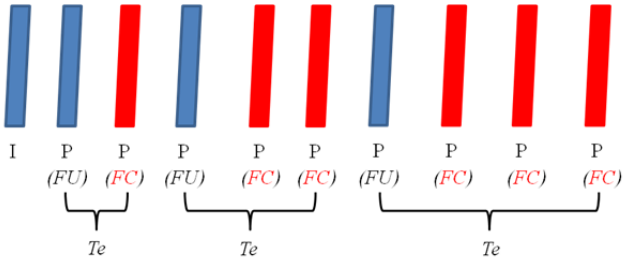


圖四：*FC* 畫面執行 MCCA-BCUT 示意圖

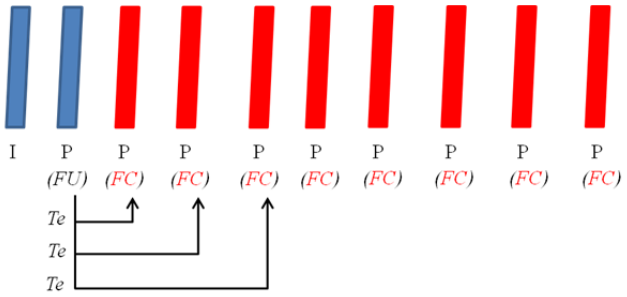
3.2 改良型複雜度控制演算法

為了進一步改善 CCA 演算法，我們提出改良型複雜度控制演算法(modified complexity control algorithm: MCCA)，由於 CCA 中畫面 CTU 樹狀結構利用 MaxCUD 來進行預測，其中 *FC* 畫面直接利用 MaxCUD 的值所代表的深度去執行相對 CTU 樹狀層之 CU、PU 和 TU 的計算，所以只有在固定的一層深度進行預測，這預測方式較為粗略，導致影像品質下降。為了改善影像品質，論文所提 MCCA 利用最佳的 CU 樹狀結構(best CU tree: BCUT)來取代 CCA 方法中的 MaxCUD 預測，簡稱為 MCCA-BCUT。而 BCUT 定義是 CU 最後完成切割的最佳畫面 CTU 樹狀結構，如圖四所示，左圖表示畫面 CU 經過 PU 與 TU 之後，所得到的最佳的 CU，而右圖則是 *FC* 畫面利用 BCUT 取代 MaxCUD 的示意圖。

MCCA 另一改良方法是畫面控制的部分，觀察 CCA 畫面編排的特性，可以發現到 *Nc* 畫面每次只增加或減少一畫面，而且必須執行一畫面組合的時間(*Te*)後才能再增加 *FC*，這種狀況導致須到後半段視訊之 *FC* 增多才能大幅降低編碼時間。因此在面對嚴苛的編碼時間限制下，就會來不及完成編碼，導致視訊壓縮失敗(fail)。為了能得到更彈性的編碼時間控制，MCCA 提出不同的畫面控制方法，能夠在一開始就判斷出更多的 *FC* 來降低編碼的時間。MCCA 畫面編碼時間控制方式與 CCA 不同的地方在於判斷的時機與 *Te* 選擇，首先是在判斷時機，以圖五來舉例說明，CCA 畫面控制方式會在每一組 *Te* 最後的 *FC* 的畫面編碼完成後，再利用公式(1)進行判斷 *Nc* 畫面增加或減少一畫面，而論文所提 MCCA 則會在每一張畫面編碼完成後，就進行公式(1)的判斷，直接決定下一張是 *FU* 畫面，或 *FC* 畫面，因此可以有連續多張 *FC* 畫面的機會。利用 MCCA 的方法可以跳脫 CCA 在畫面組合上的限制，因此 MCCA 可以在視訊編碼初期就能使用更多的 *FC* 畫面和更具彈性，如圖六所示。接下來是 *Te* 的部分，以圖五的情況來說明，CCA 的 *Te* 選擇就是每一個黑色括號括起來的畫面組的編碼時間，以第一個黑色括號來看就是當 *FU* 跟後面一張 *FC* 編完的時間點，就進入公式(1) 進行判斷，其中 CCA 所使用的 *Te* 就是第一個黑色括號括起來的畫面組的編碼時間，而 MCCA 使用的 *Te* 為當每一張畫面編碼完成後往前加到 *FU* 畫面為止的編碼時間，藉由每次畫面編碼完就進行判斷下一張畫面為 *FU* 或是 *FC*，如圖六所示。



