一個快速轉換JPEG序列影像為H.264串流視訊之方法

李相賢
元智大學資訊工程研究所
Email: ysha3843@hotmail.com

林啟芳
元智大學資訊工程研究所
Email: cscflin@saturn.yzu.edu.tw

摘要

我們提出開放式架構的方法，將 JPEG 序列影像轉換為 H.264 的視訊串流。所提方法的優點包括複雜度低容易實作、轉換速度極快、且轉換後的影像品質失真不大，維持一定的水準。但將開放式架構應用在轉換序列 JPEG 影像為 H.264 串流視訊時，會遭遇到一些問題，包括畫面模式的選擇、宏塊(macro-block)模式與區塊模式的選擇、整數轉換的問題、色彩模式的轉換、量化選擇、及預測值問題。每一項問題都必須妥善處理，否則無法得到高品質的轉換結果。我們提出直接由 DCT 區塊的係數來計算預測值的方法，可以大幅加快處理速度與提高解碼後的影像品質。除此之外，我們也討論可能導致轉換失真的原因，包括：(1)預測值的不準確；(2)DCT 區塊與整數轉換區塊的差異；(3)編碼整數連續性；及(4)錯誤累積。最後，實驗結果將展示所提方法的優越性。

關鍵詞：靜態影像、視訊串流、格式轉換、開放式架構。

1、序論

隨著網路基礎建設的普及與數位傳輸技術的進步，帶動以提供視訊(video)為主的服務網站的數量快速成長，例如 Youtube (www.youtube.com)網站，可以提供給訪客上傳或下載視訊影片，滿足人們對瀏覽視訊的娛樂需求，因此這類網站通常能在短時間內就吸引大量瀏覽人潮。但由於影片的龐大視訊串流造成資料儲存、視訊傳送、網路頻寬等諸多問題出現，這些問題都迫切需要得到解決。另一方面，以串流格式為主的網路數位監控系統，逐漸成為安全監控產業的主流，取代傳統以 JPEG 影像壓縮模式為主的安全監控系統。後者雖然可以提供高品質的儲存影像，但在網路有限頻寬的條件限制下，無法順暢瀏覽視訊而逐漸被 MPEG-4 及 H.264 [1,2]等壓縮串流格式所取代。另外一項因素是 JPEG 影像壓縮模式無法在一般播放器（例如 Realplayer 或 MediaPlayer）上播放，也無法以 3G 手機來瀏覽，因此在未來發展上受到很大的限制。

相較於 MPEG-4 或 H.264，JPEG 雖然有上述缺點與限制的存在，但仍具有下列競爭優勢，說明如下：(1) JPEG 的相關技術發展已有一段時間，所以成熟度與穩定性高；(2) JPEG 影像品質極佳，對後續發展智慧視訊辨識系統較具優勢；及(3) JPEG 晶片製作成本低且體積相對較小，在某些產業上仍具有不被取代的地位，例如 PC Camera (or Web Camera)，目前仍舊採用 JPEG 壓縮晶片。在另一方面，網路攝影機(IP Camera)的發展現況是以提供多種壓縮格式（多模）為主，例如雙模（JPEG 與 MPEG-4）或三模（JPEG、MPEG-4、與 H.264）。設計多模的目的是內部網路儲存選擇影像品質較佳的 JPEG 格式，而外部網路傳輸則採用瀏覽較順暢的 MPEG-4 或 H.264 格式。發展多模與多解析度的網路攝影機，是未來監控產業的發展趨勢。

基本上，JPEG 靜態影像也能以影片方式播出，作法是利用人類視覺暫留作用，以每秒 30 幅的播放速度將 JPEG 序列影像連續播出。這種基於靜態影像壓縮技術演變而成的動態播映，稱為 Motion JPEG 或簡稱為 MJPEG。MJPEG 承襲了 JPEG 的優點，清晰度高，對於影片任何時段的單一畫面擷取，也能得到極為理想的畫面品質。由於 MJPEG 屬於開放自訂格式，無版權與權利金等問題，因此吸引許多監控廠商開發

與使用。但因為是自訂格式導致有相容性的問題，廠商在沒有正式標準的規範下，各自開發的 MJPEG 彼此規格上會有不同差異，同時 MJPEG 也不支援影音同步。

為了同時具有 JPEG（靜態影像格式）與 H.264（視訊串流格式）的優勢，本論文提出格式轉換(transcode)的方法，在維持影像品質的前提下，將 JPEG 序列影像轉換為 H.264 串流視訊。所發展技術可應用在傳統監控設備或網路攝影機，達到高品質影像的儲存需求與瀏覽順暢的傳輸要求，並可在一般播放器或 3G 手機上瀏覽。

基本上，JPEG 壓縮格式是設計用來壓縮單張靜態影像，而 H.264 為多媒體串流的壓縮格式，兩者之間的轉換最直接的作法是將 JPEG 序列影像解壓縮後再以 H.264 方式重新編碼。但如此一來會浪費大量的轉換時間，且重複編碼會導致影像品質流失。為維持高品質影像的要求，我們排除重複壓縮的作法。目前較少論文有針對兩者格式提出轉換的方法，但仍有許多相關的文獻研究提供參考。下列文獻瀏覽可分為 H.264 區塊轉換及格式轉換的相關研究，內容敘述如下。

在 H.264 的區塊轉換相關研究中，H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky [3]對於 H.264 底層轉換與量化方式做詳細介紹。他們提出轉換公式將傳統 8×8 區塊(block) 的離散餘旋轉換 (Discrete Cosine Transform; DCT) 推導到 H.264 4×4 區塊的 DCT，並且使 4×4 區塊在運算上能避免乘除運算，僅需做位移操作 (shift operation) 和加法運算即能完成轉換功能。V. Patil and R. Kumar [4] 提出能在 DCT 空間對 H.264 做運算，並改變 H.264 串流視訊解析度大小的方法。J. Xin, A. Vetro, and H. Sun [5]推導由 8×8 DCT 區塊轉換成 4×4 H.264 區塊的轉換公式。Z. He and M. Bystrom [6]提出 DCT 區塊轉換到 H.264 整數轉換區塊的方法，並經由複雜度分析證明能使用較少的加法與乘法數量得到結果。

