行政院國家科學委員會補助專題研究計畫■ 成 果 報 告
□期中進度報告

使用分頻訊號處理之高速行動多媒體傳收器系統研究與設計

子計畫二:無線網路串流視訊之抗錯與錯誤補償技術研究(3/3)

計畫類別：□ 個別型計畫　　■ 整合型計畫
計畫編號：NSC-96-2219-E-009-002
執行期間：　94　年　8　月　1　日至　97　年　7　月　31　日

計畫主持人：王聖智（交通大學電子工程系教授）
共同主持人：
計畫參與人員：任慈澄、曾禎宇、陳奕安、黃文中、許庭瑋

成果報告類型(依經費核定清單規定繳交)：□精簡報告■完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：交通大學電子工程系

中　華　民　國　97　年　10　月　27　日

使用分頻訊號處理之高速行動多媒體傳收器系統研究與設計

子計畫二:無線網路串流視訊之抗錯與錯誤補償技術研究(3/3)

計畫編號：NSC -96-2219-E-009-002-
執行期限：94年8月1日至97年7月31日
主持人：王聖智 (交通大學電子工程系教授)
計畫參與人員：任慈澄、曾禎宇、陳奕安、黃文中、許庭瑋 (交通大學電子所研究生)

中文摘要

在本次結案報告中，我們將說明此三年計畫所研發之三項成果。

第一項成果偏向硬體實現方面，是我們針對了 H264/AVC 的影像壓縮規格，實現了一個基於 TI DSP 系統的即時影像傳輸系統，其中包含影像接收-壓縮-網路傳送端，以及網路接收-解壓縮-播放端，在一端送出經過 H264/AVC 編碼技術壓縮過的資料，經過網際網路傳輸後可以被另一端收到並進行解碼。我們使用了多線程緒的執行方式，來達成此即時系統。針對 DM642 數位處理晶片，我們提出平行化的方法，並且也對具有多顆數位訊號處理晶片的 MEX 系統做平行化的處理。

第二項成果則是屬軟體演算法層級，是我們改善了原始 H264/AVC 的碼率分配機制。而在此研究中，我們首先針對影像編碼之中碼率的控制加以研究，藉由適當的控制，使得編碼出最適合通道狀態的編碼結果。在本次研究當中我們討論的架構建立在 H.264/AVC 影像編碼標準上，將針對其 R-D 模型加以討論，主要利用 MAD、QP 與編碼數的關係去改變原先的模型，實驗證實能有效的改善傳輸時 buffer 的狀態，更進一步能對影像品質加強，而得到較佳的編碼結果。可以有效避免因為碼率控制不當所造成的編碼瑕疵。

第三項成果同樣是屬於軟體演算法層級，是我們在 H264/AVC 之外，我們提出了一個以模糊邏輯為基礎的動作向量修正技術(Motion Vector Correction with Fuzzy Logic)，讓編碼的動作向量接近真實運動軌跡，藉此提升解碼端錯誤修補的效果。在解碼端我們提出了一種針對整張畫面損失情況的修補方式，我們利用視訊中物體運動的連續性，提出了後向動作投射(Backward Motion Projection)，修補畫面損失對視訊品質的影響。

關鍵詞：H.264/AVC、平行化、最佳化、碼率控制、R-D 模型、錯誤修補、運動向量修正、模糊邏輯。

Abstract

In this report, we introduce three major achievements. The first achievement is related to hardware realization, where we implemented an H.264/AVC based real-time video communication system over a TI DSP system. This system includes video capturing, video encoding, network transmission, video decoding, and video displaying. The H.264/AVC encoded data transmit from one end to the other end. The whole procedure is implemented in terms of multiple threads. To speed up the coding process, the optimization and parallelization of the DSP codes are performed with respect to the DM642 DSP chip and the multi-DSP board, MEX. The second achievement is that we propose some modifications over the rate control technique in H.264 video compression. According to

the adaptable control, we can better encode the video to fit for channel requirement. So far, we have discussed the R-D model of the H.264/AVC video standard. Based on the relation among MAD, QP and the encoded bits, we try to modify the original R-D model. The experiments had shown that based on the new model we may better estimate the coding buffer and thus reduce some apparent coding artifacts. The third achievement is that we propose a motion vector correction technique based on fuzzy logic. In this approach, we modify the encoded motion vectors at the encoder site to better approximate the actual motion trajectories, while perform error concealment at the decoder site to further improve the video quality by using a backward motion projection method based on the continuity of object movements.

Keywords: H264/AVC、 Parallelization, Optimization、Rate Control、R-D model、Error Concealment、Motion Vector Correction、Fuzzy Logic

成果報告

A. H.264/AVC 影像編碼系統在 TI DSP 系統平台上之實現與加速

(1) 背景

隨著視訊壓縮規格的發展，新一代的 H.264/AVC 壓縮規格提供了更適合於視訊通訊。相較於先前壓縮規格，H.264/AVC 不但大幅在影像壓縮層級(VCL: Video Coding Layer)提高了壓縮效率，也提供了網路提取層級(NAL: Network Abstraction Layer)的觀念，以增強了網路適應能力。跟 Motion JPEG 相比較 H.264/AVC 可以減少 80%的影像檔案大小；而與 H.263 相比，則可以減少到 50%的影像檔案大小。但是採取各種有效的編碼方式的同時，其演算法也相對的複雜許多，大幅增加了實現 H.264/AVC 的困難程度。

另一方面，由於 DSP 的高性能以及開發成本較低，DSP 的實現方式是一種廣泛在音訊、影像、圖形、等應用的實現方式。而德州儀器(TI) C64x 系列中的 TMS320 DM642，是採用了超長指令集(VLIW)結構，時脈高達 600MHz 的高性能定點運算晶片，非常適合於數位多媒體的處理。因此，我們將會使用搭載 TMS320 DM642 的數位訊號處理版來實現 H.264/AVC 的影像編碼系統。

在 Vitec Mult-Media 所提供的 MEX 數位訊號處理版上，有著四顆 TMS320DM642。於此，我們會實現完整的視訊系統。除了影像編碼本身，也將真實的影像擷取與播放，以及網路的傳送與接收，都實現在此處理板。

如 Figure A.1.1 所示，我們將從影像裝置擷取類比的影像訊號，經由處理版接收並且編碼後傳輸至網路，並且在另一塊處理板上經過網路接收編碼後的檔案，即時的解碼並且播放到電腦上。

本文首先對 H.264/AVC 影像壓縮規格做簡單的介紹，以及介紹架設整個系統的軟體與硬體環境。接著把整個系統的架構與組成做完整的介紹，並且把核心的 H.264/AVC 影像編碼做加速。以及最後的實驗結果與討論。

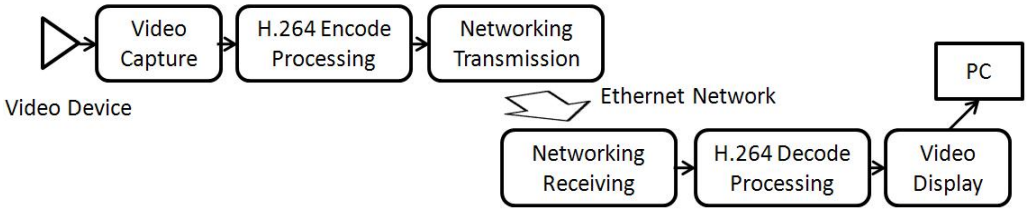


Figure A.1.1 H.264 based Visual Communication System

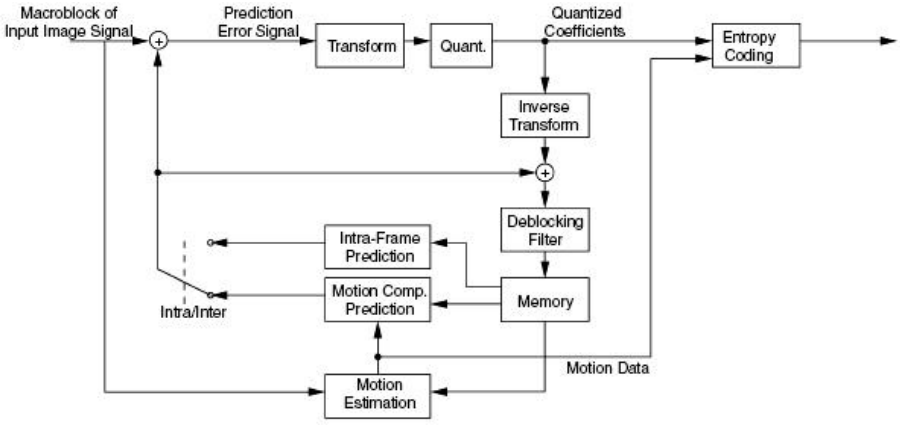


Figure A.2.1 H.264/AVC Encoder [5]

(2) 系統環境

2.1 H.264/AVC Standard

H.264/AVC 視訊編碼標準是由 ITU-T 視頻編碼專家組(VCEG)與 ISO/IEC 運動圖像專家組(MPEG)聯合組成的 JVT(Joint Video Team) 在 2003 年制定。其壓縮效率大幅優於現存的其他壓縮規格，迄今已廣被接受使用。

以編碼端為例，H.264/AVC 編碼的區塊圖如圖 Figure 所示，將每張畫面(Frame)切割成以 16x16 像素點為單位的巨區塊(macroblock)做處理。在編碼過程中，可以選擇畫面內預測(intra prediction)，或是畫面間預測(inter prediction)，去除影像在空間或是時間上的累贅資訊。而原始畫面與預測值相減的殘餘值(residual)會經過轉換、量化，在包成網路提取單元(NAL Unit)前會再經過熵編碼。以下列出其中幾個比較重要的編碼過程:

1. 畫面內預測

為了去除空間上的冗餘資訊，在亮度方面以 4x4 的區塊對較為細緻的部分做預測，而用 16 x16 大小的區塊預測較平滑的區域；而在彩度使用 8x8 大小的區塊做為預測。依照加權值的方向的不同可分為數種模式，4x4 亮度區塊九種、16x16 亮度區塊四種、8x8 彩度區塊四種，分別使用到左邊、上面、或是右上已經編碼好的區塊。

2. 畫面間預測

採用多重參考畫面，以不同大小區塊(共七種支援的大小)在參考畫面中做移動估測，將最佳估測的移動向量以及相減後的殘餘值紀錄下來去做壓縮。其中移動估測可借由內差的方式達到 1/4 個像素的精準度。

3. 轉換與量化

使用 4x4 整數轉換以降低運算複雜度，並且避免反轉換時的誤差。在量化方面提供了 52 個量化值，每個量化步階以 12%增加。

4. 熵編碼

提供兩種熵編碼:內容適應性編碼(context adopted variable length coding, CAVLC)與內容適應性二元算數編碼(context adopted binary arithmetic coding, CABAC)。

5. 方塊效應濾波器

在編碼與解碼端都可使用此濾波器來減少方塊效應，而是否使用濾波器則是由邊界上鄰近的像素值做為判斷。

2.2 Hardware Environment

我們分別在兩台個人電腦上安裝 MEX 數位訊號處理系統版來實現我們的影像編/解碼系統。MEX 是由 Vitec Mult-Media。上面搭載了四顆 TMS320 DM642，並且每顆 DM642 都配置 32MB 大小的同步動態隨機存取記憶體(SDRAM)。此外有兩顆可供調變的 FPGA，是用來設定數位訊號處理晶片到影像晶片，或是數位訊號處理晶片到電腦端的資料傳輸方向。如 Figure A.2.2 所示，除了核心的四顆數位訊號處理

晶片與兩顆 FPGA 之外，與外部的溝通界面則包含了八個影像輸入晶片、四個音訊輸入晶片、網路實體層晶片、以及和個人電腦連接的 PCI 界面。由於強大的核心運算晶片，與完善的輸入輸出介面。我們可以在此數位訊號處理板上實現 H.264/AVC 的影像編/解碼系統。

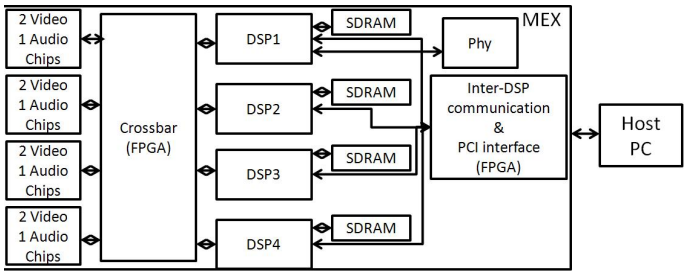


Figure A.2.2 Block Diagram of MEX board [6]

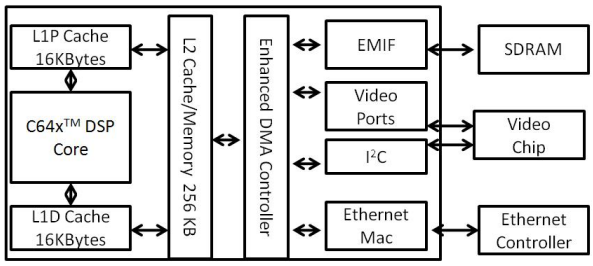


Figure A.2.3 Block Diagram of DSP Chip [9]

而 MEX 板子上面搭載的 DM642 則是工作在 600MHz 的時脈下，擁有 8 個互相獨立的工作單元，所以可以達到高達 4800MIPS 的運算。如圖 Fig. A.2.3 所示，兩層的高速緩衝結構用以提升記憶體存取的效率。而透過 Enhanced DMA 則可以方便地記憶體、中央處理核心與周邊。以下簡述幾個我們會使用到的晶片功能:

1. Memory Architecture

DM642 有兩層的快取記憶體架構，第一層的快取包含了 16K Bytes 的程式碼快取以及 16K bytes 的資料快取。而第二層的快取是一個 256K bytes 的快取，可供資料以及程式碼使用。並且可以調整大小 16、32、64、128K bytes 當作快取，其餘的空間由使用者配置。搭配 EDMA 的使用會更有效率。

2. Enhanced DMA(EDMA)

EDMA 負責在兩層快取記憶體與外部周邊做資料與程式碼的傳輸、交換。與先前數位處理晶片組不同的，在我們使用的 C64x 系列中，EDMA 對於個別不同的裝置有不同的通道，可以設定不同的優先次序，或是設定彼此連結的方式。透過 EDMA 內部記憶體、周邊裝置、與外部記憶體之間可以互相搬運資料。

