國 立 交 通 大 學

電信工程學系
碩 士 論 文

Error Concealment in Compressed Domain

壓縮領域的錯誤隱藏演算法

研究生：邱培哲
指導教授：張文鐘 博士

中 華 民 國 九十五 年 八 月

壓縮領域中的錯誤隱藏演算法

Error Concealment in Compressed Domain

研 究 生：邱培哲　　Student：Wei-Tung Chang

指導教授：張文鐘　　Advisor：Wen-Thong Chang

國 立 交 通 大 學

電信工程學系

碩 士 論 文

A Thesis

Submitted to Department of Communication Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Communication Engineering

August 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年八月

壓縮領域中的錯誤隱藏演算法

研究生：邱培哲　　指導教授：張文鐘　博士

國立交通大學電信工程學系碩士班

摘要

隨著無線網路的普及，越來越多應用服務在無線網路的環境下而開發。如大家所知，無線網路中，資料傳輸失敗或是傳輸延遲時常發生，應用服務必須採取補救措施。

在無線網路上進行多媒體串流服務是個困難的工作。影片資料首先使用 MPEG 技術編碼，然後切割成多個封包進行傳輸。在無線網路的環境下，封包極易丟失，而封包內容也可能因爲自然環境的干擾而產生 bitwise error，這兩項因素會在客戶端造成影像內容解碼錯誤和品質下降。更糟的是，一張畫面的解碼失敗也會造成接下來連續數張畫面品質下降。

爲了避免在解碼失敗後，接續的畫面品質下降，本篇論文提出了 compressed domain 的錯誤隱藏技術。根基於本實驗室建立的多媒體串流系統，客戶端可以一面解碼，一面把正確解碼後得到的 motion vector 側錄。當客戶端遇到解碼失敗時，側錄的 motion vector 便可以輔助我們進行錯誤隱藏，重建參考畫面。

就目前已知的錯誤隱藏演算法而言，絕大多數是在 pixel domain

操作，而我所提出的演算法是操作在 compressed domain，兩者的運算量相比差距十分大。本論文所提及的目標和功能都已達到，並且在多媒體串流系統上實現，可以即時進行運算。

Error Concealment in Compressed Domain

Student : Pei-Che Chiu Advisor : Dr. Wen-Thong Chang

Department of Communication Engineering
National Chiao Tung University

Abstract

With the wide spread of WLAN, more and more applications are developed under WLAN environment. As everybody knows, the channel noise in WLAN tends to cause transmission problem, such as data transmission failure or packet transmission delay. Applications should take care with compromising procedures

Providing multimedia streaming service over WLAN is a tough task. Video clips are first coded with MPEG-like techniques, then packed into several packets, and transmitted to the client end at last. In WLAN environment, packet loss happens all the time. In addition, packet bitstream might be changed, too. These 2 facts imply video clip decoding failure at the decoder. What's worse is the succeeding video clip are of quality degradation owing to they reference to wrong frame.

In order to reduce the effect brought by decoding failure, compressed domain error concealment algorithm is proposed in this thesis. Cooperating with the already-built multimedia streaming system, the client end can decode video clips and record meaning statistics at the same time. When meeting decoding failure, these statistics help us build an artificial frame as the reference for all succeeding video frames.

Most error concealment algorithms so far are designed in pixel domain and take much larger computation power than what I propose. The goals and functions mentioned in the thesis are achieved and available for multimedia streaming system client end.

誌謝

能夠順利完成這篇論文，最要感謝的人是我的指導教授 張文鐘博士。老師在我二年的碩士生涯中，給予了學業上和生活上的各種指導，處處能感受到老師的用心，讓我在遇到挫折和瓶頸時，有毅力也有智慧向前繼續前進。

另外要感謝的，是實驗室裡的各個好夥伴們。Master 林，Doctor 陳和 Leader 張，他們隨時都能給我友情支援，一起打球一起歡笑的黃金歲月，一起在實驗室唸書寫程式寫報告的日子，感謝他們爲我帶來的美好回憶。