在格式轉換的研究中，T. H. Tsai, Y. F. Lin, and H. Y. Lin [7]將視訊格式轉換分成三種類別：空間的視訊格式轉換(spatial transcoding)、暫存的視訊格式轉換 (temporal transcoding)、及特殊應用的視訊轉換 (special application transcoding)，並且強調在轉換中會特別重視在播放時的解析度轉換方式。A. Vetro, C. Christopoulos, and H. Sun [8]整理了格式轉換的相關議題，包含轉換架構、位元率 (bit rate)縮減、動態向量(Motion Vector; MV)對應、及宏塊(macroblock)選擇。Z. He and M. Bystrom [9]利用不同的拆解運算方式，將 MPEG-2 中的 DCT 區塊修正為 H.264 的整數轉換區塊。近年來，格式轉換研究則常見到 MPEG-2 與 H.264 之間互換議題 [10]，H. Kalva and B. Petljanski [11]提出在 MPEG-2 轉換到 H.264 畫面內轉碼(intra transcoding)時，藉由抽取方向特徵(directional features)來決定 H.264 宏塊模式，節省轉換上的模式判斷。H. Kalva, B. Petljanski, and B. Furht [12]提出一個低複雜度的轉換方式，利用 MPEG-2 提供的 DCT 係數，做為轉碼為 H.264 預測區塊模式與預測方向依據。在轉換中，如何延續原規格的動態補償資訊也是重點。Z. Zhou, S. Sun, S. Lei, and M. Sun [13]對於 MPEG-2 轉到 H.264 的動態資訊再利用做討論。S. E. Kim, J. K. Han, and J. G. Kim [14]則對於 MPEG-4 轉到 H.264 轉碼提出一個有效的動態補償演算法。T. Qian, J. Sun, D. Li, X. Yang, and J. Wang [15]利用失真錯誤補償(Drift-Error Compensation)讓 MPEG-2 轉至 H.264 轉換上的失真能夠降低。

到目前為止，大部分相關論文研究重點都放在串流格式如 MPEG-2、MPEG-4、H.264 等格式之間的相互轉換。至於如何將序列靜態影像 JPEG 轉換為串流格式則較少被討論。雖然有文獻提供部分技術可供參考，例如 JPEG 的 DCT 與 H.264 的整數 DCT 的轉換方法，但完整轉換 JPEG 與 H.264 的方法則較少被提出。考量在未來應用上，有愈來愈多支援 H.264 格式的產品出現，舉凡 3G 手機及個人掌上型電腦等，為了使 JPEG 靜態影像也能在這些產品上流暢播出，本論文所提方法，將 JPEG 影像格式轉變為 H.264 的標準串流格式，除能保留原來 JPEG 的高品質影像外，並且具備低複雜度與快速轉換的特性。

本論文的內容編排如下。第一節為序論，說明研究動機、相關文獻瀏覽、問題描述及簡述解決辦法。第二節介紹三種常用轉換格式的

架構。第三節說明所提方法的詳細內容。第四節為實驗結果展示與分析。最後，第五節說明結論與未來研究方向。

2、轉換格式的架構

有關影像壓縮格式相互轉換的架構，可分為下列三種：串聯式架構、開放式架構、及封閉式架構。不同的架構在影像品質與處理複雜度各有其優缺點。為了敘述上的方便，轉換前的影像壓縮格式稱為原始格式，轉換後的壓縮格式稱為目標格式。

第一種架構為串聯式架構，採用直覺的作法，對原始格式做完整的解碼動作，再經由重新編碼轉變成目標格式，處理流程如圖 1 所示，左邊虛線區塊代表原始格式的解碼流程，右邊虛線區塊代表目標格式的編碼流程。這種架構最大的好處是藉由重新完整的編碼行為，可以調整動態預測模式，得到最佳的影像補償效果。但這種架構需要原始格式的解碼流程與目標格式的編碼流程，轉換時間較長且耗費大量的計算時間，並不合適應用在需要即時轉換處理的情況。

第二種架構為開放式架構，作法如圖 2 所示，分成四個區塊。原始格式經過可變長度解碼(Variable Length Decoding)與標準反量化後，再執行目標格式的標準量化與可變長度編碼(Variable Length Coding)程序。開放式架構不需要做任何額外的離散餘旋轉換(DCT)與反離散餘旋轉換(Inverse DCT; IDCT)，在計算量上能得到極大的節省。然而對於目標格式而言，由於開放式架構不重新做動態預測處理，因此動態補償模式無法和原始格式得到同樣的效果。舉例來說，I 畫格(Intra-coded picture)在量化過程中喪失了一小部分高頻資訊，然而緊接著 P 畫格(Predictive-coded picture)是藉由參考 I 畫格來做補償，一旦維持原始格式的參考方式，I 畫格的失真會連帶影響到 P 畫格，導致 P 畫格得到差異更大的數值，這種失真稱為參考上的失真(drift) [10]。參考上的失真最根本的原因是來自於動態參考模式的選擇，因此如何避免動態參考模式產生的失真成為開放式架構必須處理的問題。

最後一種架構為封閉式架構，介於串聯式和開放式兩種架構之間，如圖 3 所示。封閉式架構改進的重點在於動態補償直接對 DCT 像素空間做調整，原始格式解碼至 DCT 區塊，即開始做目標格式的完整編碼。由於動態補償的資訊是來自於 DCT 區塊，因此封閉式架構的目標格式編碼流程必須作修正。和串聯式架構相比，封閉式架構只需執行一次離散餘旋轉換(DCT)及一次反離散餘旋轉換(IDCT)，而串聯式架構則需要執行一次離散餘旋轉換及二次反離散餘旋轉換，因此在轉換時間上能有所提昇。然而在 DCT 空間做補償還是會喪失一定程度的高頻資訊，因此在畫質上會稍遜於串聯式架構。而與開放式架構相比，封閉式架構需要重新執行動態補償動作，能有效解決參考上失真的問題。但由於需要額外的動態補償處理，因此在速度上會慢於開放式結構，且複雜度也相對提高。

由於開放式架構比其它兩種架構更具快速轉換的優勢，因此在本論文中我們採開放式架構的作法。但將開放式架構應用在轉換序列 JPEG 影像為 H.264 串流視訊時，會遭遇到一些問題，包括畫面模式(picture type)的選擇、宏塊(macroblock)模式與區塊(block)模式的選擇、整數轉換的問題、色彩模式的轉換、量化選擇、及預測值問題。我們提出解決上述問題的轉換方法，並討論可能導致轉換失真的原因。實驗結果將驗證所提方法可以達成本論文所設定的目標。

3、方法介紹

本節內容安排如下。3.1 節說明格式轉換必須解決的幾個基本問題。3.2 節提出在開放式架構中預測值的運算公式。最後，在 3.3 節討論格式轉換的失真問題與產生原因。

3.1 格式轉換的基本問題

本論文的主要目的是將 JPEG 靜態影像格式轉換為 H.264 動態串流格式。所提方法的優點是轉換快速，及轉換後的影像品質能維持 JPEG 高品質的特性。要達到此目的，我們採開放式架構的作法。首先輸入影像為 JPEG 格式之靜態影像，經由霍夫曼解碼(Huffman decoding)及反量化後得出數個 8×8 DCT 區塊。下一步驟，對這些 DCT 區塊進行 H.264 的量化與編碼處理，