3. EMIF

在 DM642 中用以配置外部記憶體裝置。而在我們使用的 MEX 板上用於配置 32MB 的同步動態隨機存取記憶體(SDRAM)、以及同步 FIFO 記憶體給每個 DM642。

4. Vide Port

有三個可配置的影像埠，可設定為影像擷取埠、影像顯示埠、或 TSI 擷取埠。經由 EDMA 協助，可把影像資料搬運到記憶體內，以方便做影像編碼處理，也把記憶體內的影像搬運到影像埠傳送到外部。

5. EMAC

EMAC 為 10/100M 乙太網路傳輸控制器，經由此一控制器，便可以控制在 MEX 板上的網路實體層晶片，以達到網路的傳輸功能。

2.3 Software Environment

在本節中我們會簡短的介紹我們系統的軟體開發環境 Code Composer Studio 與一些開發即時系統所需要的軟體環境。

2.3.1 Code composer studio

CCS(Code Composer Studio)除了提供基本的程式碼生成，亦提供即時偵錯以及即時分析的能力，並且有視窗化界面可供使用。在 CCS 下我們進行開發，偵錯、並且最佳化。

2.3.2 DSP/BIOS

DSP/BIOS 讓數位訊號處理晶片的程式開發者去分析與開發即時的程式，主要提供了靜態系統設定、即時排程、即時分析(RTA)、即時資料交換(RTDX)。對於多執行緒的程式而言，即時排程與偵錯是很重要的。而圖形介面化的操作方式，讓程式開發者能更方便的去使用。

2.3.3 Hardware emulation and software simulation

Code composer Studio 提供了在個人電腦上的模擬(simulation)以及數位訊號處理晶片上的仿真(emulation)。使用模擬時只需要設定 CCS 的模擬環境，而不需要有實際的硬體設備，即可以達到開發、偵錯的目的。而使用 CCS 做仿真時，須透過 JTAG 傳輸規格的連接，個人電腦與數位訊號處理晶片會進行即時資料交換(RTDX)。便可以看到數位訊號處理晶片上真正的運作情形。

(3) 硬體實現與加速

前面的章節介紹了軟硬體的使用環境，接下來會描述我們要實現的系統。以及如何在前述的軟硬體下實現這個影像編碼傳輸系統。

3.1 System Architecture

如 Figure A.3.1 所示，整個系統分為兩塊 MEX 板，其中一個版子負責 H.264/AVC 編碼的部分，另一個則負責 H.264/VC 的解碼端。連接兩塊板子達成達完整的影像傳輸系統。

在第一個版子上，在影像裝置擷取類比的影像訊號，經由處理版接收並且編碼後傳輸至網路，並且在另一塊處理板上經過網路接收編碼後的檔案，即時地解碼並且播放到電腦螢幕上。

我們把上述的每一個流程都看做獨立運作的任務(Task)。如 Figure A.3.1 所示，編碼端有三個主要的任務，包含了影像擷取任務、影像壓縮任務，以及網路傳輸任務。解碼端則是網路接收，解壓縮，撥放三個任務。

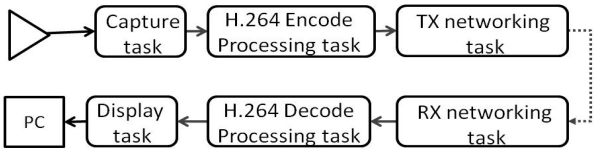


Figure A.3.1 Multitask system architecture

3.2 System Implementations

在上一節中我們把，整個系統分為數個獨立的任務，而把這系統實現在 DM642 時，我們將上述的任務對應 DSP/BIOS 裡面的 TSK 物件(Task Object)。在 DSP/BIOS 的協助下，我們將系統套用到 TI 所提供的參考架構 5 (Reference Framework Level 5)，整個系統以多執行緒(Multi thread)的方式運行。在本節中會介紹參考架構 5 (Reference Framework 5, RF5)的組成單元、以及我們系統中的各個任務包含影像擷取撥放、網路溝通、以及 H.264/AVC 的編解碼實現的方法。

3.2.1 Multi task over Reference Framework 5

參考架構 5 (Reference Framework 5, RF5) 使用 DSP/BIOS 以及 TMS320 DSP 演算法標準。提供使用者開發大量演算法、多執行緒、多通道的應用程式。其最上層的組成單元為任務(Task)，往下依序包含了通道(channel)、小單元(cell)、以及 XDAIS 演算法。在我們實現的系統中，因為使用到了多執行緒的機制，因此適用於 RF5 中。在任務層級有兩種 RF5 溝通單元: 串流輸入輸出(Streaming I/O) 與同步溝通 (Synchronized Communication)，而在小單元層級(Cell)則有跨小單元溝通單元(Inter Cell Communication)，以下做詳細的分述:

1. 串流輸入輸出(Streaming I/O)，如 Figure A.3.1 所示，使用雙緩衝的方式將任務與設備做連接，將設備所得到的資料，迭替地放置到記憶體中，在供給任務使用。

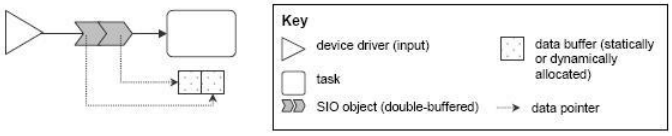


Figure A.3.2 Streaming I/O (SIO)

2. 同步溝通物件(Synchronized Communication, SCOM)則是負責任務與任務間的溝通，可將緩衝記憶體位置告知下一個使用的任務，亦可藉由同步溝通序列(SCOM Queue)，來決定任務間執行的次序，如 Figure A.3.3 所示。

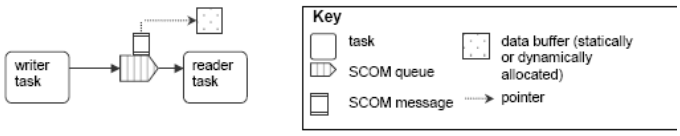


Figure A.3.3 Synchronized communication (SCOM)

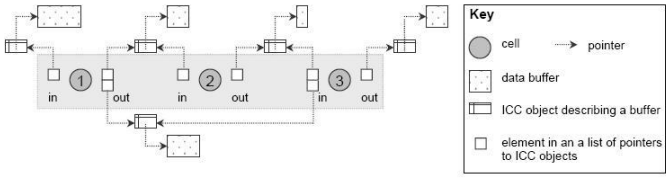


Figure A.3.4 inter-cell communication (ICC)

3. 小單元溝通單元(Inter Cell Communication, ICC)，如 Figure A.3.4 所示，定義了每個小單元的輸出輸入緩衝，用以溝通每個小單元間的資料傳輸。

藉由上述幾個溝通的物件，再加上任務、通道、小單元以及周邊的設備，可完成如圖 Figure A.3.5 所示，藉由 RF5 所架設起來的系統區塊圖。

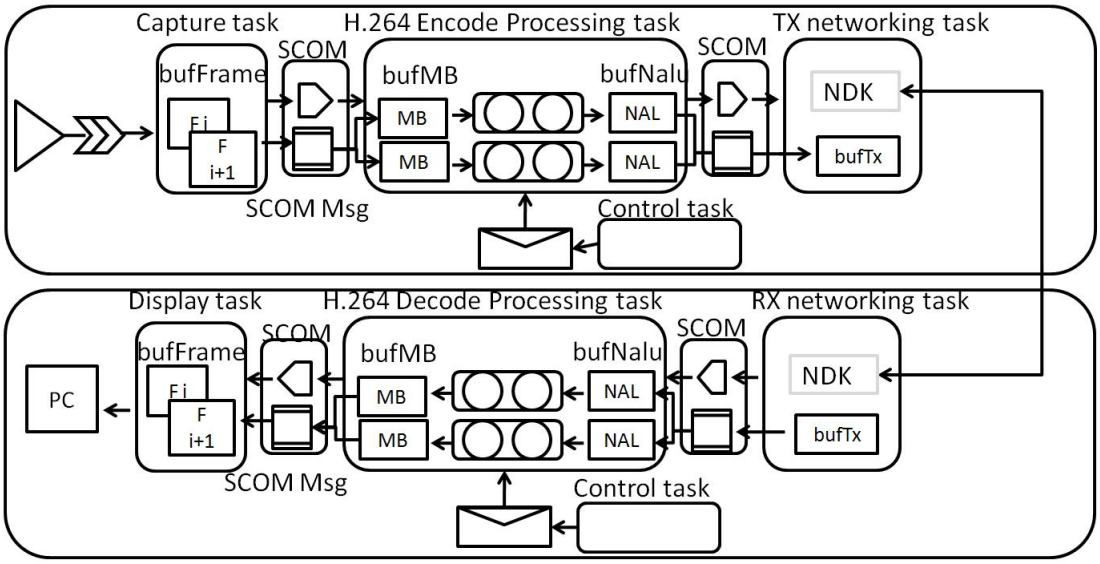


Figure A.3.5 System block diagram of Reference Framework level-5

在本系統中，有八個主要的任務，在壓縮端有三個主要的工作任務(包括:影像擷取、影像壓縮、網路傳送)以及一個控制任務。另外在解壓縮端也有三個主要的工作任務(包括網路接收，影像解壓，影像播放)以及控制任務。壓縮端的三個主要任務經借由 SCOM 彼此溝通，達到多執行緒的運作方式；同樣地解壓縮端的三個主要任務也藉由 SCOM 彼此溝通，以判斷執行的時機。以下對主要的任務分別做簡單的介紹。

3.2.2 Video capture and video display

影像擷取任務主要是設定影像晶片(SAA7113)，並將擷取的數位影取得到記憶體中使用。設定影像晶片的方式是使用 DM642 上的 I^2C 匯排流調整影像參數。設定好 EDMA 對應影像埠的通道，將擷取到的影像透過 EDMA 移動到外部記憶體，並且將影像規格從 YUV422 轉換成 YUV400。

影像播放任務則是將 YUV400 轉換回 YUV422 並且以 EDMA 搬運到由 EMIF 定義的 FIFO 外部記憶體。此 FIFO 可由主機電腦上存取的到，並且在主機端撰寫 Win32 視窗介面化程式，用以播放解壓縮完的檔案。

3.2.3 Network developer's kit

在網路通訊方面，我們採用了由德州儀器所提供的網路開發套件(Network Developer's Kit, NDK)。NDK 只需耗用 200~250KB 大小的程式記憶體，以及 95KB 的資料記憶體。即可達到完整的網路傳輸功能。

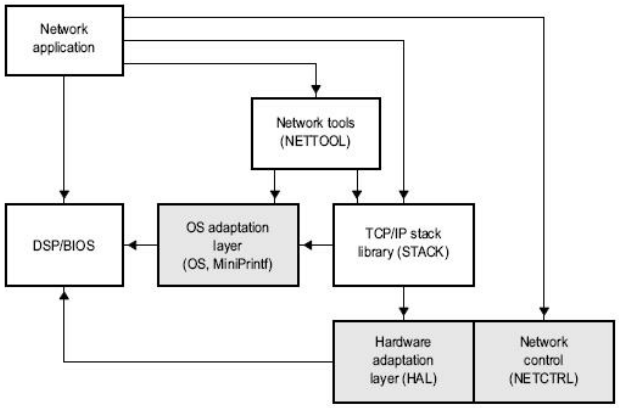


Figure A.3.6 Stack Control Flow

如圖 Figure A.3.6 所示，五個網路功能對應的函數分別為：STACK.LIB, NETTOOL.LIB, OS.LIB, HAL.LIB, 和 NETCTRL.LIB。STACK.LIB 是從上層至下層包含了主要的 TCP/IP 網路功能。NETOOL.LIB 包含了所有插口式網路服務 NDK，以及一些應用程式。OS.LIB 做為與 DSP/BIOS 對應的函式。而透過 HAL.LIB 做為接面，可以連接硬體周邊。而 NETCTRL.LIB 控制著 TCP/IP 與外界聯繫與互動。由上述五個函式庫彼此合作即可達成 TCP/IP 的網路傳輸。

3.2.4 H.264/AVC source code

在編碼端我們使用的是 x264 的軟體，而在解壓縮端我們使用的是 JM 的軟體。其中 JM 是 ITU 所提供的官方參考軟體，雖然功能較為完善，但是運算速度卻相當的慢，若是用來當編碼端，恐難達到即時編碼的速度。另一方面，x264 則是一自由軟體，因為 x264 較低的運算複雜度，x264 廣被接受，因此我們也特別使用 x264 來實現編碼器。在 DM642 上處理 QCIF 大小時編碼器與解碼器分別花費的時間如下表所示。

Table A.3.1 Profile of each function decoder.

	x264 encoder Cycle count	%
Inter	385955140	66.80
Intra	76744048	13.28
DCT/IDCT	36449650	6.31
Quantization.	27206566	4.71
Deblocking filter	10887230	1.88
Entropy coding	9221182	1.60
Total	577769679	100

Table A.3.2 Profile of each function of encoder.