最後，我要感謝我的家人，父母親和兄長的支持總讓我能心無旁騖地從事研究工作。

感謝所有幫助過我、陪伴我走過這一段時光的師長、同儕和家人。

謝謝！

誌於 2005.08 風城 交大

Peter

目錄

List of Figures

Chapter 2

Chapter 3

Chapter 4

Chapter 5

List of Tables

Chapter 2

Chapter 4

第一章 介紹

1.1背景說明

隨著網路技術的發展，各種網路的軟/硬體設備逐漸成熟與售價降低，每個人皆有機會與能力連線到網際網路。隨之而生的服務，尤其是多媒體領域的服務，越發受到關注。

多媒體串流技術是目前網際網路上最爲流行的多媒體服務技術。想像一下，當你在咖啡廳裡點選了一部短片，但是要等到下載完畢才能觀賞，這是一件多惱人的事情？多媒體串流技術提供了人們一個解決方案。人們能夠一邊下載檔案，同時間也能觀賞已經下載的部份。串流技術提供了人們多種可能性，它節省了人們等待的時間。它能提供 VOD(Video-on-Demand)的服務，人們不需要走到街口的光碟出租店即可觀賞想看的影片。串流技術也有能力提供即時視訊的服務。這項服務可以延伸至視訊聊天，視訊會議，遠距教學和遠距醫療。以前無法想像或是無力提供的影像電話，能夠藉由多媒體串流技術得而實現。在資源受限的環境裡，使用者可以藉由網路設備和串流技術的搭配，實現自己的目的。

現實生活中，某些地點因爲場所本身的限制和架射網路設備的種種考量，鋪設有線網路並不是最經濟的作法，便利的無線網路設備才

是較爲便宜的選擇。像是在網際網路深入各個家庭的同時，能夠輕鬆架設的無線網路也會是優先選擇。新設的辦公室也會爲了避免整線的麻煩而採用無線網路。或者，在某些場所在考量了室內設計後，隱形的無線網路設備能夠維持設計風格的一致感。無線網路的便利性讓人能隨時隨地上網。你可以在咖啡廳一面享受下午茶，同時間瀏灠購物網頁。你也可以在餐館一邊點菜，一邊了解今天的股價。但與便利性相對的則是資料傳輸的穩定性。

爲了能在這些場所中使用相同的多媒體串流服務，與經由有線網路設備有著一致的品質，克服無線網路的不穩定性成了多媒體串流的最優先課題。過於複雜的環境會導致信號的衰弱，封包的遺失，或是封包無法即時到達目的地。信號的衰弱意即信號的強度減弱，這意謂每個位元都有機會發生錯誤(bitwise error)，進一步看，封包的內容將不再令人信賴。封包的遺失即代表內容的遺失，在擁擠的無線網路環境中，這是極有可能發生的事情，如果應用服務並沒有採用封包重送機制，那麼遺失的封包有可能導致系統的不穩定，資料的不完整性或是通訊雙方的互動失敗。然而，如果應用服務使用了封包重送機制，遺失的封包會被要求再送一次，這會讓擁擠的網路更加擁擠，產生惡性循環。當應用服務很注重「即時性」的品質而封包無法即時到達目的地時，應用服務將無法根據即時的資料提供可信賴的服務。無法提

供可信賴的服務相信是各項應用服務的設計者及使用者所無法忍受的事情。

由於目前無線網路技術的限制，無線網路的頻寬略嫌有限，無線網路設備處理封包的速度較慢，各種應用服務皆以傳送最少位元量或是最少數量的封包爲通訊原則。爲了減低佔用的頻寬，多媒體串流技多採用 MPEG-4 的編碼。目前最爲流行的 MPEG-4 編碼技術，和 MPEG-1 / MPEG-2 編碼技術相比，MPEG-4 更能夠有效地壓縮影像。這代表了傳遞的資料量能有效降低，所需的頻寬也將能減少。對多媒體串流技術而言，MPEG-4 是最好的選擇。但由於 MPEG-4 所使用的編碼技術內含 VLC (Variable-Length Coding)技術，VLC 的特性是每個符號(symbol)會由不同長度的位元流(bit-stream)代表，其中一個位元的錯誤會導致整串位元流(bit-stream)的解碼失敗。根據網路的運作原理，進行傳輸前，必須先將將這個位元流切成數個封包。考慮到底層實體網路採用無線網路設備，我們難以確保使用者端能及時和完整地接收全部的封包。在偶有遺失封包的狀況下，使用者端以完整的位元流(bit-stream)爲解碼的輸入項目，使用者端將以圖片解碼失敗告終。又，MPEG-4 另外一項特性是將時間連續的畫面相似性隱藏於之前解碼成功的畫面。這意謂著一張畫面的解碼錯誤極有可能導致之後數張畫面的解碼錯誤(error propagation)。