最後輸出 H.264 的串流視訊。然而在執行 H.264 的量化處理時會遭遇到一些問題，這些問題包括：(1)畫格模式(picture type)的選擇；(2)區塊(block)模式的選擇；(3)整數轉換的問題；(4)色彩模式的轉換；(5)量化選擇；及(6)預測值問題。這些問題的產生原因來自於兩種編碼格式的使用對象在本質上有很大的差異，一個是提供給靜態單張影像的壓縮標準，而另一則是動態視訊串流的標準。但每一項問題都必須妥善處理，否則無法得到高品質的轉換結果。細節於下各節中說明。

3.1.1 畫格模式選擇

如前所述，我們採用開放式架構來執行格式轉換的工作，但開放式架構最大問題在於有參考失真(drift)的問題 [10]。為了達到快速轉換並維持高影像品質，本論文在格式轉換的過程中，將 H.264 的畫格均設定在 I 畫格（這是因為對 I 畫格而言，不管採用何種轉換架構均能得到很好的效果）。依照 H.264 規格書所定義，滿足所有畫格均為 I 畫格的配置文件(profile)共有四種，分別為 High 10 Intra、High 4:2:2 Intra、High 4:4:4 Intra、及 CAVLC 4:4:4 Intra。選擇上述四種配置文件，除了可以解決因轉碼時畫格間動態預測(inter prediction)必須耗費巨大的時間消耗外，同時也能避免上述參考失真的問題產生，並讓轉碼過程簡單化。因此我們使用上述四種規格做為 H.264 的畫格模式選擇。

3.1.2 區塊模式選擇

在格式轉換中，必須決定採用 H.264 的宏塊(macroblock)大小及預估模式。以 H.264 的畫格內預測(intra prediction)模式而言，在編碼流程中，需要決定的宏塊大小，可選擇 16×16 (I16MB)、8×8 (I8MB)、4×4 (I4MB)、及不預測宏塊(IPCM)四種。決定預測宏塊大小後，計算預測宏塊所提供的每個模式，選擇一個差異值最小的模式做為該宏塊的預測模式。

在 H.264 的眾多編碼程式中，JM 是 H.264 官方認可的開放原始碼之一，也是最常被使用在學術研究與公開討論的用途上，所以又稱為校驗模型(calibration model)。JM 的特性是支援度高但實用性差(編碼速度緩慢)。以 JM 14.2 的版本而言，使用單一預測宏塊可以加快編碼速度，而使用混合預測宏塊(例如混合使用 I16MB 和 I8MB 或混合使用 I16MB 和 I4MB）則可得到較佳的壓縮率。

選擇好的宏塊模式能得到壓縮效果較佳的 H.264 檔案，然而選擇模式的過程卻會需要額外的計算時間。本論文所擬定的目標重點，在於快速封包轉換及轉換後能維持 JPEG 的高影像品質。若在轉換封包時，使用大小可調整的宏塊(macroblock)與區塊(block)，雖能得到較小的輸出檔案(壓縮效果好) ，但處理時間卻會增加。我們考慮在 JPEG 中所解得 DCT 區塊的大小為 8×8 ，而一個宏塊的大小為 16×16 ，使用 I8MB 宏塊剛好可以切成四個 8×8 區塊，在編碼時無需再做額外的變更運算，因此在轉換一致性和效能取捨的考量下，我們選擇每個畫格內預測區塊均設定成 I8MB 宏塊。

在決定使用 I8MB 做為預測宏塊大小後，我們觀察在九種 8×8 區塊的預測模式中，其中 DC 模式在計算預測值時，必須計算上方相鄰 8×8 區塊的最下一列(the Bottom Most Row; BMR)的平均，和左方相鄰 8×8 區塊最右一行(the Right Most Column; RMC)的平均。相較其餘八種預測模式，DC 模式比較容易快速取得預測值。由於我們不希望在轉碼過程中，花費太多時間去做九種參考模式的分析，因此我們選擇 DC 模式作為區塊的預測模式。

3.1.3 整數轉換問題

H.264 轉換到頻率域可分為 4×4 整數轉換與 8×8 整數轉換兩種。早期規格書中並沒有列出 8×8 區塊的轉換方式，因此許多研究，例如 J. Xin, A. Vetro, and H. Sun [5]，提出將帶有小數的 DCT 區塊轉為 4×4 IDCT(整數離散餘旋轉換)區塊的方法。Z. He and M. Bystrom [6]對上述轉換方式提出進一步改進效能的做法。在使用開放式架構的轉換中，JPEG 原本使用的轉換區塊即為 8×8 大小，因此直接沿用 DCT 區塊是一個最簡便與省時的做法。

3.1.4 色彩模式的轉換

JPEG 所使用的色彩模式一般為 YC_bC_r，而 H.264 所使用的色彩模式則稱為 YUV。針對 JPEG 轉換到 H.264 色彩模式在轉換中是否需要作調整，說明如下。YUV 格式是在 80 年代由新

力(SONY)公司所發展出來的視訊格式，將訊號資訊分成三個頻道，分別是 Y（亮度）、U（藍色色差）、及 V（紅色色差）。發展成熟之後，YUV 成為傳統類比電視的主要規格，但是不同國家對於 YUV 的計算方式還是存在不同之處，大致上分為歐規(PAL)、美規(NTSC)、及法規。在電視訊號朝數位化轉變時，開始制定 YC_bC_r 的標準。轉換到數位化的取樣過程有個重要因素需要被考慮，在於人眼對明暗變化敏感程度高於顏色變化，感官上的敏感曲線大致呈現對數分佈，因此對 YC_bC_r 規格做相對應調整。因此可以說，YC_bC_r 是 YUV 色度的一種經由不同縮放及抵補的版本。在電腦中做封包轉換，對於 H.264 與解到 DCT 係數的 JPEG 而言，兩者的色域空間同屬數位化 YUV 規格。對數位化資料而言，YUV（或 YC_bC_r）對於紅(R)、綠(G)、藍(B)三個顏色分量有著相同比重的轉換，轉換公式會遵守 BT.601 或 BT.709 等協定，因此省略色彩轉換模式的選擇並不會影響整體轉換的正確性。