	JM decoder **Cycle count**	**%**
Inter	55418473.24	26.96
Intra	3740215.647	1.80
IDCT	7638903.176	3.72
Quantization	42497.23529	0.02
Deblocking filter	115774629	56.32
Entropy coding	22219077.35	10.81
Total	205535479	100

3.3 System optimization on DSP

經由 RF5 以及個別地完成上述的功能，我們可以把整個 H.264/AVC 影像編碼系統架設起來。雖然說在系統中影像編碼與解碼主要是各別開發並最佳化，但類似的最佳化方式會被使用在壓縮端與解壓縮端，以下我們一併介紹所採取的最佳化的手法:

1. 編譯器最佳化(Compiler Optimization):我們使用 CCS 中最高的最佳化層級 3(-o3)，此時編譯器(Compiler) 針對迴圈會使用許多的最佳化的方式: 軟體導管化(Software pipeline)、迴圈展開，甚至是運用到 SIMD 的去做編譯。
2. 軟體導管化(Software Pipeline)

為了增加編譯器在採取軟體導管化時的效率，需要給編譯器額外的資訊。如使用“#pragma MUST_ITERATE”，或是” #pragma UNROLL (n)” 來告知編譯器迴圈要拆解幾次，可以避免編譯器在最拆解時多出了冗餘的迴圈，讓軟體導管化更好。此外，對於多重迴圈來說，編譯器只能拆解最內部的迴圈，外部的迴圈無法自動做拆解，因此我們也手動做了迴圈拆解的動作，增加軟體導管化的程度。

3. 配置記憶體位置:在程式的執行中會有常存取的資料，如熵編碼中所查的表、畫面間預測的小數精準度的宜動向量時的內插。因為要使用 NDK 而無法將所有 ISRAM 指定為 Cache[17]，至少要配置 128KB 當做一般內部記憶體，因此我們可以配置這些常存取的資料到內部記憶體。所使用的語法主要有，針對資料區塊使用" DATA_SECTION” 以及針對程式區塊使用

#pragma CODE_SECTION”來做記憶體的配置。

4. 本質函數(intrinsic functions)的使用。本質函數是經過組合語言層級最佳的程式，可供給C/C++開發者使用。在這邊我們使用了 max_2、min_2、abs_2 等的本質函數，直接取代原本的 C 函數即可。

5. 開啟快取(Cache)

由於使用 NDK 之故無法完全將記憶體配置給快取使用，故在我們的系統中使用了 128KB 大小的快取。

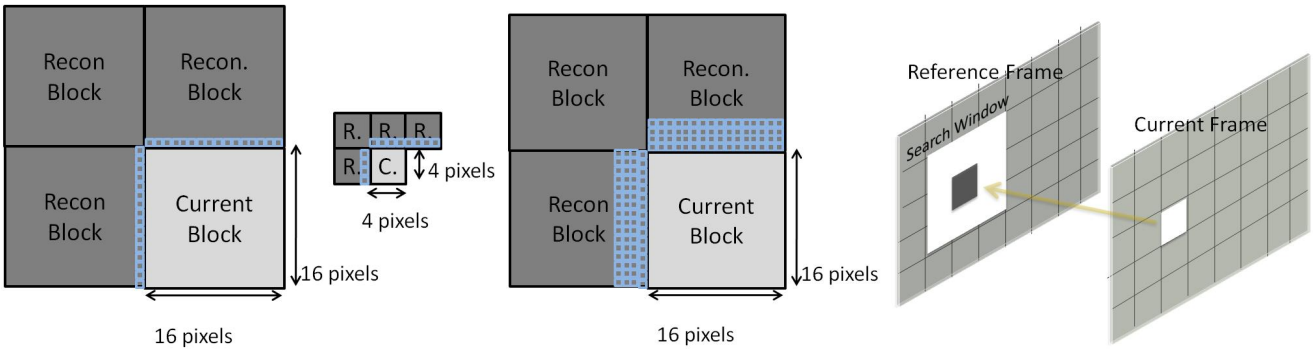


Figure A.3.7 Data dependency induced by(a)Intra prediction (b)Deblocking filtering (c)Inter prediction.

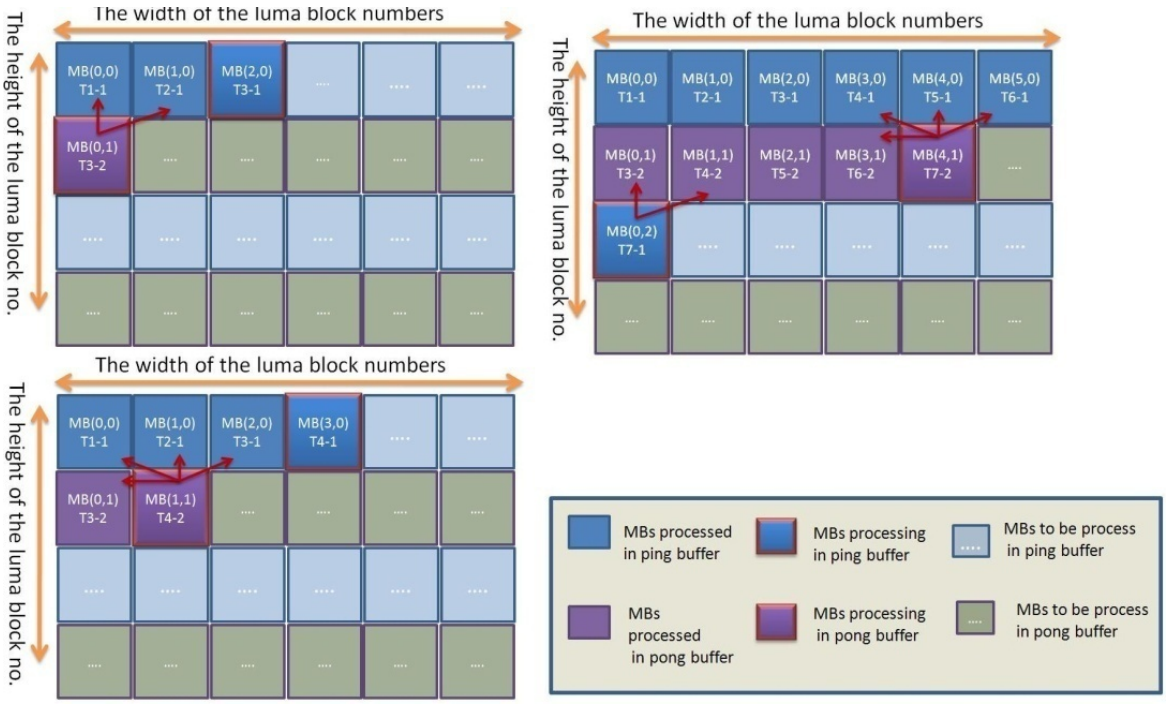


Figure A.3.8 Single DSP macroblock parallelization

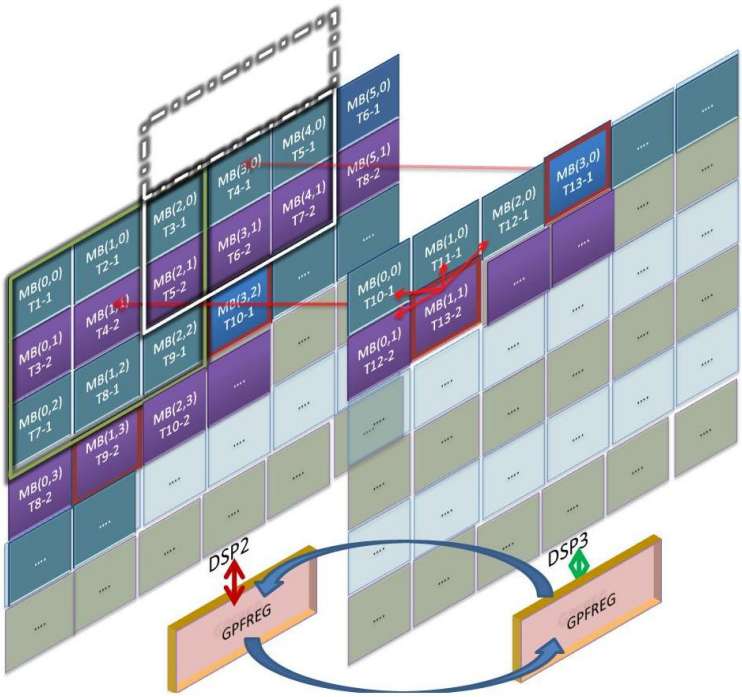


Figure A.3.9 Multi DSP macroblock parallelization

3.4 Parallelization of H.264/AVC

在使用上一節中所提到的最佳化的方法之後，我們仍要做更進一步的加速。因此我們在最核心的地方，也就是 H.264/AVC 的部分考慮加速的方式。在這裡我們考慮如何將 H.264/AVC 在我們使用的硬體上做平行化的處理。

3.4.1 Parallelization of H.264/AVC

在平行化之前，必須先把欲平行化的單位與單位間的相關性排除。在本文中我們以巨區塊(macroblock)做為單位，而巨區塊與巨區塊之間的相關性如 Figure A.3.7(a)~(c)所示。畫面內預測需要以上方的巨區塊，左方的巨區塊以及右上方的巨區塊當作參考。而去方塊效應濾波器則需要上方與左方的巨區塊。而在做畫面間預測時則須要參考畫面中移動搜尋範圍內的所有巨區塊。若要達成 H.264/AVC 的平行化，則需要滿足這些相依性。

3.4.2 Parallelization over One DSP

在單一晶片中，由於 EDMA 的幫助，可以達成雙緩衝的機制(double buffer)，如 Figure A.3.10 所示。借由 ping 緩衝區與 pong 緩衝區交互作用，在 EDMA 搬運 pong 的資料時，ping 可以做運算，反之亦然。所以可以減少整體運算時間。

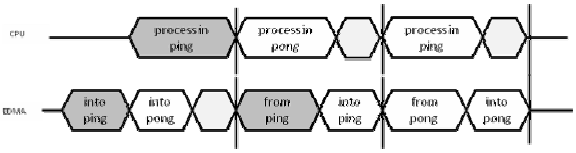


Figure A.3.10 Ping-Pong buffering diagram

在單一晶片中在，我們實行空間上的平行化，主要需要滿足畫面內預測與以及去方塊濾波器所產生的資料相依性。如 Figure A.3.8 所示，將欲平行化處理的兩個巨區塊分別放到 ping 緩衝區與 pong 緩衝區裡，在時間點 N~1 時 ping 緩衝區裡的資料被處理，而隨即在 N~2 的時間點 pong 緩衝區的資料也被處理，中間省去了記憶體搬運的時間，這樣一對 ping 與 pong 的緩衝區稱為一對，在 Figure A.3.10 中可以看到，時間點 3 時間點 4 與時間點 7 的幾個 ping-pong 對。

3.4.3 Parallelization over Multi DSP

而在多晶片中我們實現時間上的平行化處理。前一小節中，我們克服畫面內預測與去方塊效應濾波器所產生的資料相依性。同樣的，我們

以 Figure A.3.9 所表式的方式克服由畫面間預測所產生的資料相關性。當欲執行 I+1 frame 中(1,1)與(3,0)巨區塊時，第 I frame 的(3,2)之前的巨區塊都要先執行完畢。因此在此時 I+1 frame 中(1,1)與(3,0)巨區塊以及 I frame 的(3,2) (1,3)的巨區塊可以同時處理。

(4) 模擬結果

4.1.1 Single DSP Optimization

為了了解程式最佳化的效果，我們使用模擬器去量測最佳化前與最佳化之後，H.264/AVC 各個重要功能所增進的速度以及整體增進的速度。比較結果如 Table A.4.1,Table A.4.2 所示，在壓縮端或是解壓縮端，畫面內預測以及畫面間預測都會有不錯的速度提升，這是因為此兩功能大量了使用重複的迴圈，經由最佳化的展開迴圈之後，有著不錯的效果。然而如去方塊效應濾波器則是因為大量的資料存取而比較慢，所以在最佳化之後無法有明顯的效能提升，也導致速度只增加五倍。

4.1.2 Single DSP Parallelization

我們使用 ping-pong 緩衝機制以達成的平行化的處理，與使用模擬器模擬的 4.1.1 不同，此部分使用的是仿真器模擬，可得出真正硬體實現後的速度。然而如 Table A.4.3，Table A.4.4 所示，經過單晶片的平行化增進的速度只有 1.199 以及 1.033，主要是因為 EDMA 搬運的時間與 CPU 運算的時間相差甚遠，所以能夠減少的 EDMA 搬運時間相對於整體時間只有一小部分，導致增進的效能不彰。

4.1.3 Multi-DSP Parallelization

Table A.4.5 是多顆 DSP 平行化處理 H.264/AVC 壓縮解壓縮的情形。因為相較於壓縮端，解壓縮端的運算時間遠小於壓縮端的運算時間。因此我們只對壓縮端做多顆 DSP 的平行化。我們可以看到增進的效能只達到 1.63 倍。主要是因為 DSP 與 DSP 之間的傳輸沒有經過硬體上的加速，導致平行化的效果不彰。若要更進一步增加平行化的效果則需要針對 DSP 間的傳輸做修改。

Table A.4.1 Average execution cycle of a frame of x264 encoder

	Non optimized		Optimized		
	Cycle count	Percentage	Cycle count	Percentage	Ratio
Inter	385955140	66.80	31345836	58.71	12.3
Intra	76744048	13.28	5435452	10.26	14.1
DCT/IDCT	36449650	6.31	2470440	4.66	14.7
Quantization.	27206566	4.71	2718834	5.13	10.0
Deblocking filter	10887230	1.88	2424248	4.58	4.49
Entropy coding	9221182	1.60	2430716	4.59	3.79
Total	577769679	100	52979974	100	10.91

Table A.4.2 Average execution cycle of a frame of JM10.3 decoder

	Non optimized		Optimized		
	Cycle count	**Percentage**	**Cycle count**	**Percentage**	**Ratio**
Inter	55418473.24	26.96	4163457	10.82	13.3
Intra	3740215.647	1.80	244461	0.64	15.3
IDCT	7638903.176	3.72	1135516	2.95	6.7
Quantization	42497.23529	0.02	14976	0.039	2.8
Deblocking filter	115774629	56.32	17790497	46.25	6.5
Entropy coding	22219077.35	10.81	7875195	20.47	2.8
Total	205535479	100	38467286	100	5.3

Table A.4.3 Single DSP parallelization of x264 encoder

	Non- parallelized	Parallelized	Ratio
ms per frame	491.26	475.53	1.033

Table A.4.4 Single DSP parallelization of JM10.3 decoder

	Non- parallelized	Parallelized	Ratio
ms per frame	76.44	63.73	1.199

Table A.4.5 Multi DSP parallelization result

	One DSP (original)	Two DSP	Three DSP	Four DSP
ms per frame	475.53	397.55	323.81	290.75
Speed up ratio	1	1.196	1.4685	1.6355

(5) 結論與未來工作

在本文中，我們在 MEX 版子上建立了一個 H.264/AVC 的影像通訊系統，與其他只做 H.264/AVC 的不同而直接使用檔案做為 IO 不同，我們以多執行緒的方式實現了影像擷取、壓縮、網路傳輸、網路接收、解壓縮以及影像撥放的系統。我們主要建置一個較現實的 H.264/AVC 系統。將系統以多執行緒的方式實現，並且將系統對於編譯器做最佳化，以及提供了單一晶片可能平行化的方式，最後還有多顆晶片平行化的實現。

未來若要增加速度，可以再對程式進行組合語言的撰寫。若需要讓平行化更有效率，要改善 DSP 與 DSP 間的溝通機制。

Reference

[1] JVT “Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264– ISO/IEC 14496-10 AVC),” March 2003, JVTG050 available on http://ip.hhi.de/imagecom_G1/assets/pdfs/JVT-G050.pdf .