在無線網路進行多媒體串流技術是一項困難的任務。無線網路的特性極易造成串流系統的失敗。面對位元錯誤(bitwise error)，封包遺失(packet loss)和封包延遲到達(delayed packet)三種問題，串流系統有必要針對這三個問題做出措施。錯誤隱藏(Error Concealment)因此被提出來。

錯誤隱藏相當於在使用者端進行解碼失敗後的 post-processing，錯誤隱藏能修補破因爲解碼錯誤而受損的畫面。錯誤隱藏通常只在使用者端進行。設計良好的錯誤隱藏演算法可利用已知的資訊進行畫面修補，無需伺服器端重送封包，降低無線網路的負擔。重建的畫面可以交給 MPEG-4 解碼器利用，避免 error propagation 的產生。

1.2動機

- 目前市場上的多媒體串流系統並不支援錯誤隱藏的功能。發生錯誤時往往造成物體的扭曲，無意義的畫面。
- 根據已經解碼成功的資訊進行影像處理，能避免 error propagation(錯誤擴散)的產生。
- 除了阻止 error propagation 的產生，同時也避免了解碼器浪費資源處理錯誤的數據。

1.3目標

研究的目標有以下幾點

- 關於 motion vector 的討論與利用。
- 關於錯誤隱藏的演算法設計與實作。
- 關於演算法細節的探討。
- 關於執行結果的探討。

1.4論文大綱

本篇論文的架構如下。第二章將會介紹及討論與我的演算法有關的各項協定。在第三章我們迅速地瀏覽各項錯誤隱藏的技術。第章詳細討論演算法的實作細節。第五章是實驗結果。最後，第六章將爲本篇論文作出結論。

第二章 多媒體串流系統概論

在本章裡，我將瀏覽各項多媒體串流系統所使用的技術，包含 video coding 和 data transmission 所使用的通訊協定[1][2]。

2.1 MPEG-4 影像編碼概論

2.1.1 影像編碼技術 coding

Color Space: from RGB to YUV

對於彩色影像，每一個彩色像素的資訊至少需要三個種類的色彩才能呈現。決定亮度及色彩的搭配表示方式，這稱作 color space。

Color space 有許多種表示方式：

RGB: 使用 Red, Green, Blue 紅綠藍三原色組成 color space，一個像素將由 RGB 三原色構成。RGB 的數值表示紅色，綠色和藍色所組成的比例。藉由紅綠藍的任意比例組合，RGB 可以呈現相素的各種顏色。

YCbCr: 對於人類視覺系統(Human Vision System, HVS)而言，人眼對於顏色的亮度比較敏感，彩度則否。在 YCbCr 中，Y 是亮度(luminance)，Cb 和 Cr 是彩度(chrominance)。YCbCr 和它的變型

YUV 來自於 RGB color space 的換算(conversion)。

- $Y = k_r*R + k_g*G + k_b*B$

k 是 weighting wactor。Cb, Cr, Cg 可以藉由 RGB 和 Y 的差值推算。

- Cb = B – Y
- Cr = R – Y
- Cg = G – Y

在這裡我們使用了四個變數(Y, Cb, Cr, Cg)儲存(R, G, B)三個變數，Cb + Cr + Cg 是一個定值，因此在實作上我們只使用(Y, Cb, Cr)三個變數。根據人類視覺系統，人眼對顏色的亮度比色彩敏感，這暗示著 Cb 和 Cr 的資訊較不重要。在一張彩色畫面中，我們可以捨棄保留全部的 Y 值，捨棄大部份的 Cb, Cr 值，而仍然能呈現原來的畫面。

在 MPEG-4 Video Coding 中，最原始的圖案來源使用 RGB 格式，之後將轉換至 YCbCr 格式進行下一步壓縮，其中的 Cb 和 Cr 使用的解析度低於 Y。這有效地降低了資料量，但對於人眼而言，觀看這種 YCbCr 格式的圖片和 RGB 格式的圖片，並不會察覺明顯的差異處。