3.1.5 量化選擇

JPEG 使用的量化表隱藏在標頭檔(header file)中，允許使用者採用自行定義的區塊量化方式。而在 H.264 中，量化表是搭配整數轉換使用，分為 0 到 51 個量化依據(QP)，QP 設定愈高則壓縮率愈高，但得到的壓縮品質愈差。在本質上整數轉換與量化運算是相互配合來完成彼此工作。JPEG 解碼所得到的 DCT 區塊，當 QP = 4 時，正好相等於 H.264 的區塊，而在 H.264 的量化意義即代表不做量化。對於 H.264 的其他量化值而言，DCT 區塊雖然可藉由數學運算式得到對應的 H.264 區塊值，然而需要做額外運算且造成品質的耗損。因此，我們選擇將 H.264 的量化 QP 值設定為 4。

3.1.6 預測值問題

最後還有一個問題必須加以解決，否則會對整體轉換的效果產生很大的影響。這個問題產生的原因是因為 JPEG 對 DCT 區塊內的每一個成員，會在像素空間統一減掉預測值(predictor)128 後，再做離散餘旋轉換。但對 H.264 而言，預測值的產生是由各種不同的預測模式計算而得，並非如 JPEG 般為一固定常數。忽略兩者的差異，解碼器(decoder)還是會依照規格書所規定的解碼方式來執行解碼與播放的動作，但此時所解得影像品質會非常糟糕。

由於在開放式架構下，我們沒有將 JPEG 的 DCT 區塊解到像素(pixel)區塊，因此無法估算出預測值的大小。如前所述，預測值在轉換過程中扮演極為重要的角色，所以有必要取得像素區塊的資訊，但如此一來勢必耗費許多時間在 JPEG 影像的解壓縮。因此在下節中我們提出直接由 DCT 區塊取得預測值的做法，藉由所提方法來解決此問題，並大幅改善轉換效果。

3.2 預測值的取得

如前所述，我們將預測區塊模式全部設為 DC 模式，因此需要計算上方區塊最下一列(BMR)之像素值總和與左方區塊最右一行(RMC)之像素值總和作為預測值。然而在開放式架構下，我們並沒有解出原始格式之像素(pixel)區塊，因此無法直接取得上述 BMR 與 RMC 的像素值總和。底下我們提出直接由 DCT 區塊的係數來計算預測值的方法。為方便說明，我們以 4×4 DCT 與 IDCT 轉換為例，但計算式可以擴展應用到 8×8 的 DCT 區塊。假設 4×4 像素區塊表示為 $\{x_{ij}, i, j, =1,2,3,4\}$，其對應 DCT 矩陣為 $\{X_{ij}, i, j, =1,2,3,4\}$，DCT 轉換公式定義如下：

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

其中 $a = 1/2$，$b = \sqrt{1/2}\cos\left(\frac{\pi}{8}\right)$，$c = \sqrt{1/2}\cos\left(\frac{3\pi}{8}\right)$。

IDCT 轉換公式定義如下：

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \tag{2}$$
$$= \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{bmatrix} \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

其中 a, b, c 的定義與式(1)相同。所謂上方區塊 BMR 之像素值總和以式子表示為

$$S_{\text{BMR}} = \sum_{i=1}^{4} x_{4i}, \tag{3}$$

而左方區塊 RMC 之像素值總和表示為

$$S_{\text{RMC}} = \sum_{j=1}^{4} x_{j4}. \tag{4}$$

由式(2)的化簡可以求出下列關係式：

$$\begin{aligned} S_{\text{BMR}} &= 4a^2X_{11} - 4abX_{21} + 4a^2X_{31} - 4acX_{41}, \text{and} \\ S_{\text{RMC}} &= 4a^2X_{11} - 4abX_{12} + 4a^2X_{13} - 4acX_{14}. \end{aligned} \tag{5}$$

利用同樣的推導方式，我們可以求出 8×8 區塊簡化後的關係式如下：

$$\begin{aligned} S_{\text{BMR}} &= 8m_0m_0X_{11} + 8m_0m_1X_{21} + 8m_0m_2X_{31} + \\ &\quad 8m_0m_3X_{41} + 8m_0m_4X_{51} + 8m_0m_5X_{61} + \\ &\quad 8m_0m_6X_{71} + 8m_0m_7X_{81}, \text{and} \\ S_{\text{RMC}} &= 8m_0m_0X_{11} + 8m_0m_1X_{12} + 8m_0m_2X_{13} + \\ &\quad 8m_0m_3X_{14} + 8m_0m_4X_{15} + 8m_0m_5X_{16} + \\ &\quad 8m_0m_6X_{17} + 8m_0m_7X_{18}. \end{aligned} \tag{6}$$

其中

$$\begin{aligned} &m_0 = 0.3536, m_1 = -0.4904, m_2 = 0.4619, \\ &m_3 = -0.4157, m_4 = 0.3536, m_5 = -0.2778, \\ &m_6 = 0.1913, m_7 = -0.0975. \end{aligned}$$

由於解碼得到的 DCT 區塊為像素值減去 128 後的值，因此修改式(6)如下：

$$\begin{aligned} S_{\text{BMR}} &= 8m_0m_0(X_{11} + 128\times 8) + 8m_0m_1X_{21} + \\ &\quad 8m_0m_2X_{31} + 8m_0m_3X_{41} + 8m_0m_4X_{51} + \\ &\quad 8m_0m_5X_{61} + 8m_0m_6X_{71} + 8m_0m_7X_{81}, \text{and} \\ S_{\text{RMC}} &= 8m_0m_0(X_{11} + 128\times 8) + 8m_0m_1X_{12} + \\ &\quad 8m_0m_2X_{13} + 8m_0m_3X_{14} + 8m_0m_4X_{15} + \\ &\quad 8m_0m_5X_{16} + 8m_0m_6X_{17} + 8m_0m_7X_{18}. \end{aligned} \tag{7}$$

我們將式(7)計算而得的 S_{BMR} 及 S_{RMC} 相加後除以 8 作為預測值 λ，再依據下列公式計算出新的 DC 值來取代原始 DCT 區塊之 DC 值：

$$\text{DC}_{\text{new}} = Round(\text{DC}_{\text{old}} + 8\times(128 - \lambda)). \tag{8}$$

至於 DCT 區塊之其餘 AC 值部分，則予以保留。下一步驟指定該區塊為 I8MB 並將此區塊設為 DC 預測方式，不做量化處理直接對該區塊進行編碼。圖 4 為開放式架構但不考慮預測值的轉換結果，而圖 5 為同樣架構下考慮預測值的轉換結果，可以看出後者在影像品質上有很明顯的改善。

3.3 轉換失真

所提方法中可能產生轉換失真的原因包括：(1)預測值得不準確；(2)DCT 區塊與整數轉換區塊的差異；(3)編碼整數連續性；及(4)錯誤累積。部分項目的失真或許不會造成畫質上太大的損失，但是在轉換上仍舊需要特別留意，尤其是錯誤累積所造成失真問題，對高解析影像就會有影響。依序說明如下。