[2] R. Schäfer, T. Wiegand and H. Schwarz, “The Emerging H.264/AVC Standard”, EBU Technical Review, Jan. 2003.

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

[4] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, “Overview of the H.264/AVC Video Coding Standard,” IEEE Trans. on Circuits Syst. Video Technol., Vol. 13, No. 7, pp.560 – 576, July 2003.

[5] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockammer and T. Wedi, “Video Coding with H.264/AVC: Tools, Performance, and Complexity”, IEEE Circuits and Systems, Vol. 4, No. 1, 2004.

[6] www.vitecmm.com, Preliminary of MEX

[7] www.blackhawk-dsp.com/Usb560bp.aspx, Preliminary of USB 560BP

[8] Texas Instruments, “TMS320C6414T, TMS320C6414T, TMS320C6416T fixed point Digital Signal Processor”, Literature Number SPRR226, November 2003.

[9] Texas Instruments, “TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor: Data manual ”, Literature Number SPRS200G, July 2002 – Revised August 2004.

[10] Texas Instruments, “TMS320C6000 CPU and Instruction Set Reference Guide”, Literature Number SPRU189F, January 2000.

[11] Texas Instruments, “TMS320C64x DSP Two-Level Internal Memory Reference Guide”,

Literature Number SPRU610C, August 2004.

[14] Texas Instruments, “Video port/VCXO Interpolated Control (VIC) Reference Guide”, Literature Number SPRU629F, January 2007.

[15] Texas Instruments, “TMS320C6000 DSP Ethernet Media Access Controller (EMAC)/Management Data Input/ Output (MDIO) Module Reference Guide”, Literature Number SPRU628A, March 2004.

[16] Texas Instruments, “TMS320C6000 Code Composer Studio Tutorial” Literature Number SPRU301C, February 2000

[17] Texas Instruments, “TMS320C6000 DSP/BIOS User's Guide”, Literature Number SPRU303B, March 2000

[18] Texas Instruments, “TMS320C6000 Optimizing Compiler User’s Guide” Literature Number SPRU187G, March 2000

[19] Texas Instruments, “TMS320C6000 TCP/IP Network Developer’s Kit (NDK) Technical Data Quick Reference GuideTMS320C6000 TCP/IP Network Developer’s Kit (NDK) Technical Data Quick Reference Guide”, Literature Number SPRU568, October 2001

B. H.264/AVC 之碼率控制技術研究

(1) 背景

現今通訊系統的發達，使得及時視訊傳輸的重要性變的更加重要，而龐大的視訊資料在有限的頻寬下傳送，必須仰賴高效率的壓縮技術，目前較廣泛被研究與應用的壓縮技術，是由 JVT(Joint Video Team) 組織所制定出的 H.264/AVC(Advance Video Coding)影像壓縮標準，它提供了比先前所有壓縮標準更高的壓縮效率，特別在高壓縮率的應用，因此在無線通訊上的應用，使用 H.264/AVC 可以提供相當好的壓縮效果。

而在通訊的傳輸上，通道將會影響影像的壓縮品質，低頻寬所可以傳送的資料較少，因此影像品質會較差，而在高頻寬則能傳送較佳的影像品質。在影像的編碼端提供了一個機制，可以依照頻寬的特性調整影像的壓縮率，進而提供最適合當時頻寬的壓縮品質，這個機制稱之為碼率控制(Rate Control)。在圖 Fig. B.1 中，當資料做完 Entropy Coding 後，會先送到 encoder 的 buffer 等待送入 channel，太多的資料量會使得 buffer overflow，而太少的資料量則會造成使用的效率過低，因此我們往往藉由控制壓縮時的量化係數(Quantization Parameter)去達到最佳的使用效率，這個過程就是所謂的 Rate Control。在過去的影像壓縮標準都已經制定出適合的 Rate Control Model，例如：MPEG-2 的 TM5[1]、H.263 的 TMN8[2] 和 MPEG-4 的 VM-18[3]。而在 H.264/AVC 標準當中，目前也提供了一套 Rate Control 的機制在 Joint Model(JM)[4]之中。而我們演算法的架構將會建立在此 Joint Model 上面。

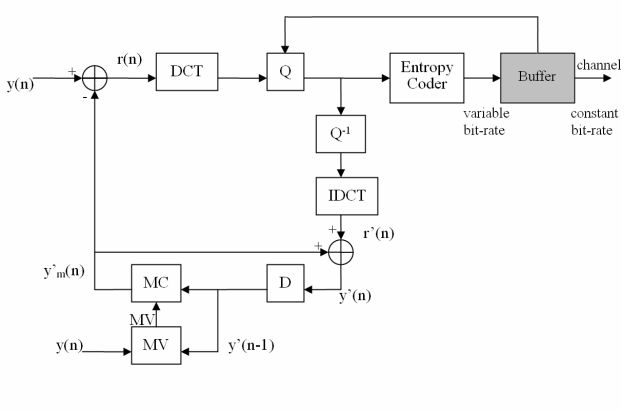


Fig B.1 Rate Control 在影像編碼架構圖

(2) 架構

2.1 Joint Model 中 Rate Control 的架構

在 JM 的 Rate Control 的架構當中主要分為三個階段。1.GOP(Group of pictures)層級，2.Pictures 層級，3.Basic Unit 層級。在 GOP 層級當中會計算此 GOP 總共應該編碼出的資料量，例如：如果頻寬為 64Kbits，每秒傳送 15 張畫面，一個 GOP 大小為 30 張時，則一個 GOP 內可使用的資料大小則為 128Kbits，而在此層級會控制使得壓縮出來的資料量不會超出原本預期的大小。另外，在此層級也會決定每個 GOP 第一張畫面的 QP 值。而在 Picture 層級，將會決定出每張畫面的 QP 值，主要會分成 Stored 跟 Non-stored 畫面，在 Stored 畫面會利用當時 buffer 的狀況跟這個 GOP 剩下可用的資料量來決定出這張畫面的 QP，數學式可以表示成：

$$\tilde{T}_i(j) = \frac{R_i(j)}{f} + \gamma \times (S_i(j) - V_i(j)) \quad \gamma = 0.5$$

$$\hat{T}_i(j) = \frac{W_{p,i}(j-1) \times B_i(j)}{W_{p,i}(j-1) \times N_{p,r} + W_{b,i}(j-1) \times N_{b,r}} \qquad (1)$$

$$T_i(j) = \beta \times \hat{T}_i(j) + (1-\beta) \times \tilde{T}_i(j) \quad \beta = 0.5$$

在這邊，T 代表要用來編碼的 Target Bits，而 R 代表通道每秒可傳送的資料量，f 則是每秒須送幾張畫面，S 是預期 buffer 飽滿的程度，而 V 是當時 buffer 飽滿的程度，B 代表當時 GOP 內剩下可以用的 bits 數，W 代表畫面的複雜度，而 N 則是畫面的數量。當決定好 Target Bits 後，則需要一

個 model 來預測要用多大的 QP 才能達到此 Target Bits，在 JM 裡面所使用的是之前在 MPEG-4 當中被提出來使用的 quadratic Rate-Distortion model[5]，數學式如下：

$$T_i(j) = c_1 \times \frac{\tilde{\sigma}_i(j)}{Q_{step,i}(j)} + c_2 \times \frac{\tilde{\sigma}_i(j)}{Q^2_{step,i}(j)} - m_{h,i}(j) \qquad (2)$$

在這裡 σ 表示做完運動補償後兩張畫面的 MAD(Mean of Absolute Difference)值，而 m 表示 header bits 的大小，T 則是之前已經求得的 Target Bits，經由這個式子可以求得出 Quantization Step size(Q_{step})的值，最後經由簡單的轉換就可以得到 QP 值。而在 Basic Unit 層級當中，將會更進一步決定每個 Basic Unit 的 QP 值。首先說明 Basic Unit 的定義，Basic Unit 是由一張畫面中連續的幾個 macroblock 組成的單位，假設今天使用 Basic Unit 的大小為 11 個 macroblock，則整張畫面將會被分成數個 Basic Unit，以 QCIF 影像(176x144)為例，共有 99 個 macroblock，將會被分成 9 個 Basic Unit，所以在此層級當中將會決定出每個 Basic Unit 的 QP 值。如果 Basic Unit 的大小愈小，則愈能夠滿足一張畫面 Target Bits 的值。上面簡單的介紹完在 JM 裡面所提供的 Rate Control 後，本研究將會針對裡面的架構作調整，進而希望能得到更佳的控制。

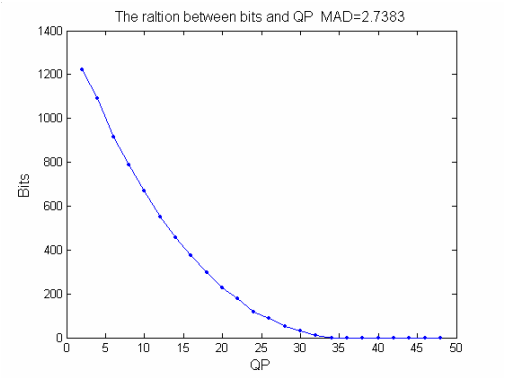


Fig. B.2 固定 MAD 時，QP 跟 Bits 的關係

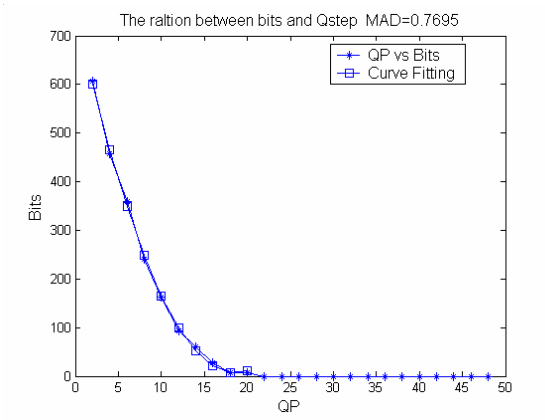


Fig. B.3 MAD=0.7695，QP 跟 Bits 的關係

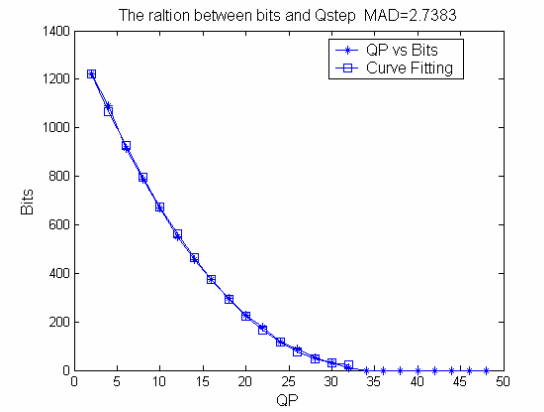


Fig. B.4 MAD=2.7383，QP 跟 Bits 的關係

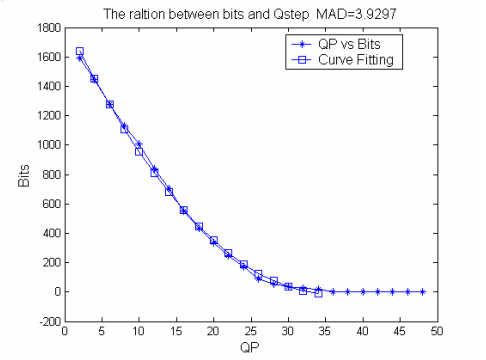


Fig. B.5 MAD=3.9297，QP 跟 Bits 的關係

2.2 Rate-Distortion model

在上面已經介紹過目前在 JM 裡面所使用的 quadratic R-D model，而在最近的研究裡面有其他人提出不同的 R-D model[6][7][8]，而我們將會先分析 QP、MAD 跟壓縮出資料量(Bits)關係，在 Fig. B.2 當我們固定 MAD 之後，可以發現 QP 跟 Bits 之間可以用二階的多項式來描述，因此接下

來我們利用二階多項式針對不同的 MAD 值作實驗，觀察是否皆能符合此特性，Fig. B.3、B.4、B.5 各代表當 MAD 等於 0.7695、2.7383 和 3.9297 時的結果，各可以代表小中大的 MAD 值。而二階的多項式可以表示如下：

$$Bits = a \times QP^2 + b \times QP + c \qquad (3)$$

在這裡 a、b、c 代表此多項式的係數，我們用最小平方誤差可以求得此係數，因此不同的 MAD 可以得到不同的係數。

接下來我們利用大量的數據去觀察這些係數和 MAD 的關係。Fig B.6 是針對不同的影像與不同的 MAD 值所得到 a 跟 MAD 的關係圖，在這邊我們用簡單的一維線性來描述此關係；Fig. B.7 則是 b 跟 MAD 的關係圖，可以明顯的發現 b 跟 MAD 並沒有明顯的關係存在，所以係數我們假設維持不變，最後 Fig. B.8 則是 c 與 MAD 的關係圖，在此可以發現有很重要的關係存在，經過不同的數學式實驗，最後發現使用下面式子得到的誤差最小，而且對整體式子簡化最有幫助：

$$c = z_1 \times MAD + z_2 \times \sqrt{MAD} \qquad (4)$$

接下來可以將上面得到的關係，將原本的式子(3)改寫如下：

$$\begin{aligned} Bits &= a \times QP^2 + b \times QP + c \\ &= (x_1 + x_2 \times MAD) \times QP^2 + y_1 \times QP \\ &+ z_1 \times MAD + z_2 \times \sqrt{MAD} \\ &= k_1 \times MAD \times QP^2 + k_2 \times MAD + \\ &k_3 \times QP^2 + k_4 \times \sqrt{MAD} + k_5 \times QP \\ &= k_1 \times A^2 B^2 + k_2 \times A^2 + k_3 \times B^2 + k_4 \times A + k_5 \times B \end{aligned} \qquad (5)$$