4:2:0 是目前的 YCbCr 取點規則，4 個 RGB 格式像素轉成 YCbCr 格式時，取得 4 個相素的 Y 值，但只取得 1 個像素的 Cb 和 1 個像素的 Cr 值。這種取點規則在人眼視覺系統上並不會產生明顯

差異，然而在佔用記憶體和硬碟空間上，少了一半的空間，是變相的影像壓縮。

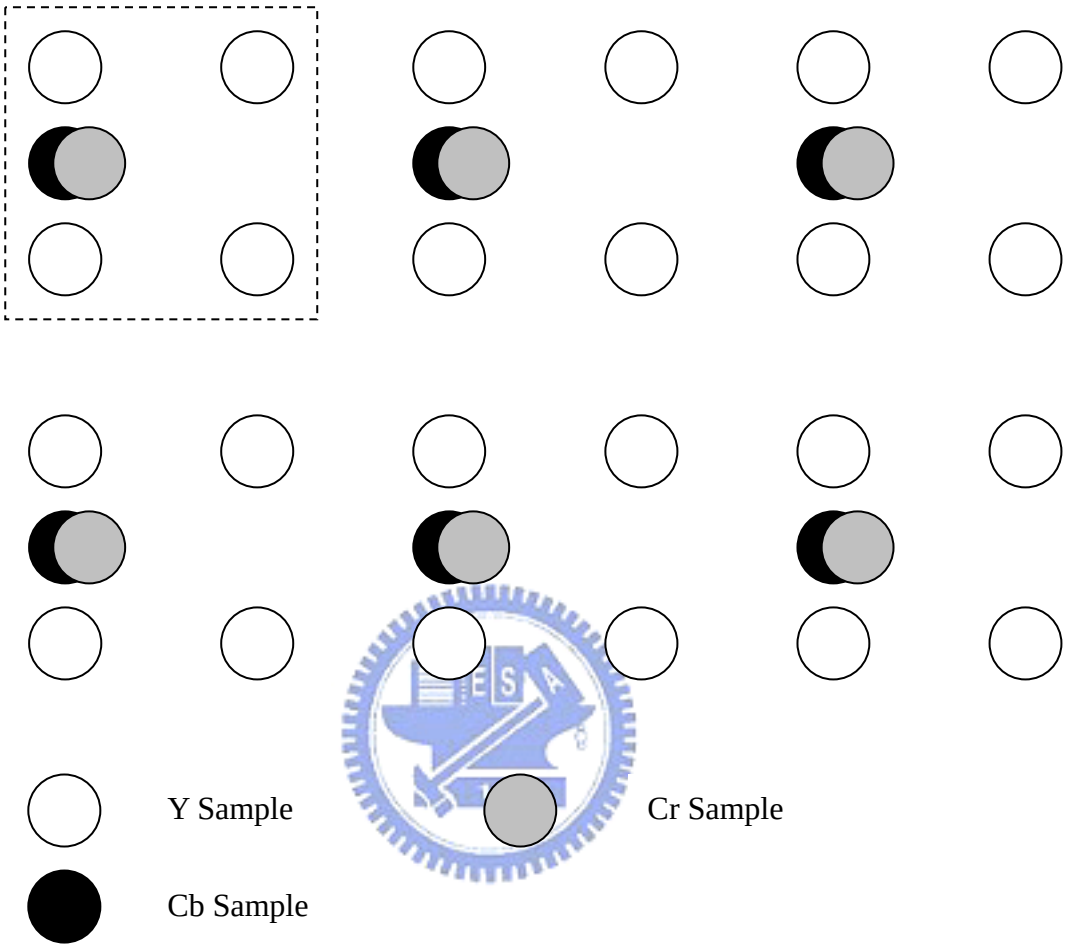

Figure 2.1 – 4:2:0 sampling pattern (progressive)