3.3.1 預測值的失真

在開放式轉換架構上，預測值的失真來自於省略掉 H.264 編碼重建流程。圖 6 為 H.264 的標準編碼流程，其中虛線部份為重建畫面的流程。重建流程能確保編碼端和解碼端擁有的資料是相同的，避免預測上的誤差。省略掉重建流程就會產生編碼端與解碼端資料上的差異。以 DC 預測模式做說明，編碼端有四個 8×8 區塊為 $\{M_i, i = 1,2,3,4\}$，由於在解碼時會有一定的失真產生，因此假設解碼所得四個 8×8 區塊為 $\{M_i', i = 1,2,3,4\}$。當使用標準的 H.264 編碼流程時，會以編碼重建流程來確保編碼端與解碼端所得到的資料是一致的，因此預測值的取得方法如下(如圖 7)：

(1)對於 M_1 區塊而言，由於不存在上方區塊與左方區塊，因此預測值設為 128；

(2)對於 M_2 區塊而言，存在解碼端的左方區塊(即 M_1')，因此取 M_1' 區塊最右邊一行(RMC)像素值的總和，加上 4 再除以 8 作為預測值，即預測值設定為 $(RMC(M_1') + 4) >> 3$ (假設 $RMC(M)$ 代表取出矩陣 M 最右邊一行所有元素像素值的總和)；

(3)對於 M_3 區塊而言，存在解碼端的上方區塊(即 M_1')，因此取 M_1' 區塊最底下一列(BMR)像素值的總和，加上 4 再除以 8 作為預測值，即預測值設定為 $(BMR(M_1') + 4) >> 3$ (假設

$BMR(M)$ 代表矩陣 M 最底下一列所有元素像素值的總和)；及

(4)對於 M_4 區塊而言，存在解碼端的上方區塊(即 M_2')與左方區塊(即 M_3')，因此取 M_2' 區塊最底下一列像素值與 M_3' 區塊最右邊一行像素值的總和，加上 8 再除以 16 作為預測值，即預測值設定為 $(BMR(M_2')+RMC(M_3')+8)>>4$。

然而對於所提方法裡使用的開放式架構，並無編碼重建流程這個步驟，因此無法使用 $\{M_i', i=1,2,3,4\}$ 這四個區塊來協助預測值的取得，所以方法修正如下(如圖 8)：

(1)對於 M_1 區塊而言，由於不存在上方區塊與左方區塊，因此預測值設為 128；

(2)對於 M_2 區塊而言，以 M_1 取代 M_1' 作為 M_2 之左方區塊，因此預測值設定為 $(RMC(M_1)+4)>>3$；

(3)對於 M_3 區塊而言，以 M_1 取代 M_1' 作為 M_3 之上方區塊，因此預測值設定為 $(BMR(M_1)+4)>>3$；及

(4)對於 M_4 區塊而言，以 M_2 取代 M_2' 作為 M_4 之上方區塊，及以 M_3 取代 M_3' 作為 M_4 之左方區塊，存在解碼端的上方區塊(即 M_2')與左方區塊(即 M_3')，因此預測值設定為 $(BMR(M_2)+RMC(M_3)+8)>>4$。

如上所述可以得知，由於無法使用 $\{M_i', i=1,2,3,4\}$ 的資訊來協助預測值的取得，導致所取得的預測值與真實的預測值會有差異而產生失真。但藉由上述方法的處理，可以降低預測值失真所造成的影響。

3.3.2 DCT 區塊與整數轉換區塊的差異

H.264 所使用的整數轉換是由修正 DCT 轉換公式而來，為了達到整數運算的優勢，實際 H.264 轉換的結果並不完全等於 JPEG 解碼得到的 DCT 區塊，所以在反轉換時，這項差異會造成 H.264 解碼時的失真。詳細內容請參考[1]的整數轉換公式。

3.3.3 編碼整數的失真

H.264 共有三種編碼方法，分別是(1)基於上下文的二元算數編碼（CABAC）；(2)基於上下文的變動長度編碼（CAVLC）；及(3)指數哥倫布（Exponential-Golomb）編碼。指數哥倫布編碼主要用於編碼控制參數，或用於編碼語法元素例如量化參數、移動向量(motion vector)、及參考畫面索引(reference frame index)等。對於實際需要編碼的影像資訊內容，會在 CABAC 或 CAVLC 二者擇一使用，而這兩種編碼方法均採整數型態的編碼方式。H.264 為加速編解碼的處理時間，因而採用整數型態的運算，例如在時域(time domain)轉為頻域(frequency domain)的核心轉換或是量化處理，都是採用整數運算，因此不會有浮點數的進位或捨棄的誤差。但在所提方法中，取得 JPEG 的 DCT 區塊內容是浮點數型態，不論是先採四捨五入轉換為整數型態，或是經過預測值的運算後再做四捨五入轉為整數型態，都是屬於強迫轉換方式而產生進位或捨棄的誤差，導致失真問題的發生。

3.3.4 錯誤累積

在 H.264 畫面內預測(intra-prediction)中，如果依照完整的編碼流程去轉換，每個宏塊的正確性不會因為畫面內預測而產生任何失真，可確保 I 畫面的品質。然而在所提修正之開放式轉換架構中，前面宏塊(macroblock)所產生的輕微失真，就有可能在計算下一個宏塊時，產生預測值上的錯誤，而造成接下來連續宏塊錯誤的累積。這種因錯誤累積所導致的畫面失真，尤其在高解析影像的轉換過程中會更加明顯。

上述的失真產生原因都是無可避免的，也都會多少對最後結果產生影響。在下一節的實驗結果中，可以看出這些失真將導致最後結果的影像品質降低。但即便如此，整體而言都還在可接受的範圍內。對於最後一項失真(即錯誤累積)，在高解析影像的轉換上的確會有較明顯的影響。如何在格式轉換的處理過程中消除錯誤累積，是未來研究的一個主要課題。

4、實驗結果與討論

在本節我們以實驗結果來展示所提方法在轉換速度與高畫質的優勢。4.1 節介紹實驗中所使用的軟硬體設備，及列出 JM 的各項編碼參數設定值。4.2 節說明本論文所採用之影像品質檢測方法。最後，4.3 節列出實驗結果並做分析與討論。

4.1 實驗環境

實驗中所採用的設備包括個人電腦 AMD Athlon 64 X2 Dual Core 4800+，2GB DDRII 記憶體，系統環境為 Microsoft Windows XP SP3。JPEG 解碼使用開放原始碼組織 Saillard [16]所開發的解碼程式(TinyJPEG Decoder)。H.264 編碼與解碼之比較使用 JM 14.2 [17]參考軟體，其中使用 JM 做編碼時，調整參數檔案(encoder.cfg)的幾個重要設定如表 1(共用參數之設定)及表 2(搜尋參數之設定)所示。最後所提方法之撰寫語言為 Dev-C++。