$$A = \sqrt{MAD} \quad B = QP$$

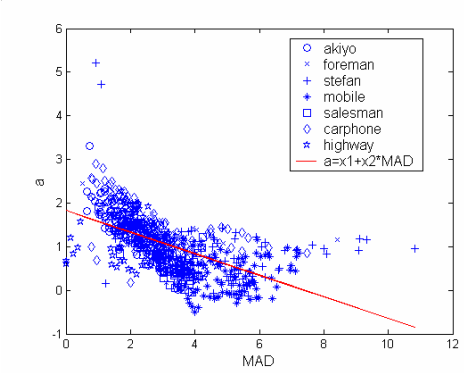


Fig. B.6. a 與 MAD 之間的關係

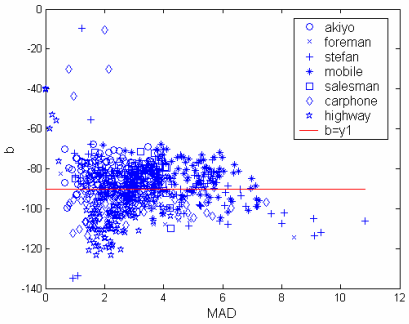


Fig. B.7 b 與 MAD 之間的關係

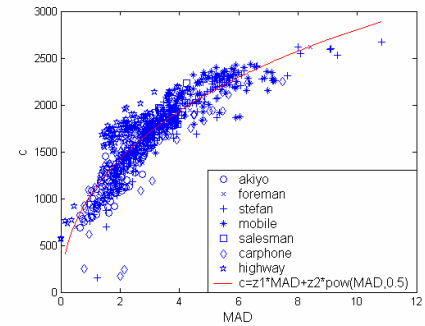


Fig. B.8 c 與 MAD 之間的關係

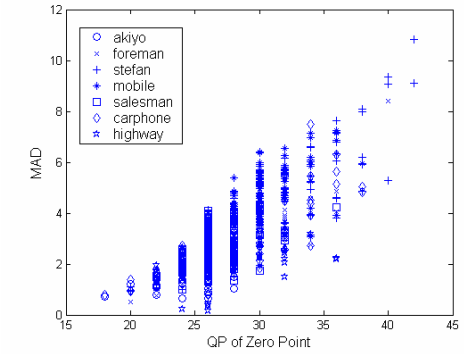


Fig. B.9 Zero Point 與 MAD 關係圖

而在上面的式子裡面只適用在 non-zero 的壓縮結果，因為當 QP 值超過某個值時，編碼出來的量都會是零，從 Fig. B.3 到 Fig. B.5 可以看得出來，當 QP 超過某個值後壓縮出的 Bits 數都會是零，在這邊我們稱此 QP 為 Zero Point，而 Fig. B.9 是 Zero Point 與 MAD 的關係，由圖可以發現 Zero Point 與 MAD 值呈現正比關係，也就是說愈大的 MAD 值需要愈大的 QP 才會編碼為零，所以可以利用此線性關係針對式子(5)加以限制，改寫如下：

$$Bits = k_1 \times MAD \times QP^{t^2} + k_2 \times MAD + k_3 \times QP^{t^2} + k_4 \times \sqrt{MAD} + k_5 \times QP^t \quad (6)$$

$$QP = \begin{cases} QP^t & QP^t \le \dfrac{MAD - h_1}{h_2} \times \sigma \\ QP^t - 1 & QP^t > \dfrac{MAD - h_1}{h_2} \times \sigma \end{cases}$$

上面我們已經建立出了壓縮資料量跟MAD與QP之間的關係，在這邊的壓縮資料指的是做完 DCT 轉換、量化一直到 Entropy Coding 後的資料，尚未包含 Header Bits，而在 Rate Control 決定的 Target Bits 是兩者一起包含的數量，所以接下來我們將針對 Header Bits 的特性作一些分析。

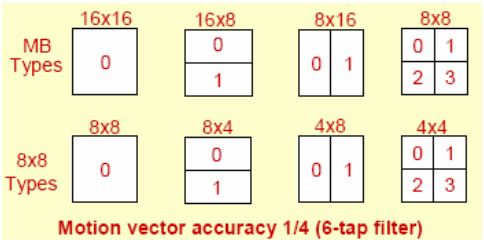


Fig. B.10. Macroblock 種類，使用於移動補償[9]

2.3 Header Bits 分析

首先說明在 H.264/AVC 的 Inter Prediction 提供有多種的 Macroblock Mode，能夠針對不同的影像內容找到最適合的 mode，進而能增加 inter prediction 的效能。Fig. B.9 就是 Macroblock Mode 的種類，愈大的愈適合使用在影像細節較少的部分，而愈小的 block size 則適合在較複雜的部分。而不同的 Macroblock Mode 所需儲存的 Header Bits 也有不同，Fig. B.11 針對 16x16、16x8、8x16 和 8x8 四種形式的 Header Bits 跟 MAD 的比較，在這邊可以發現 16x16 的 Header Bits 比其他三種來的低，而且跟 MAD 大小沒有關係，而 16x8 跟 8x16 兩種形式的 Header Bits 大小相當接近，而且也跟 MAD 大小關係不明顯，但是在 8x8 的 macroblock mode, 可以發現 Header Bits 數量與 MAD 呈現正比的關係，要說明這個原因首先要先分析 Header Bits 的組成成分。由表一可以發現，因為 16x16 只需要存一個移動向量，而 16x8 跟 8x16 需要存兩個移動向量，所以後面的 Header Bits 平均來看會比前者來的高，而在 8x8mode 裡面因為可以更細分為四種 8x8 的 types，所以需要儲存的移動向量差異較大，從四個到十六都有，而我們更進一步觀察造成 Header Bits 差異是否是因為移動向量的差別，根據 Fig. B.12 我們可以了解切的愈細的 block 需要存愈多的移動向量，而所產生的 Bits 數也會愈多，另一方面愈大的 MAD 也愈容易切成較細 block mode，所以在這邊可以將 8x8mode 的 Header Bits 用一階線性的數學式來描述此關係：

表一. P-slice 的 Macroblock Header Bits 成分

成分	說明
Macroblock skip run	記錄是否為 skip
Macroblock type	記錄為何種 moe
Motion vector	記錄移動向量資訊
CBP(Coded Block Patterns)	記錄被編碼為零的資料
Delta QP	記錄跟前者 QP 的差

$$Hbits_{8\times8} = a \times MAD + b \quad (7)$$

而其他的三種 Macroblock Mode 的 Header Bits 則用平均值來代表即可，因此我們將原來 JM 裡的 R-D model(2)改用上面的 model，接下來作實驗觀

察最後得到的結果。

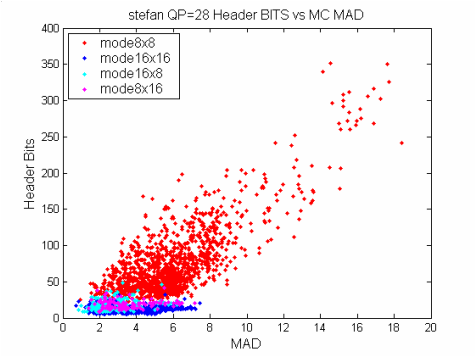


Fig. B.11. Header Bits 與 MAD 之間關係

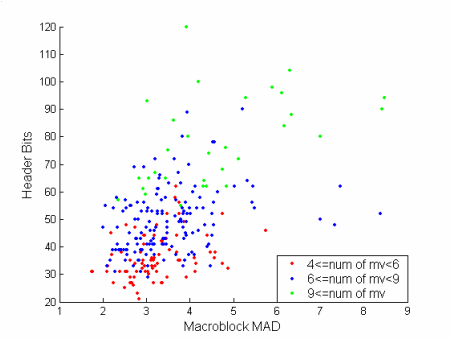
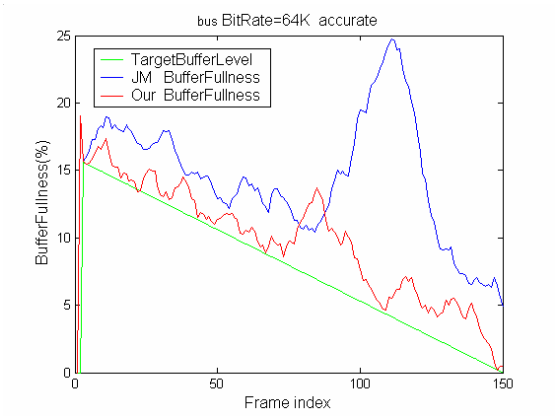


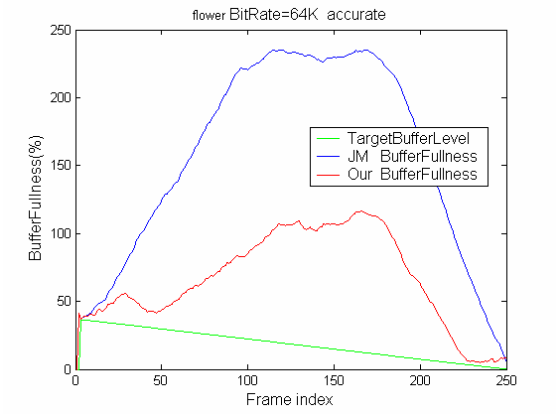
Fig. B.12. 8x8mode 中，不同數目的移動向量和 Header Bits 與 MAD 之間關係

(3) 實驗結果

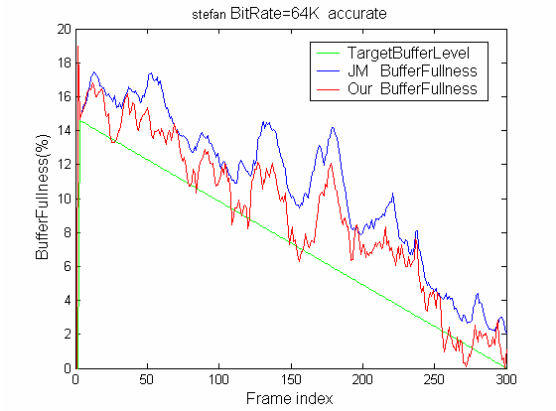
我們將我們的 model 跟原本 JM 的 model 作比較，首先設定 Basic Unit 的大小為一個 macroblock，畫面的大小使用 QCIF，Profile 設定在 baseline profile，Frame Rate 設定為 30f/sec，Buffer 大小設定為頻寬的 0.5 倍，GOP 大小為所有的畫面數，也就是只有第一張為 I-frame，而 RDO(Rat Distortion Optimization)為開啟。在這邊我們選擇的影像為 Bus、Flower 和 Stefan，頻寬為 64Kb。Fig. B.13 為 buffer 額滿的情形，在這邊我們先忽略當 buffer 額滿而造成的畫面遺失的問題，Target Buffer Level 是指預期希望 buffer 的位置，而使用我們的 model 跟 JM 原本的比較，可以發現改變 R-D model 後，對於 buffer 更能有效率的控制，也能比較少發生畫面遺失的問題，而這是因為改變 R-D model 後對於符合



(a) Bus sequence



(b)Flower sequence

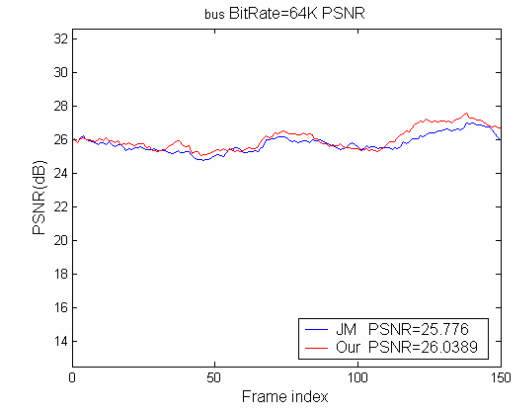


(c)Stefan sequence

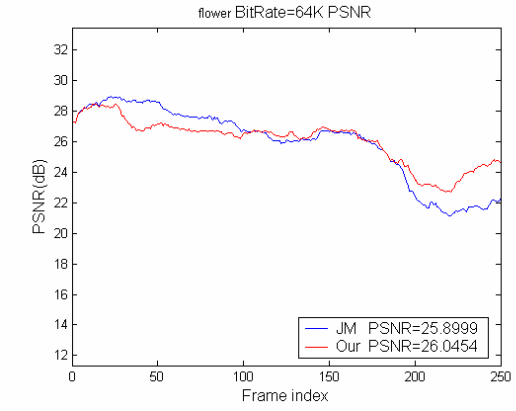
Fig. B.13. Rate Control 過程 Buffer 的狀態

表二. R-D model 預測精確度比較

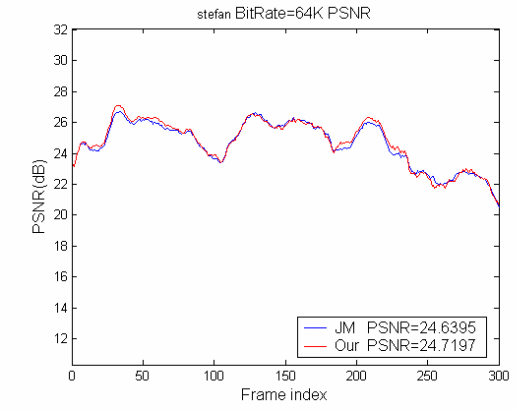
Sequence	JM	Our	Improvement
Bus	569.7	193.7	0.66
Flower	1185.9	995.2	0.16
Stefan	245.2	163.28	0.33



(a) Bus sequence



(b) Flower sequence



(c)Stefan sequence

Fig. B.14. PSNR 的比較

Target Bits 的誤差有相當的減少，在表 2 我們定義誤差為實際編碼出來的數量跟預期編碼數量的平均絕對差值，改善比較是計算改進量對於原本 JM 誤差的比值，可以發現都有不小的改善，接下來將針對每張畫面的 PSNR 作比較。經過 model 精確度的改善，可以發現對於畫面的品質並沒有造成不好的影響，反而對於 GOP 最後面的幾張影像，因為 buffer 有效的控制，使得畫面品質有不小的提升，而對於整體 PSNR 看來也都能有不錯個改善，最後我們比較此改善對於實際視覺影像上的改進，Fig. B.15 到 Fig. B.17 是其實際壓縮出來的影像，可以很明顯的觀察出來改善的結果，特別是像 Bus 跟 Stefan 在地上的線段，還有 Flower 右半邊花的部分，這是因為當編碼到畫面的一半時，不精確的 model 很容易就將這張畫面所分配到的 Target Bits 用完，而造成剩下 Basic Unit 會使用過大的 QP 來減少編碼出過多的資料數，所以對於畫面中後面的 Basic Unit 我們能效果不錯的改善。

(4) 結論

在這篇報告當中，我們針對 H.264/AVC Rate Control 的 R-D Model 分析，然後我們根據實驗觀察出 MAD、QP 跟 Bits 之間的特性，而設計出 R-D Model，能有效的利用在影像的傳輸上，不但能安全的控制 buffer 的狀態，更能對於影像品質有明顯而重要的影響，分析完 R-D Model 的特性，更進一步的我們需要去討論如何決定 Target Bits，不論是以畫面為單位還是 Macroblock 為單位，決定不同的 Target Bits 也會影像畫面的品質，而這是下一步需要去研究的部分。

(5) 參考文獻

[1] ISO/IEC JTCI/SC29/WG11, *Test Model* 5, 1993.