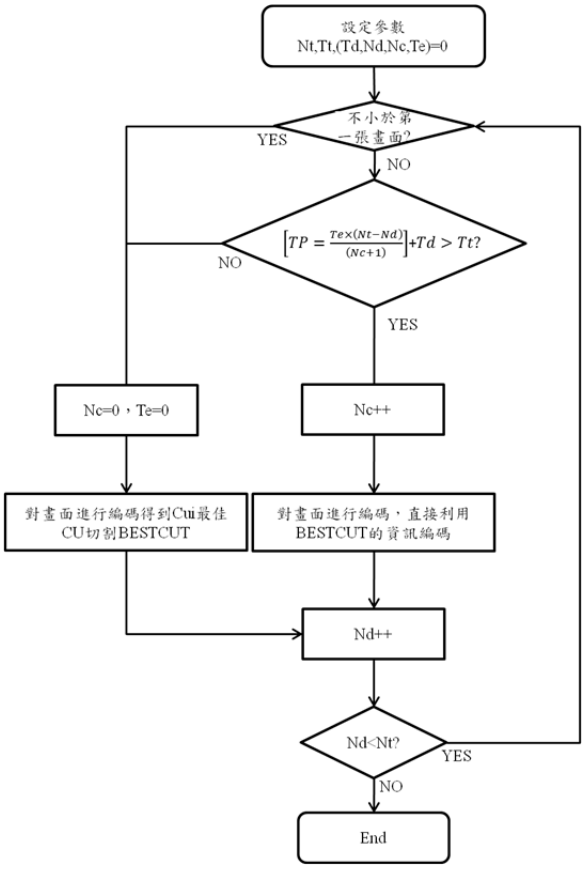
圖五：CCA 畫面編碼時間控制方式



圖六：MCCA 畫面編碼時間控制方式

圖七是論文所提 MCCA 之流程圖，詳細步驟如下:

步驟一、計算出影像全部待編碼的畫面(*Nt*)
步驟二、輸入我們要讓編碼器在多少時間編碼完成(*Tt*)
步驟三、 設參數 *Td*(目前已編碼完成的時間)、*Nd*(目前已處理的畫面總數)、*Nc*(*Fc* 的連續張數)為 0。
步驟四、 判斷是否不小於第一張畫面，是的話進行步驟五，不是就進行步驟 十一。
步驟五、 進行 *FU* 畫面的編碼，並將所得到的 BCUT 給儲存下來。
步驟六、 因為編完一張 *FU*，所以 *Nd* 要加一。
步驟七、 判斷畫面是否編完，如果編完就結束，沒有就進行步驟四。
步驟八、 檢查 *Nc* 的值，*Nc* 值不為 0，就必須編碼 *FC*，並將所得到的 BCUT 給儲存下來。
步驟九、 因為編完一張 *FC*，所以 *Nd* 要加一。
步驟十、 判斷畫面是否編完，如果編完就結束，沒有就進行步驟四。
步驟十一、計算目前 *FC* 往前到 *FU* 為止的時間，得到 *Te*。
步驟十二、計算目前為止編碼完成的時間，得到 *Td*。
步驟十三、利用式子(1)去計算預測的編碼時間，得到 *Tp*。
步驟十四、進行判斷 if $(Td+Tp>Tt)$ 如果是進行步驟十五，不是則進行步驟十六。
步驟十五、*Nc* 加一張，回到步驟八執行 *FC* 畫面編碼。
步驟十六、*Nc* 歸 0，*Te* 歸 0 回到步驟五執行 *FC* 畫面編碼。



圖七：MCCA 流程圖

四、 實驗結果與討論

本論文採用 HEVC 的編碼參考軟體為 HM8.1，以 Visual++. 2008 作為編譯環境，完成 PC 上的軟體模擬測試後對數據進行分析，主要的實驗條件如下：

1. 影像序列：BasketballDrill(832×480)和 vidyo3(1280×720)，如圖八所示。
2. 編碼規範：Low Delay (LD) profile
3. 畫面張數：100
4. 量化參數：32
5. 參考畫面：1
6. 畫面率：25、30

為了更進一步評估改善後控制的效能，我們定義時間改善率(time improving ratio: TIR)如下：

$$TIR=\left(T_{HMCT}-T_{EHMCT}\right)/T_{HMCT}\times 100\% \qquad (2)$$

其中 T_{HMCT} 代表 HM 在編碼的運算時間，T_{EHMT} 代表其他方法畫面編碼的運算時間。論文將就各種不同的視訊採用原始的 HEVC、CCA 和 MCCA 三種方法在 HM8.1 版本測試平台的模擬結果，進行分析討論。