4.2 品質檢測之方法

所提方法在比對品質時使用三種檢驗方法，分別為 *PSNR* (Peak Signal-to-Noise Ratio)、*UQI* (Universal Quality Index; UQI) [18]、及 *MSSIM* (Mean of Structural Similarity Image Measurement; MSSIM) [19]，而比較方式為比對原始 JPEG 解碼得到的 YUV 檔案，與轉換後 H.264 解碼得到的 YUV 檔案，比較兩者之間的差異。*PSNR* 是常見的影像品質檢測方法，單位為 dB，*PSNR* 值愈大代表失真愈少。基本上 *PSNR* 值大於 30 以上，則人的視覺感官就不太會察覺有差異性存在，而 *PSNR* 值大於 40 以上，則人眼幾乎無法分辨其差異。*SSIM* (Structural Similarity Image Measurement;結構相似度影像評估)是一種計算相似度的做法，用來判斷兩張影像是否相似，是一種客觀的影像品質評量方法。*SSIM* 能改善傳統檢測方式如 *PSNR* 或 *MSE* 等方法的缺點，後者的缺點是易受雜訊影響計算結果且無法考慮人眼主觀敏感區的感受。*SSIM* 檢測使影像品質能更符合人眼的感官曲線。*SSIM* 值越接近 1 代表測試影像品質越好。

本論文計算結構相似度時使用 *UQI* 與 *MSSIM*，兩種測量值都是代表一群 *SSIM* 之平均值。作法是將影像 *A* 及影像 *B* 分別切割成 *M* 個 $p \times p$ 大小相等的視窗(window)，對此 *M* 個視窗分別計算其 *SSIM* 值後再予加總。在本實驗中，*UQI* 的 *p* 值設定為 8。*UQI* 的定義為

$$UQI = \frac{1}{M}\sum_{j=1}^{M} SSIM_j \cdot \qquad (9)$$

MSSIM 與 *UQI* 作法相似，但在計算視窗上加入了權重的概念，能避開 *UQI* 可能產生區塊效應(blocking effect)的缺點。*MSSIM* 計算各分割視窗的 *SSIM* 值時，視窗內的每一個像素依所在位置做權重分配，權重值採高斯對稱權重函數(circular symmetric Gaussian weighting function) $G = \{g_i \mid i = 1,2,..., p \times p\}$，其中高斯函數的標準差設定為 1.5，且加以正規化使權重總合為 1(即 $\sum_{i=1}^{p\times p} g_i = 1$)。*MSSIM* 在本實驗中 *p* 值設定為 11。

4.3 實驗結果分析與討論

此節分別對下列三種轉碼方式做比較，分別為(1)使用串聯式架構的 JM_FullSearch；(2)使用串聯式架構的 JM_FastSearch；及(3)我們所提開放式架構方法。為方便起見，三種方法依序簡稱為 JM(I)、JM(II)、及所提方法。實驗結果對轉換所需時間與壓縮品質分別作比較。

測試圖分為三組測試圖，皆為 JPEG 影像。第一組與第二組之壓縮品質(quality factor)設為 94，來源為數位相機所拍攝的影像。第三組之壓縮品質設為 70，來源為 AXIS 網路攝影機所拍攝影像。每組影像均有四種解析度，分別為 QCIF (176×144)、CIF (352×288)、VGA (840×480)、及 4CIF (704×576)。

4.3.1 時間分析

表 3 將三組測試圖片的轉換所需時間列出。JM(I)在轉換速度上表現最慢，JM(II)在轉換速度上略優於 JM(I)。而本文提出的方法在轉換速度上遠勝於 JM(I)與 JM(II)，尤其當影片解析度增加時，JM(I)和 JM(II)的轉換速度大幅增加，更顯出我們所提方法在這方面的卓越性。JPEG 的壓縮品質對於 JM(I)與 JM(II)會產生一定影響，JPEG 品質高時轉碼速度會變慢，反之則變快。但對於本論文所提出的方法則不會有太大的影響，仍維持相同的快速轉換特性。整體而言，本論文所提方法在轉換速度上明顯優於一般串聯式的架構 JM(I)及 JM(II)。

4.3.2 品質分析

表 4 顯示三組測試圖的 *PSNR* 值。對於 JM(I)及 JM(II)而言，在編碼環境 QP 設為 4(即不做量化)的情況下，*PSNR* 值在所有的情況下都能穩定的維持在 50 以上。對於本論文所提兩種方法而言，由於省略了一部分的編碼處理，因此 *PSNR* 值變動的幅度較為明顯，介於 30 到 40 之間。

PSNR 值並不能完全表示一個圖像品質的好壞。假設某一個預測值得到了錯誤的結果，那麼對於該區塊整體都會有影響，錯誤預測值的影響結果通常導致同區塊的像素值一起改變，這對於人眼的感官上也許不會造成太大影響。然而對於 *PSNR* 值而言，整個區塊的像素值改變會計算出一個很糟糕的結果。因此除了比較 *PSNR* 值以外，實驗結果同時計算了 *UQI* 與 *MSSIM* 值做為比較的依據，表 5 為三組測試圖的 *UQI* 值，而表 6 為三組測試圖的 *MSSIM* 值。結果顯示所提方法轉換出來的影像品質都有很好的結果。

5、結論

進入 21 世紀後，電腦逐漸成為生活中普及的配備，數位化串流格式在研究學者的努力下推陳出新，出現了愈來愈多種類的壓縮格式方便使用者利用，於是格式間轉換變的相形重要。本論文提出了一個快速的轉換架構，允許 MJPEG 影像格式轉為 H.264 串流格式，提出的轉換方法運算量低且能夠維持極佳的影像品質。

在未來研究上，希望能解決影像解析度大小影響轉換後影像品質高低的問題，同時希望能降低轉換後所得 H.264 檔案的大小(size)。本論文由於轉換架構簡單且運算量低，未來也希望嘗試在嵌入式系統環境中加入 H.264 的編解碼的流程。

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 version 1," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050, 2003.

[2] Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley & Sons, 2003.

[3] Henrique S. Malvar, Antti Hallapuro, Marta Karczewicz, and Louis Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, July 2003.

[4] Vasant Patil and Rajeev Kumar, "A fast arbitrary factor H.264/AVC video re-sizing algorithm," in *IEEE International Conference on Image Processing*, 2007.