[2] ITU-T/SG15, *Video Codec Test Model*, TMN8, Portland, June 1997.

[3] MPEG-4 Video Verification Model V18.0, Coding of Moving Pictures and Audio N3908, ISO/IEC, JTC1/SC29/WG11, Jan. 2001.

[4] Z. Li et al., "Adaptive Basic Unit Layer Rate Control for JVT," JVT-G012, 7th Meeting: Pattaya, Thailand, March 2003.

[5] H. J. Lee and T. H. Chiang and Y.-Q. Zhang, “Scalable rate control for MPEG-4 video,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 878-894, 2000.

[6] Thung-Hiung Tsai and Jin-Jang Leou, “A Rate Control scheme for H.264 Video Transmission.” IEEE International Conference on Multimedia and Expo (ICME), 2004.

[7] Satoshi Miyaji, Yasuhiro Takishima and Yoshinori Hatori, “A Novel Rate Control Method for H.264 Video Coding.” IEEE ICIP vol. 2, 11-14 Sept. 2005

[8] Siwei Ma and Wen Gao, “Rate-Distortion Analysis for H.264/AVC Video Coding and its Application to Rate Control.” IEEE Trans. on Circuits and Syst. For Video Technol.,vol.15,No.12,Dec. 2005

[9] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, “Overview of the H.264/AVC Video Coding Standard,” IEEE Trans. on Circuits. Syst. Video Technol.,vol.13,No.7,July 2003

(a) JM

(b) Proposed approach

Fig. B.15 Bus 第 145 張畫面

(a) JM

(b) Proposed approach

Fig. B.16 Flower 第 220 張畫面

(a) JM

(b) Proposed approach

Fig. B.17 Stefan 第 50 張畫面

C. 以動作向量修正為基礎的抗錯性編碼技術與以損失畫面修復為基礎 的錯誤修補技術

(1) 簡介

隨著視訊壓縮技術的提升，如 H.264/AVC 的發展[1-2]，增進了編碼的效益，同時 H.264/AVC 也讓視訊傳輸更有效率。另一方面由於無線傳輸技術的進步，如 3G 行動通訊系統的普及，無線網路視訊傳遞的發展，如視訊會議或是視訊電話都是常見的應用。而在使用無線網路傳遞視訊時，最容易遇到的問題之一就是傳輸時的錯誤造成傳送的封包損失。

由於壓縮視訊解碼時後面接收的畫面會以前面的畫面當作參考，當封包損失時，封包內的資料都會損失，這會讓之後參考這些損失的資料的畫面發生錯誤，而又進一步讓參考錯誤畫面的繼續錯下去，這種現象我們稱為錯誤傳遞。

錯誤傳遞會降低解碼的視訊品質，但是由於通常使用無線網路傳遞的視訊需要即時的解碼，例如視訊電話，這種情況下我們沒有辦法要求重新傳送損失的封包。因此為了克服這樣的問題，常見的一種方式是在解碼端使用錯誤修補技術(Error Concealment)，利用之前或是之後接受到的資訊對損失的部分進行估測。

傳統的錯誤修補技術是修補以巨方塊(Macroblock)為單位，利用空間周圍資訊修復損失的巨方塊[9-17]，但是當應用層面為低位元的視訊傳輸時，封包的損毀很有可能會讓整張畫面的資訊都損失[22]，這種情況下就無法使用空間周圍附近的資訊針對巨方塊進行修復。這時候必須在畫面層級對整張畫面用時間上的相鄰資訊進行修復，由於畫面前後有著動作連續的特性，因此我們可以利用動作資訊來進行畫面之修復。

因為要使用動作資訊進行錯誤修補，編碼時的動作向量(motion vector)就是我們最容易使用的資訊，但是由於一般編碼方式是採用 block matching 的方式，因此動作向量不見得真的符合真實運動狀況，這樣所修補的畫面就無法達到最佳狀況。為了讓動作向量接近真實運動軌跡，我們提出了動作向量修正技術(Motion Vector Correction)，利用模糊邏輯判斷動作向量的可信度。

另一方面，我們提出了一種新的畫面修補的方式，採用後向的動作投射(Backward motion projection)，可以利用比傳統方法簡單的做法，達到整張畫面的估測，降低錯誤傳遞對於視訊品質的影響。

(2) 背景

2.1 H.264/AVC in Wireless Environment

H.264/AVC 是由 ITU-T 視頻編碼專家組(VCEG)與 ISO/IEC 運動圖像專家組(MPEG)聯合組成的 JVT(Joint Video Team)所提出的視訊編碼標準，是目前廣泛使用的視訊編碼標準[1]。

經過壓縮過後的視訊會以封包為基礎(packet-based)進行傳輸[3]，每一個封包裡面所包含的是數個巨方塊的編碼資料，如果經過無線網路傳輸，由於網路狀況不穩定，所傳送的封包很有可能會在過程中損失。對於 H.264/AVC 來說，若傳輸的封包損失，會造成整個封包裡的資料都會損失，也就是封包中所裝的巨方塊都會無法解碼出來。封包損失不單只損失巨方塊的畫面會受到影響，由於編碼時我們利用了時間域上的連續性進行壓縮，例如 Inter prediction，後面的畫面會採用前面的畫面作為參考，當參考畫面中的巨方塊損失時，連帶會影響到之後解碼的畫面，而再更後面畫面又會繼續影響更後面的畫面，這種關係我們稱之為錯誤傳遞(Error Propagation)。為了讓傳送時的錯誤對於解碼後之視訊品質影響降低，一般會在解碼端採用錯誤修補(Error Concealment)來修補損失的部分，接下來我們會在之後的小節介紹基本的錯誤修補技術。

2.1.1 Simulation Tools for Video Transmission

為了要瞭解編碼後的視訊經過傳輸過程之後，當傳輸過程中發生封包損失會對解碼後的視

訊造成怎樣的影響，我們可以採用一些傳輸的模擬。由於傳輸時封包會發生損失的情況，所以在[4][5]中就是模擬在特定封包損失機率下對於解碼視訊所造成的影響。不過在這些模擬過程中，封包的損失是採用固定機率損失的方式，如果要更進一步接近無線網路的傳輸狀況[6][7]，可以再加入真實的網路傳輸或是模擬，如

Fig. C.2.1 所示[7]。

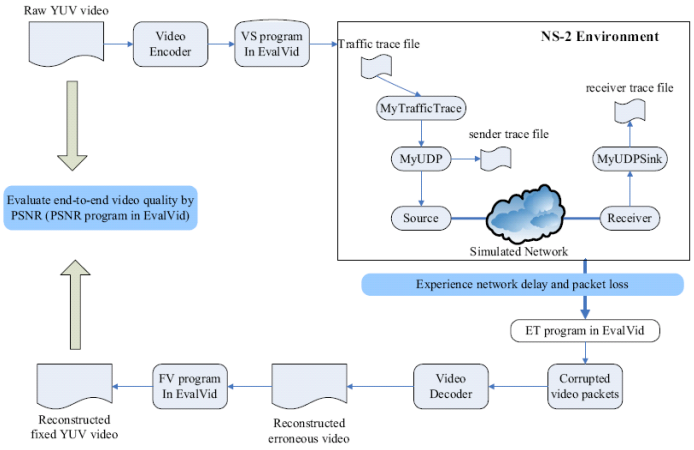


Fig. C.2.1 Schematic of evaluation framework with NS-2 [7]

2.2 Error Concealment

在前一小節中我們介紹了壓縮視訊在透過無線網路傳輸之後，封包損失會造成錯誤傳遞而影響解碼視訊的品質。而這種錯誤傳遞的現象會一直等到新的 Intra Frame 出現時才會停止，所以當傳輸過程有封包損失時，除非我們要求重新傳送損失的封包，否則一定會解碼視訊品質造成不好的影響。但是如果是在即時影像傳輸播放情況下，傳輸與解碼是在同時進行的，封包的重送會造成解碼時的時間延遲，也會造成記憶體使用上的額外負荷，所以一般我們並不允許封包的重送。在這樣的情況下，我們必須仰賴其他的技術才能降低封包損失對於解碼視訊所造成的傷害，而解碼端錯誤修補技術(Error Concealment)就是一種常被使用的方式。

2.2.1 Block-Level Concealment

傳統的錯誤修補方式是將損失的畫面以巨方塊為層級進行修補，假設與損失的巨方塊相鄰的仍正確接收的情況下，可以利用周圍資訊來修補損失的部分，通常可以分成空間域的修補，例如空間上的內插方法[9-11]；或是時間上的修補，例如 Boundary Matching Algorithm(BMA)[12-13]；或是整合性的時間與空間修補方法[14-15]。也有在空間域進行動作向量的修補，利用鄰近的動作向量修補損失部分的動作向量的做法[16-17]。

但是前面我們提到的這些做法都必須利用空間相鄰的資訊修補損失的部分，但是當損失情況嚴重到周圍也都損失時，甚至整張畫面都失去時，上面的方法就無法採用，這時候我們就需要其他的演算法針對整張畫面失去的情況進行修補。

2.2.2 Frame-Level Concealment

當壓縮視訊在低位元傳輸情況下時，舉例來說：當我們使用 3G 行動通訊系統情況下，我們使用 64kbits/channel 傳送每秒 10 張的 QCIF 畫面，換算之後平均每張畫面約為 800 bytes，而一般使用的封包大小約為 1 kbyte，因此一個封包可以裝入整個畫面，所以當封包損失時就會造成整個畫面的損失[22]。

在我們蒐集的資料中，針對整張畫面進行修補的演算法最早是由 S. Belfiore *et al.* 在所提出的[22]，論文中提到在他們資料中，之前沒有針對全畫面進行修補的方法，所以在這篇論文著作時間(2003)之前沒有對於全畫面修補的技術，這原因主要和 H.264/AVC 的發展以及低位元壓縮視訊傳輸發展的狀況有關，在這之後發展的相關論文有[24][25][26]。這些技術主要都是在時間域上以 Optical Flow 為基礎，進行動作向量之修復，我們將在後續的小節中做更詳細的討論。

2.2.2.1 Optical Flow

Optical Flow (OF)的基本概念是假設在視訊畫面連續的情況下，畫面中的亮度值(intensity)在

不同時間點只是做了位移，亮度值本身沒有改變，因此一般最常見的亮度與 OF 之間的關係可以用式子(2.1)表示[27]。

$$\frac{dx(s_H,s_V,t)}{dt}=0 \qquad (2.1)$$

$$\frac{\partial x(s_H,s_V,t)}{ds_H}v_H(s_H,s_V,t)+\frac{\partial x(s_H,s_V,t)}{ds_V}v_V(s_H,s_V,t)+\frac{\partial x(s_H,s_V,t)}{dt}=0 \qquad (2.2)$$

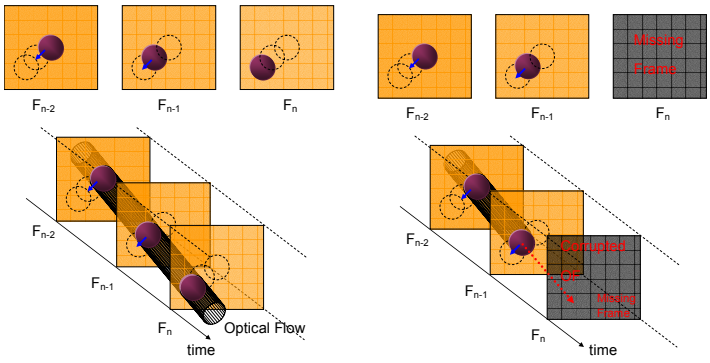
如果把整個視訊想成以空間上的垂直水平軸再加上時間軸三個維度所構成的空間，如**錯誤! 找不到參照來源。**所示，而各個不同時間點的畫面其實可以想成是在時間軸上做取樣。觀察**錯誤! 找不到參照來源。**(a)我們會發現在不同時間點的畫面中圓球位置從中間逐漸往左下移動，由於畫面前後間隔時間短暫，也就是取樣時間短暫，我們可以把圓球的運動當成一種連續的運動，會看到在三維空間中就是一整束的 Optical Flow (OF)，而整個視訊就是由許多一束一束的 OF 所組成的，在不同時間點取樣我們就會得到不同的畫面。

Optical Flow 的偵測可以用上面我們提到的公式計算，但是由於我們應用的層面是已經經過壓縮編碼的視訊，當中已經有 MV(motion vector)提供了畫面中的動作資訊，另一方面由於 OF 的偵測是因為我們在解碼端要做錯誤修補，但是由於應用層面是需要即時處理的視訊解碼，此時太過複雜的做法並不適合採用，因此通常會採用 MV 替代 OF 的方式。

但是我們在此還是得特別說明一下，一般我們編碼時的 MV 是採用 block matching 的方式，到參考畫面中去尋找最符合待編碼的方塊，把測得的位移量當作 MV，但是這並不代表找出來的 MV 就是真正的運動軌跡，當畫面中實際上是有亮度的變化，或是發生物體形變、遮蔽或是進出畫面等情況，我們找到的 MV 其實都很有可能與實際運動有很大差距，此時如果我們運用一般 OF 特性時，就會很容易出錯，因此我們在後面的章節中將介紹如何修正這種錯誤的動作資訊。

2.2.2.2 Motion Recovery of Missing Frame

當封包損失的情況發生時，會造成畫面損失，此時也可想成 OF 發生了如斷層般的情況，如 Fig. C.2.3 所示，在封包損失的同時，由於動作向量的損失，後續的畫面找不到與前畫面對應的關係，如 Fig. C.2.3 (b)所示。換言之如果我們能夠修補斷層的 OF，我們就可以重建損毀畫面與前畫面之間的動作關係，就可利用動作補償來對損失的畫面進行修補。



(a) Illustration of motion consistency (b) Frame lost and optical flow corruption