Motion Estimation / Motion Compensation

在 MPEG-4 中，我們使用 Temporal Model 消除連續畫面中的多餘性(temporal redundancy)。由於連續畫面中，相似之處很多，像是在 t 時刻出現的物體，在 t+1 時刻仍然出現在同一個位置(或是鄰近的位置)，爲了要儘可能地節省(縮減)代表圖片用的位元，我們建立 temporal model。在 temporal model 中，根據 reference

frame，我們預測 current frame 的各個像素，建立 predicted frame。因爲預測無法達到百分之百精確，reference frame – current frame 剩下 residual。Residual 代表 current frame 和 predicted frame 的不同之處。在 MPEG-4 的 Simple Profile 中，定義了 I-frame 和 P-frame。P-frame 大量使用了 temporal model。

在 MPEG-4 中，temporal model 的輸入單位是 Macroblock (MB)，我們將一張畫面(frame)切割成許多 16x16 的 Macroblock，Macroblock 再細分成四個 8x8 的 block。在 temporal model 中消減連續畫面間的多餘性的處理單位是 block。基本的概念如下。

連續畫面中的物體會隨著時間移動(或靜止)，它在 reference frame 和 current frame 間的位移可以用 motion vector (MV)表示。爲了決定 motion vector，我們將 current frame 中的每個 Macroblock(或是 block)和 reference frame 中的像素進行比對(物理的意義即是找出相似的物體)。這個動作稱作 block-based motion estimation。由於運算量的考量，motion estimation 設定了 search range (SR)，避免過多而無謂的比對。當 search range 內都比對過了，擁有最高相似度的 motion vector 就是 Video codec 中使用的 motion vector。若以 Macroblock 進行比對無法得到令人滿意的相似性比對結果，那

麼 Macroblock 會被拆成四個 block，進行各別的比對，得到各別的 motion vector。每一個 Macroblock 都會有一個或四個 motion vector。是否將 Macroblock 拆成四個 block 進行 motion estimation 依各個 codec 製作者的意願而定，這並不是強制的要求。

得到 motion vector 後，reference frame 減去 current frame，剩下 residual，這個動作稱作 motion compensation。Residual 是兩個對應的 Macroblock(或是 block)中不相似之處。因此，在 temporal model 中，predicted frame 是由 motion vector 和 residual 合作還原原本的畫面。Motion estimation/motion compensation 使用的 block size 將會決定 residual 的多寡。如果使用 4x4 的 block，那麼可以得到很精確的 predicted frame，和 current frame 相減後幾乎不會剩下 residual，但是要得到 predicted frame，必須耗費大量的運算資源進行 motion estimation。

Motion estimation 失敗時，或是相似性低於某個門檻時，Video codec 並不會產生 motion vector。因爲相似性過低的兩個 Macroblock(或是 block)產生的 residual 太多，考量到之後還要進行的壓縮，residual 無法明顯地縮減使用的位元(甚或是多於原來不作 motion estimation/motion compensation 所使用的位元)，Video