[5] Jun Xin, Anthony Vetro, and Huifang Sun, "*Converting DCT coefficients to H.264/AVC* transform *coefficients*," in *Proceedings of IEEE Pacific-Rim Conference on Multimedia (PCM '04)*, vol. 2, pp. 939-946, Tokyo, Japan, November 2004.

[6] Zhihua He and Maja Bystrom, "Improved conversion from DCT nlocks to integer cosine transform blocks in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 6, June 2008.

[7] Tsung Han Tsai, Yu-Fun Lin, and Hsueh Yi Lin, "Video transcoder in DCT-domain spatial resolution reduction using low-complexity motion vector refinement algorithm," *EURASIP Journal on Advances in Signal Processing* , 2008.

[8] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp.18–29, 2003.

[9] Zhihua He and Maja Bystrom, "A fast algorithm for conversion from DCT blocks to integer cosine transform blocks," *International Conference on Image Processing*, October 2006.

[10] Y. Sun, J. Xin, A. Vetro, and H. Sun, "Efficient MPEG2 toH.264/AVC intra transcoding in transform domain," *IEEE International Symposium on Circuits and System(ISCAS2005)*, pp.1234-1237, May 2005.

[11] H. Kalva and B. Petljanski, "Exploiting the Directional Features in MPEG-2 for H.264 Intra Transcoding," *IEEE Transactions on Consumer Electronics*, vol 52, no 2, May 2006.

[12] H. Kalva, B. Petljanski, and B. Furht, "Complexity reduction tools for MPEG-2 to H.264 video transcoding," *WSEAS Transactions on Information Science*

&Applications, vol.2, Issues, pp.295-300, March 2005.

[13] Z. Zhou, S. Sun, S. Lei, and M. Sun, "Motion information and coding mode reuse for MPEG-2 to H.264 transcoding," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.

[14] S.E. Kim, J.K. Han, and J.G. Kim, "Efficient motion estimation algorithm for MPEG-4 to H.264 transcoder," *IEEE International Conference on Image Processing*, 2005.

[15] T. Qian, J. Sun, D. Li, X. Yang and J. Wang, "Transform domain transcoding from MPEG-2 to H. 264 with interpolation drift-error compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, No.4, April 2006.

[16] *TinyJPEG Decoder* [Online]. Available: http://www.saillard.org/index.php

[17] *H.264/AVC JM Reference Software* [Online]. Available: http://iphome.hhi.de/

[18] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, pp. 81-84, March 2002.

[19] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, Apr. 2004.