Fig. C.2.2 The relationship between optical flow and motion consistency

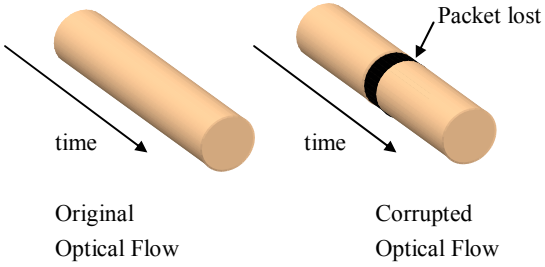
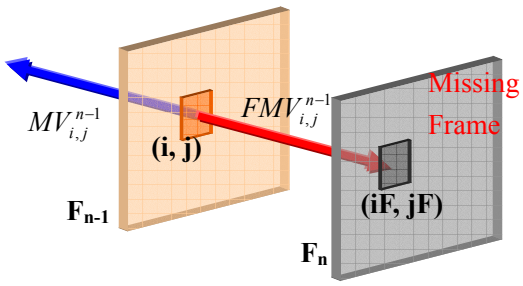


Fig. C.2.3 The Relationship between packet lost and optical flow corruption

由於動作向量有著類似 OF 的特性，因此 S. Belfiore et al.提出了一種前向動作投射的方法[22]，將前一張畫面的動作向量投射到損失的畫面中，如 Fig. C.2.4 所示。



$$FMV_{i,j}^{n-1} = -MV_{i,j}^{n-1}$$
$$MV_{iF,jF}^{n} = MV_{i,j}^{n-1}$$
$$\text{where} \begin{cases} iF = i + FMV_{i,j,H}^{n-1} \\ jF = j + FMV_{i,j,V}^{n-1} \end{cases}$$

Fig. C.2.4 Forward motion projection and MV recovery

系統中為了要讓 MV 接近真實的運動軌跡，會採用空間域的 median filter 對 FMV 進行修正再進行投射，而在 Belfiore 的架構中，被投射的損失畫面是以像素為單位進行修補的。為了提升修補效率 Baccicht *et al.*提出了以方塊為基礎的修補方式[25]，如**錯誤! 找不到參照來源。**所示。

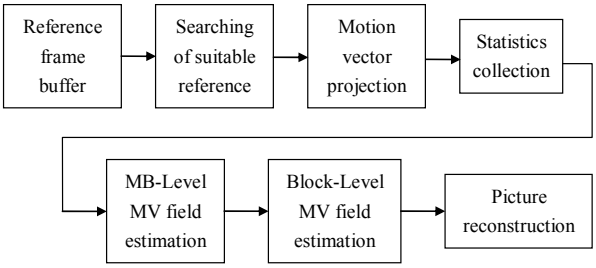


Fig. C.2.5 Schematic of Baccicht's algorithm [25]

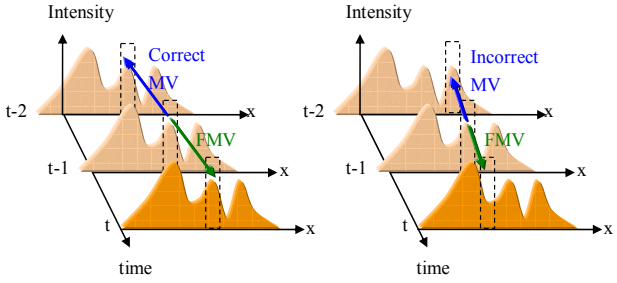
(3) Proposed Methods of Concealment

在前面的小節中我們討論了之前的損失畫面修補技術，利用動作向量的前向投射，修復損失畫面的動作向量，在這種做法中，被修復的畫面屬於被動的角色，由於投射的向量不會正好落在被修復畫面的方塊中，必須額外處理。另一方面由於不同的 FMV 有可能投射到損失畫面重疊的地方，或是有的損失畫面有部分可能沒有被投射到。這些情況都會造成修補上的障礙，尤其當原始的 MV 與真實運動軌跡不符合時更容易影響修補結果。

為了改善 MV 與真實運動軌跡不符合的現象，我們提出了利用模糊邏輯的動作向量修復技術，提升修補效果。另一方面為了改善前向動作投射遇到的被動性的問題，我們提出了一種新的主動性修復技術，採用後向的動作投射，可以用較簡單的方式達到畫面的修補。

3.1 Motion Vector Correction

由於一般我們編碼時所測的 MV 是找參考畫面中最接近方塊與目標方塊的位移量，有時候這樣的 MV 並不符合真正的運動軌跡向量，這樣的情況下做動作投射就會容易造成錯誤的修補，如 Fig. C.3.1 所示。為了讓 MV 能夠接近真實的運動軌跡，我們提出了一種利用模糊邏輯判斷向量可靠度，藉此修正錯誤向量的方式。



(a) Motion projection with correct MV

(b) Motion projection with incorrect MV

Fig. C.3.1 Motion projection with: (a) correct MV; (b) incorrect MV.

3.1.1 Proposed Fuzzy MV Correction

一般情況下，物體的運動有著時間上與空間上的連續性，舉例來說：一台移動的車子，由於畫面間隔時間短暫，在這短暫時間前後移動的速度應該是接近的，這指的是時間上的連續性；而車頭與車尾的運動應該也是接近的，這指的是空間上的連續性。由於有這樣的特性，當畫面中有某個 MV 時間前後不連續，與周圍動作也不連續

的情況，此 MV 很有可能就是錯誤的動作。

我們所提出的動作向量可靠度判斷方式如 Fig. C.3.2 所示，其中 $R_{i,j}^{t}$ 表示在 t 時間、位置(i, j)的 $MV_{i,j}^{t}$ 之可靠度，$TD_{i,j}^{t}$ 表示時間上的不連續性，$SD_{i,j}^{t}$ 表示空間上的不連續性，如(3.3)、(3.4)所示，當 $TD_{i,j}^{t}$ 或 $SD_{i,j}^{t}$ 越大的時候，可信度就越低，當 $TD_{i,j}^{t}$ 和 $SD_{i,j}^{t}$ 都小的時候可信度高。

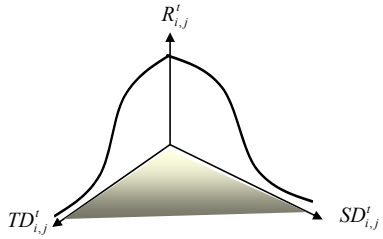


Fig. C.3.2 Temporal-Spatial fuzzy reliability.

$$TD_{i,j}^{t} = \left\| 2 \cdot MV_{i,j}^{t} - MV_{iB,jB}^{t-1} - MV_{iF,jF}^{t+1} \right\|$$

$$\text{where} \begin{cases} [iB, jB] = [i, j] + MV_{i,j}^{t} \\ [iF, jF] = [i, j] - MV_{i,j}^{t} \end{cases}$$

(3.1)

$$SD_{i,j}^{t} = \left\| MV_{i,j}^{t} - \overline{MV_{i,j}^{t}} \right\|,$$

$$\text{if } \left\| MV_{i-1,j}^{t} - MV_{i-1,j}^{t} \right\| \le \left\| MV_{i,j-1}^{t} - MV_{i,j+1}^{t} \right\|$$

$$\overline{MV_{i,j}^{t}} = \frac{\left(MV_{i-1,j}^{t} + MV_{i+1,j}^{t} \right)}{2},$$

$$\text{else } \overline{MV_{i,j}^{t}} = \frac{\left(MV_{i,j-1}^{t} + MV_{i,j+1}^{t} \right)}{2}.$$

(3.2)

我們將時間域與空間域可信度的 membership function 定義如下：

$$TR_{i,j}^{t} = \frac{1}{1 + \exp\left(\alpha \cdot (TD_{i,j}^{t} - TD_{Th}) \right)}$$

(3.3)

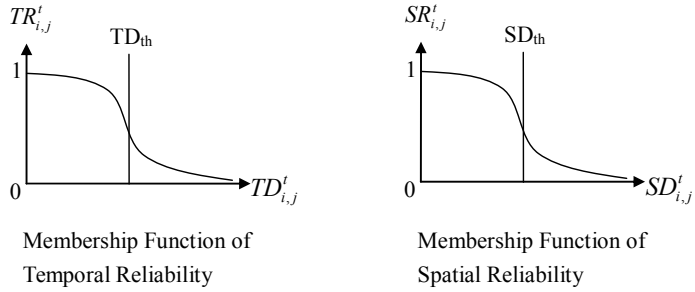
$$SR_{i,j}^{t} = \frac{1}{1 + \exp\left(\beta \cdot (SD_{i,j}^{t} - SD_{Th}) \right)}$$

(3.4)

其中 $TR_{i,j}^{t}$ 表示時間域的可信度而 $SR_{i,j}^{t}$ 表示空間域的可信度，藉由可信度我們可以進行動作向量的修正將可信度較低的動作向量用附近可信度較高的向量取代，如下所示，其中 N 代表(i,j)鄰近的點，$MVC_{i,j}^{t}$ 代表修正後的 MV。

$$MVC_{i,j}^{t} = \sum_{i,j \in N} R_{i',j'}^{t} \cdot MV_{i',j'}^{t}$$

$$R_{i',j'}^{t} = \frac{SR_{i',j'}^{t} \cdot TR_{i',j'}^{t}}{\sum_{i,j \in N} SR_{i,j}^{t} \cdot TR_{i,j}^{t}} \quad , \quad (3.5)$$



Membership Function of Temporal Reliability

Membership Function of Spatial Reliability

Fig. C.3.3 Membership function of temporal reliability and spatial reliability.

3.2 Backward Motion Projection

到目前為止我們介紹了許多編碼端錯誤修補的技術，這些方法都希望能夠將損失的部分修補回來。回到錯誤修補的最初目的，我們知道我們之所以要做錯誤修補其實並不是為了損失的那張畫面資訊，真正最大的目的是為了要降低錯誤傳遞的影響，因為後面的畫面都要參考前面的畫面進行解碼，當參考畫面出錯會造成錯誤一直傳遞下去，為了這個原因我們才需要做錯誤修補。

這時候若我們跳出修補損失部分的想法，有沒有可能讓後面的解碼略過損失的部分，不參考錯誤的位置進行解碼的動作？如 Fig. C.3.4 所示。

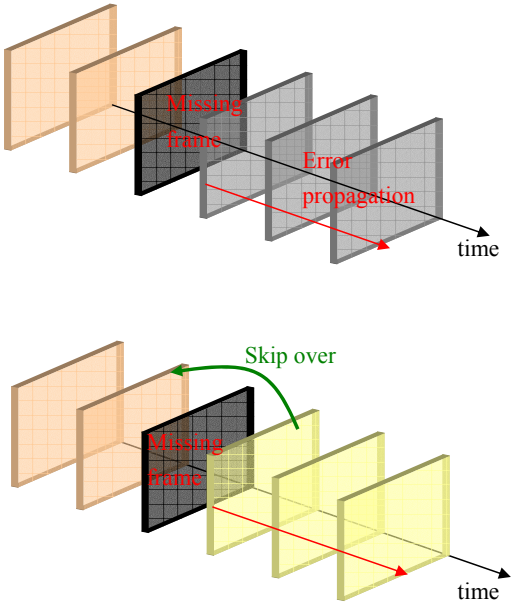


Fig. C.3.4 Missing frame skipped decoder.

基於這樣的想法，我們發展出來新的錯誤修補技術，可以略過錯誤的畫面採用更前面的畫面作為參考畫面，和前面提到的演算法概念最大差異是我們採用的動作修補方式是用後向的動作投射，如 Fig. C.3.5 所示。理論上前向的投射與後向的投射原理都相同，都是利用 OF 時間上的連續性，所以做出來的效果理論上是差不多的，但是在運算上後向投射的做法會比前向投射的做法容易實行。

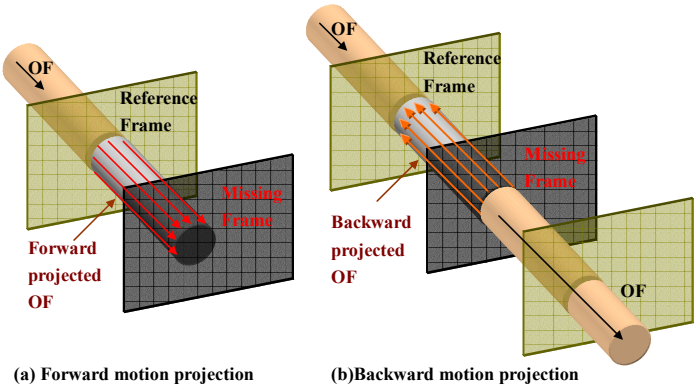


Fig. C.3.5 (a) Forward (b) Backward motion projection.

前向投射的做法中，修復的向量場使屬於「被動」的角色，我們必須從前參考畫面的動作向量做投射，在投射到的位置紀錄下來動作向量，而修復的的向量場在投射過程有可能有些區域會發生沒有被投射到或是重複投射的情況，這些情況都需要額外的處理[26]。

而若採用後向的投射時，我們是在損毀畫面的後一張畫面，使用這張畫面自己的動作向量往參考畫面做延伸，這樣的做法是從修復位置為起點，屬為「主動」的做法，所以可以避免前面所說的被動做法遇到的問題。

我們所提出的畫面修復技術如 Fig. C.3.6 所示，和以往最大差異是我們略過了真正損失的畫面，而是修正之後會使用損失畫面的後續畫面，例如 Fig. C.3.6 原本 F_{n+1} 必須使用 F_n 作為參考畫面，但是 F_n 是損失的畫面，所以我們就直接把原本指向 F_n 的動作向量做線性延伸，向量兩倍延伸至 F_{n-1}。

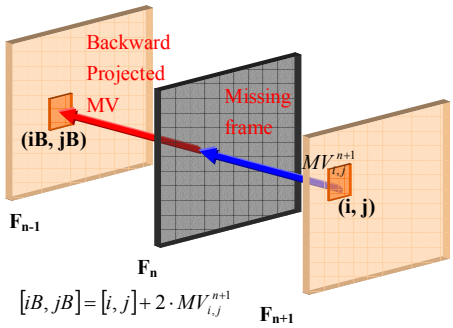


Fig. C.3.6 Missing frame recovery with backward motion projection.