表 I 為 HM8.1 與其他所提方法並且沒有使用 *Tt* 進行限制的編碼時間比較表，從表中可以發現 CCA 和 MCCA 的編碼時間均能大幅降低，CCA 和 MCCA 比 HM8.1 的 TIR 平均分別改善 61.1%和 74.85%，兩者均能有效的改善 HM8.1 的編碼時間。

(a) (b)

圖八：HEVC 標準影像序列

表 I
各種方法編碼時間改善比較表

Sequence	Methods	Encoding time(sec)	*TIR*(%)
BasketballDrill	HM8.1	8889	0
	CCA	3743	57.8
	MCCA	2584	70.9
Vidyo3	HM8.1	11024	0
	CCA	3525	68
	MCCA	2548	76.8

表 II
BasketballDrill 在時間限制在 3800 秒，各方法之間的比較

	HM8.1	CCA	MCCA
實際編碼畫面數	100	100	100
實際編碼時間(sec)	8889	3743	3776
Bitrate(kbps)	959.46	1060.25	1048.34
PSNRY(dB)	34.22	34.06	34.08

表 III
BasketballDrill 在時間限制在 3200 秒，各方法的複雜度控制比較

	HM8.1	CCA	MCCA
實際編碼畫面數	100	76	100
實際編碼時間(sec)	8889	fail	3174
Bitrate(kbps)	959.46	fail	1071.06
PSNRY(dB)	34.22	fail	34.05

表 II 和表 III 是測試影像序列為 BasketballDrill 在編碼限制時間分別為 3800 秒和 3200 秒之模擬結果，從表 II 中可以發現，當編碼時間限制為 3800 秒時，在犧牲些許的位元率和視訊品質下，CCA 和 MCCA 兩種演算法均能在如期的時間限制內，成功的完成視訊編碼，另從表 II 中也可以發現，在相當接近編碼的時間下， MCCA 比 CCA 完成更好的編碼效能(Bitrate 和 PSNRY)。當編碼時間限制降低至 3200 秒時，從表 III 可以發現，CCA 只能編到第 76 張畫面，表示它無法在限制的時間內編完 100 張的畫面，造成視訊編碼失敗，因此不需要比較 Bitrate 跟 PSNRY。綜合表 II 和表 III，可以得知 MCCA 在同樣的編碼時間下，具有最好的影像品質與壓縮率，而且時間上的控制更有效率。

表 IV
Vidyo3 在時間限制在 3600 秒，各方法之間的比較

	HM8.1	CCA	MCCA
實際編碼畫面數	100	100	100
實際編碼時間(sec)	11024	3525	3595
Bitrate(kbps)	585.28	636.27	624.54
PSNRY(dB)	38.09	37.90	37.96

表 V
Vidyo3 在時間限制在 3300 秒，各方法之間的比較

	HM8.1	CCA	MCCA
實際編碼畫面數	100	90	100
實際編碼時間(sec)	11024	fail	3227
Bitrate(kbps)	585.28	fail	638.15
PSNRY(dB)	38.09	fail	37.92

表 IV 和表 V 則是另一測試影像序列 Vidyo3 在編碼限制時間分別為 3600 秒和 3300 秒之模擬結果，論文所提 MCCA 的編碼效能一樣優於 CCA，當編碼時間限制降低時，MCCA 依然能成功完成視訊編碼。

結論

本論文藉由自然視訊在時間軸的高相關性，利用前一張已編完的 CU 樹狀結構來加快 HEVC 的編碼時間，並且利用目標時間去調控編碼時間，以達成在不同時間限制下均能完成視訊編碼，並且比最近之 CCA 獲得更佳的編碼複雜度控制，來達到即時視訊的應用。

參考文獻

[1] G. J. Sullivan, J. R. Ohm, W. J. Han and T. Wiegand,“Overview of the high-efficiency video coding standard,” *IEEE trans. on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649-1667, Dec. 2012.

[2] G. Corrêa, P. Assuncao, L. Agostini, and S. Cruz, “Complexity Control of High Efficiency Video Encoders for Power-Constrained Devices,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, Nov. 2011.

[3] B. Bross, W. J. Han, G. J. Sullivan, J. R. Ohm, and T. Wiegand, “High efficiency video coding (HEVC) text specification draft 8,” ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-J1003, July 2012.

[4] C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng, and L. G. Chen, "Power-aware multimedia: concepts and design perspectives," *IEEE circuits and systems magazine*, vol. 7, pp. 26-34, 2007.

[5] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, “Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC)”, *IEEE trans. on circuits and systems for video technology*, vol. 22, no. 12, pp. 1668-1684, Dec. 2012.

[6] 葉則汎編著，“應用於H.264之高效率次像素運動估測演算法”，義守大學電子工程學系碩士論文，中華民國九十七年七月。