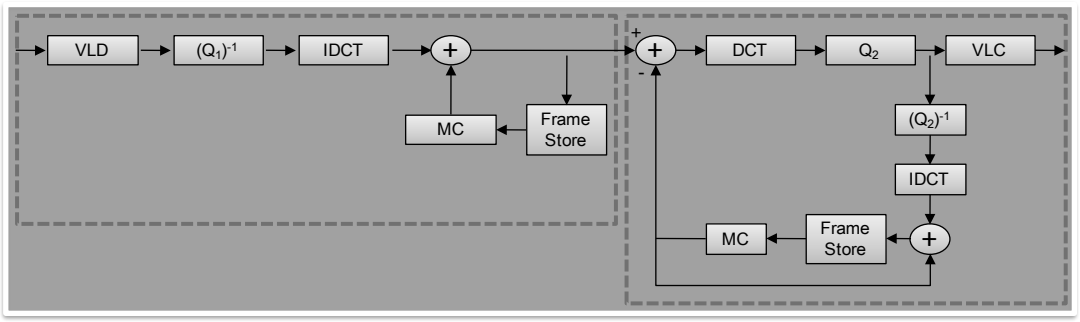

圖 1、影像壓縮格式相互轉換之串聯式架構。

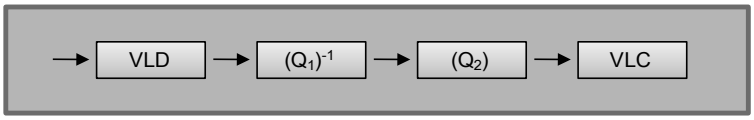

圖 2、影像壓縮格式相互轉換之開放式架構。

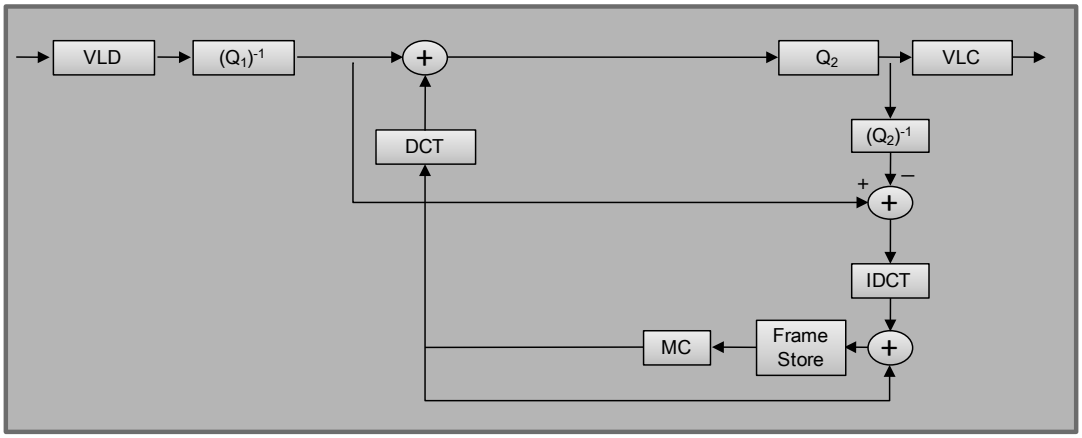

圖 3、影像壓縮格式相互轉換之封閉式架構。

圖 4、無預測值之開放式架構的轉換結果。

圖 5、有預測值之開放式架構的轉換結果。

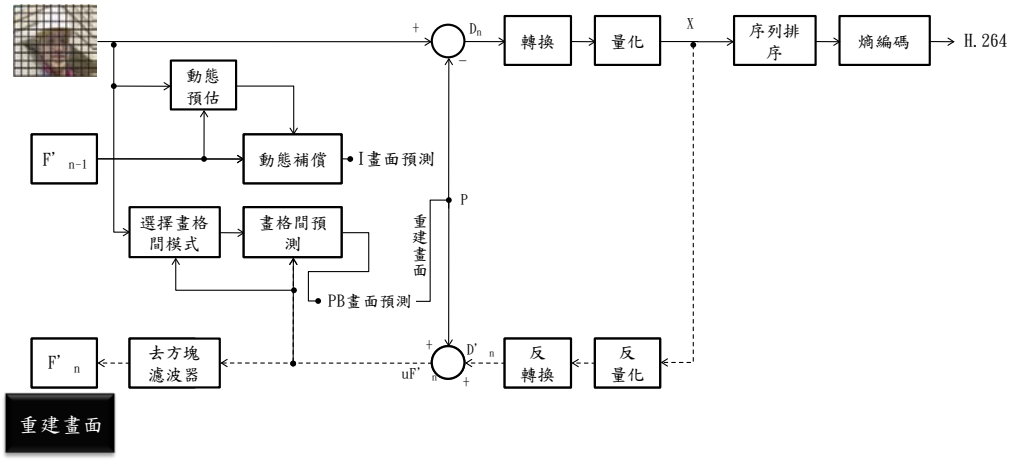

圖 6、H.264 標準編碼流程。

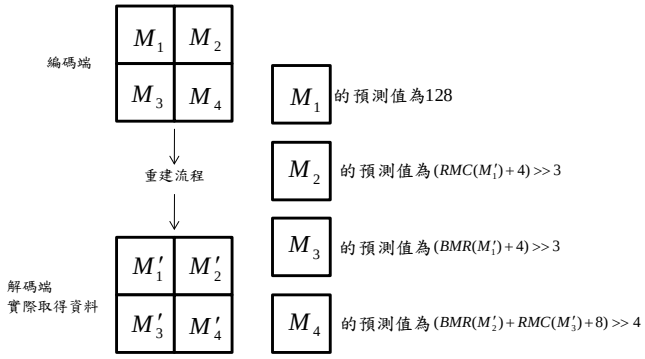

圖 7、重建流程取得的預測值。

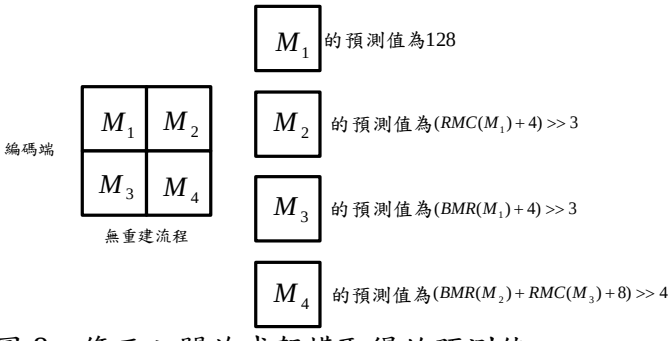

圖 8、修正之開放式架構取得的預測值。

表 1、JM 共用參數之設定。

共用參數之設定			
Profile-IDC	100 (High)	Symbol Mode	1 (CABAC)
Intra-Profile	1	OutFile Mode	0
Level IDC	40	Partition Mode	0 (No partition)
Intra Period	1	YUV Format	1 (4:2:0)
IDR Period	1	Enable IPCM	1
QPI Slice	4		

表 2、快速搜尋參數與全域搜尋參數之設定。

快速搜尋參數設定 (JM_ FastSearch)		全域搜尋參數設定 (JM_FullSearch)	
Disable Intra4x4	1	Disable Intra4x4	0
Disable Intra16x16	1	Disable Intra16x16	0
Disable IntraInInter	1	Disable IntraInInter	0

Intra4x4 ParDisable	1	Intra4x4 ParDisable	0
Intra4x4 DiagDisable	1	Intra4x4 DiagDisable	0
Intra4x4 DirDisable	1	Intra4x4 DirDisable	0
Intra16x16 ParDisable	1	Intra16x16 ParDisable	0
Chroma IntraDisable	1	Chroma IntraDisable	0
SearchMode	-1 (Full)	SearchMode	0 (Fast)

表 3、格式轉換之時間分析(單位：秒)。

第一組	JM(I)	JM(II)	所提方法
QCIF	1.765	0.1695	0.062
CIF	6.856	4.916	0.313
VGA	17.384	13.432	1.079
4CIF	20.454	17.633	1.719
(第二組)	JM(I)	JM(II)	所提方法
QCIF	1.933	1.395	0.078
CIF	6.532	5.214	0.328
VGA	19.074	15.451	1.188
4CIF	24.859	20.361	1.469
(第三組)	JM(I)	JM(II)	所提方法
QCIF	1.272	1.045	0.078
CIF	4.895	4.052	0.313
VGA	12.308	10.138	1.031
4CIF	15.655	15.686	1.452

表 4、三組測試圖之 *PSNR* 分析。

第一組	JM(I)	JM(II)	所提方法
QCIF	58.251	57.333	38.25
CIF	58.024	57.988	39.47
VGA	58.667	58.401	35.79
4CIF	58.588	58.451	32.43
(第二組)	JM(I)	JM(II)	所提方法
QCIF	58.741	57.238	36.21
CIF	58.667	57.833	35.77
VGA	58.871	58.296	32.91
4CIF	58.785	58.386	28.90
(第三組)	JM(I)	JM(II)	所提方法
QCIF	58.547	57.916	37.87
CIF	58.272	58.547	37.17
VGA	59.063	59.002	32.74
4CIF	59.204	59.232	34.86

表 5、三組測試圖之 *UQI* 分析。

第一組	JM(I)	JM(II)	所提方法
QCIF	0.9999	0.9999	0.9987
CIF	0.9922	0.9924	0.9529
VGA	0.9945	0.9945	0.9625
4CIF	0.9960	0.9960	0.9657
(第二組)	JM(I)	JM(II)	所提方法
QCIF	0.9999	1.0000	0.9984
CIF	0.9998	0.9998	0.9953
VGA	0.9999	0.9999	0.9968
4CIF	0.9997	0.9997	0.9803
(第三組)	JM(I)	JM(II)	所提方法
QCIF	0.9999	0.9999	0.9986
CIF	0.9994	0.9994	0.9968
VGA	0.9981	0.9981	0.9923
4CIF	0.9967	0.9967	0.9962

表 6、三組測試圖之 *MSSIM* 分析。

第一組	JM(I)	JM(II)	所提方法
QCIF	0.9999	0.9991	0.9991
CIF	0.9985	0.9985	0.9946
VGA	0.9991	0.9991	0.9910
4CIF	0.9985	0.9985	0.9884
(第二組)	JM(I)	JM(II)	所提方法
QCIF	0.9999	0.9999	0.9979
CIF	0.9996	0.9996	0.9946
VGA	1.0	1.0	0.9969
4CIF	0.9999	0.9999	0.9780
(第三組)	JM(I)	JM(II)	所提方法
QCIF	1.0	1.0	0.9988
CIF	0.9990	0.9999	0.9989
VGA	0.9997	0.9970	0.9994
4CIF	0.9988	0.9988	0.9988