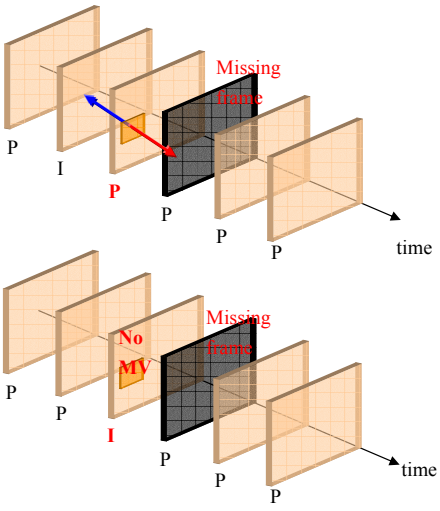


Fig. C.3.7 Forward MV projection with intra frame

這樣的做法最大好處在於運算簡單，只需將後續動作向量做線性的延伸即可，也不需額外對於 intra frame 做考慮。比較前向動作投射方式，如 Fig. C.3.7 所示，由於是使用損失畫面之前的動作向量，當在損失畫面之前的是 inter frame 時，我們可以使用它的動作向量，但是當前一張畫面是 intra 時，我們就會需要另外找其他向量來做投射，這樣就需要額外的處理方式。相對若我們使用的是後向動作投射時，我們是利用受損畫面的後一張畫面進行動作向量投射，當它屬於 inter frame 時，很自然就使用他的動作向量進行；而若當他屬於 intra 時，雖然沒有動作向量，但是 intra 本來就不需要參考前面的受損畫面，可以直接正確解出畫面，這樣是屬於最好的情況。所以當我們採用後向的動作投射作為修補方式時，無論投射畫面是 intra 或是 inter 都可以正確的進行，不需額外的處理。

(4) Experimental Results

我們將經過 H.264 壓縮過的視訊檔案通過 2.1.1 的傳輸模擬系統，設定不同的封包損失機率，觀察使用我們所提出的演算法補償效果，Fig. C.6.1 是 foreman sequence 經過封包損失率(PLR)為 1%的通道之後，沒有錯誤修補(without EC)、前向動作投射(FMP)、以及我們所提出的後向動作投射(BMP)的 PSNR 比較圖，由於我們的方法加入了動作向量修正，因此會有比較好的修補效果。另外由於使用後向動作投射屬於主動性的修正，在影像邊緣部分也會有比較好的效果，如 Fig. C.6.2 所示。另外我們也討論不同視訊在不同 PLR 下，修補的 PSNR 比較，如 TABLE C.I 所示。

TABLE C.III Average PSNR for the considered sequence at different packet loss rates (PLRs).

Sequence	PLR	No EC	FMP	BMP
Foreman	1%	32.43	35.26	35.50
	2%	29.75	34.51	34.94
	5%	24.66	33.52	34.31
Flower	1%	32.66	34.14	34.52
	2%	30.45	33.47	34.18
	5%	20.65	30.29	32.27
Stefan	1%	31.27	33.43	33.48
	2%	26.33	31.91	32.09
	5%	18.43	29.20	29.51

(5) Conclusion and Future Works

在本文中我們提出了一種新的錯誤修補方法，利用後向的動作投射，讓解碼視訊在惡劣的傳輸狀況下，即使發生整張畫面都損失的情況，仍能有效修補，在作法上也較之前的技術簡單。

為了提升修補的效果，我們也提出了一種新的動作向量修正技術，藉由動作向量時間域與空間域的連續性，經過模糊邏輯判斷動作向量之可靠度之後進行修正，能夠讓錯誤修補時使用的動作向量更接近真實運動軌跡，讓動作投射更加準確。

再之後的研究中，我們希望能夠將動作向量修正的技術移至編碼端，這樣能夠讓解碼端的錯誤修補更為簡潔快速，也可以增進錯誤修補的效果。除此之外，在編碼端進行動作向量修正將有助於更多需要正確動作向量的影像處理技術，例如時間域畫面內插或是畫面解析度增進技術。應用動作向量修正技術可以改善編碼動作向量與真實動作不符時所產生的瑕疵問題。

(6) Reference

[1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra, “Overview of the H.264/AVC Video Coding Standard,” *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, No. 7, pp.560 – 576, July 2003

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

[3] Thomas Stockhammer, Miska M. Hannuksela, and Thomas Wiegand, “H.264/AVC in Wireless Environment,” *IEEE Trans. On Circuits Syst. Video Technol.*, vol. 13, No. 7, July 2003

[4] Stephan Wenger, “H.264/AVC Over IP,” *IEEE Trans. On Circuits Syst. Video Technol.*, vol. 13, No. 7, July 2003

[5] Thomas Stockhammer, Thomas Wiegand, Tobias Oelbaum, and Florian Obermeier, “Video coding and transport layer techniques for H.264/AVC-based transmission over packet-lossy networks,” *Proc. ICIP 2003*, vol. 3, September 2003.

[6] J. Klaue, B. Rathke, and A. Wolisz, "EvalVid – A Framework for Video Transmission and Quality Evaluation", In Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Urbana, Illinois, USA, September 2003.

[7] Chih-Heng Ke, Cheng-Han Lin, Ce-Kuen Shieh, Wen-Shyang Hwang, “A Novel Realistic Simulation Tool for Video Transmission over Wireless Network”, The *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006)*, June 5-7, 2006

[8] Network Simulator, http://www.isi.edu/nsnam/ns/

[9] Zhang Rongfu, Zhou Yuanhua, and Huang Xiaodong, “Content-adaptive spatial error concealment for video communication,” *IEEE Trans. On Consumer Electronics,* vol. 50, Feb. 2004.

[10] Olivia Nemethova, Ameen Al-Moghrabi and Markus Rupp, “Flexible Error Concealment for H.264 Based on Directional Interpolation,” *International Conference on Wireless Networks, Communications and Mobile Computin,* vol. 2, June 2005.

[11] Steven Beesley, Andrew Armstrong, Christos Grecos, “An Edge Preserving Spatial Error Concealment Technique for the H.264 Video Coding Standard,” *Research in Microelectronics and Electronics 2006, Ph. D.*, June 2006.

[12] W. M. Lam, A. R. Reibman, and B. Liu, “Recovery of lost orerroneously received motion vectors,” *in Proc. ICASSP’93*, pp. V417-V420, Apr. 1993.

[13] E. T. Kim, S.-J. Choi, and H.-M. Kim, “Weighted boundary matching algorithm for error concealment in the MPEG-2 video bit stream,” *Signal Process.*, vol. 73, pp. 291–295, Mar. 1999.

[14] Yan Chen, Au, O., Chiwang Ho, and Jiantao Zhou, “Spatio-temporal boundary matching algorithm for temporal error concealment,” *Proc. ISCAS 2006,* May 2006.

[15] Agrafiotis, D., Bull, D.R., Canagarajah, C.N., “Enhanced Error Concealment With Mode Selection,” *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 16, August 2006.

[16] M. E. Al-Mualla, C. N. Canagarajah, and D. R. Bull, “Error concealment using motion field interpolation,” *Proc. ICIP 1998,* Vol. 2, pp. 512—516, October 1998.

[17] Jinghong Zheng, Lap-Pui Chau, “A temporal error concealment algorithm for H.264 using Lagrange interpolation,” *Proc. ISCAS 2004,* vol. 2, May 2004.

[18] Jae-Young Pyun, Jun-Suk Lee, Jin-Woo

Jeong, Jae-Hwan Jeong, and Sung-Jea Ko, "Robust error concealment for visual communications in burst-packet-loss networks," *IEEE Trans. On Consumer Electronics,* vol. 49, November 2003

[19] S. Gnavi, M. Grangetto, E. Magli, and G. Olmo, "Comparison of rate allocation strategies for H.264 video transmission over wireless lossy correlated networks," in *Proc. ICME\—IEEE Int. Conf. Multimedia and Expo*, 2003.

[20] E. N. Gilbert, "Capacity of a burst-noise channel," Bell Syst. Tech. J., vol. 39, pp. 1253-1265, Sept. 1960.

[21] E. 0. Elliott, "Estimates of error rates for codes on burst-noise channels," Bell Syst. Tech. J., vol. 42, pp. 1977-1997, Sept. 1963.

[22] S. Belfiore, M. Grangetto, E. Magli, G. Olmo, "Concealment of whole-frame loss for wireless low bit-rate video based on multiframe optical flow estimation," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 316-329, Apr. 2005.

[23] S. Belfiore, M. Grangetto, E. Magli, G. Olmo, "An error concealment algorithm for streaming video," *Proc. ICIP 2003*, 2003.

[24] Baccichet, P.;and Chimienti, A., **"**A low complexity concealment algorithm for the whole-frame loss in H.264/AVC,**"** *IEEE 6th Workshop on Multimedia Signal Processing*, October 2004.

[25] P. Baccicht, D. Bagni, A. Chimienti, L.Pezzoni, and F. Rovati, "Frame concealment for H.264/AVC decoders," *IEEE Trans. Consumer Electronics,* vol. 51, no. 1, pp. 227-233, Feb. 2005.

[26] Zhenyu Wu and J. M. Boyce, "An error concealment scheme for entire frame losses based on H.264/AVC," *Proc. ISCAS 2006,* May 2006.

[27] A. M. Tekalp, Digital Video Processing. Englewood Cliffs, NJ: Prentice- Hall, 1995.

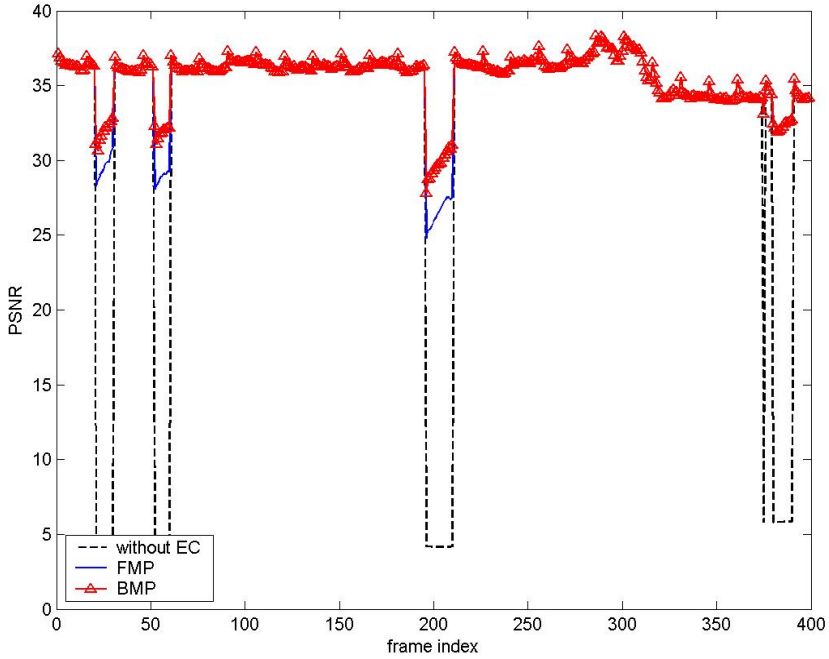


Fig. C.6.1 Performance comparison on the Foreman sequence (packet loss rate 1%).

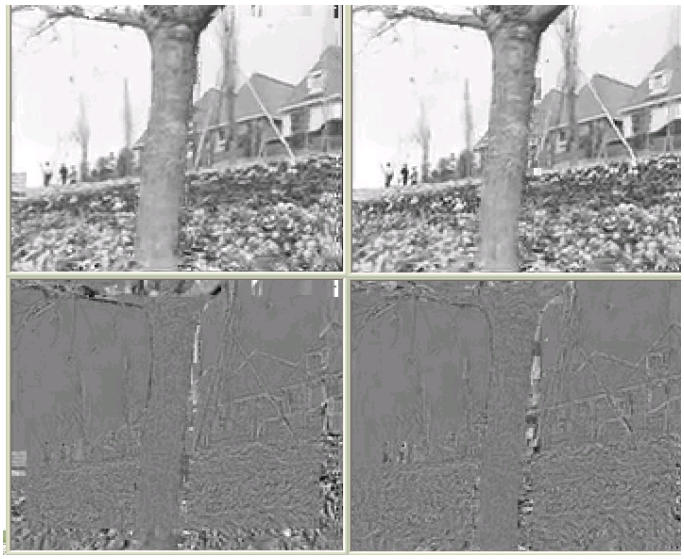


Fig. C.6.2 Example of concealed frame (flower sequence, PLR 2%);
(top left) concealed frame by FMP;
(top right) concealed frame by BMP;
(bottom left) residual by FMP;
(bottom right) residual by BMP.

成果自評

本次計畫中，我們達成了三項成果。第一，我們成功的在 TI DSP 實驗版上，根據 H264/AVC 的影像壓縮規範，實現了一個即時影像傳輸系統。此系統包含影像接收-壓縮-網路傳送端，以及網路接收-解壓縮-播放端。當一端送出經過 H.264/AVC 編碼技術壓縮過的資料，經過網路傳輸後，可以被另一端收到並進行解碼。此外，我們也採用了多線程緒以及平行化的方式，來達到即時執行的目的。而本專案所撰寫之程式碼，日後除了可供學界研究外，也可供業界日後的參考與使用。

此外在本計畫中，我們也針對 H264/AVC 原始的碼率控制做一改善。我們首先分析量化參數、移動補償資料與壓縮後資料量之間的關係，進而針對壓縮後的檔頭資料作分析，然後重新建立一個針對 H.264 /AVC 編碼特性的碼率失真模型。對於每張畫面的位元配置，我們也利用前後張影像的資料關係去調整，以改善影像品質跟穩定度。最後，為了改良原本使用在 JM 上的 MAD 值預測，我們利用移動向量來預測每個巨區塊的 MAD 值。整合上述的方法，能改善原本在低碼率時在緩衝器上的不佳效果，特別是藉由較精準的碼率失真模型能預測出準確的量化參數。經過實驗，可以發現在緩衝器的穩定性上能有不錯的效果，在影像品質上更能獲得明顯的效果。

最後在本次計畫中，我們也完成在解碼端一種新的錯誤修補方法，利用後向的動作投射，讓解碼視訊即使發生整張畫面都損失的情況，仍能有效修補，在作法上也較先前的技術簡單許多。在解碼端，我們也提出了一種新的動作向量修正技術，藉由動作向量時間域與空間域的連續性，經過模糊邏輯判斷動作向量之可靠度之後進行修正，能夠讓錯誤修補時使用的動作向量更接近真實運動軌跡，讓動作投射更加準確。以上兩項研究成果除了具有學術上之創新價值外，事實上也具有工業界的實用性。