codec 會捨棄 temporal model，進行下一步壓縮。

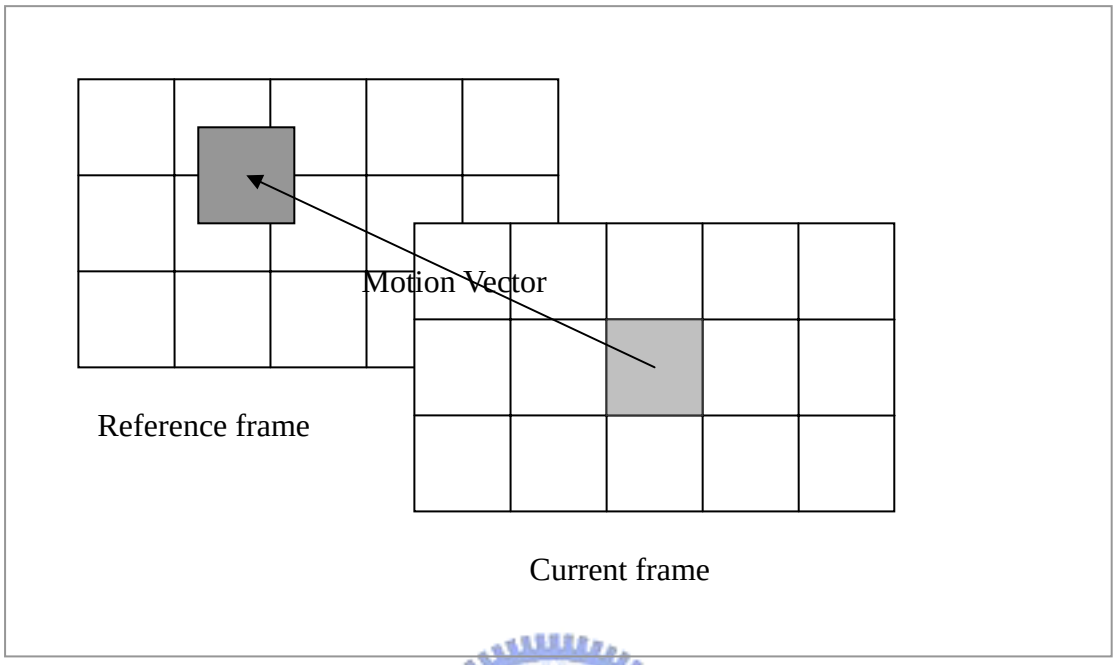

Figure 2.2 – Motion Estimation

DCT (Discrete Cosine Transform)

接著要進行 transform coding，將 pixel domain 的值用 transform domain 的值代表，transform coding 並無法降低像素所使用的位元數，但它能將無關緊要的資訊分離出來，留待後面的 VLC 和 RLE 壓縮法處理(而這一步通常能進一步降低使用的位元數)。最流行的 transform coding 有兩種類別：block-based 和 image-based。DWT(Discrete Wavelet Transform)屬於 image-based，對於記憶體的要求較高(因爲是以整張 frame 爲對象進行處理)，對於使用 block-based motion estimation / motion compensation 的 MPEG-4 並不適合。DCT(Discrete Cosine Transform)是目前 block-based

transform 中最熱門的選擇。

Reordering

在 Quantization 之後的 quantized transform coefficients，必須儘可能地壓縮，以供進一步的檔案儲存或是傳輸之用。Transform coefficients 在各個頻率上皆有不同強度係數，在 Quantization 之後，大部份頻率的強度係數被量化爲 0。如果能對於這些強度爲 0 的係數進行有效的排列，在這之後的壓縮可以進行的更有效率。由於圖片本身的特性，許多圖片在進行 DCT 之後在高頻項目的強度爲 0，在 quantization 之後亦爲 0，因此 Reordering 的辦法採用 Zigzag Scan。Zigzag Scan 是一種優先排列低頻項目，將高頻項目置放於尾巴處的排列法。

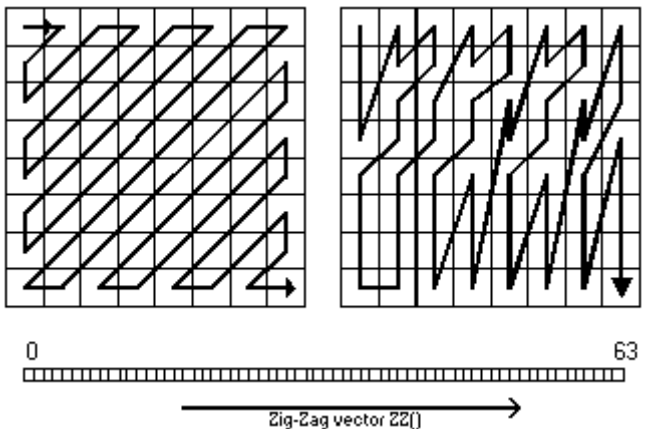

Figure 2.3 – Reordering (Zigzag scan)

VLC (Entropy Coding)

經過 Reordering 後，quantized DCT coefficients (transform coefficients)照著頻率排列。此時進行 Entropy Coding。Entropy

Coding 有許多種作法，像是 Variable Length Coding 和 Huffman Coding，兩者都可以有效壓縮目前的資料量。但是 Huffman Coding 有兩項缺點。一是 probability table 會跟著傳輸傳送到解碼端，這對於傳輸系統是個多餘的負擔；二是壓縮大尺寸的畫像時，probability table 必須等待壓製完畢時才能產生，這會產生延遲。因此，根據於 Huffman Coding 概念的其他 Variable Length Coding 也在選擇之列。MPEG-4 採用 Variable Length Coding。

RLC (Run-Level Coding)

Run-Level Coding 接在 VLC 之後。Run-Level Coding 主要功能是在於處理一連串的 0 值，當尾隨著非零值時，可以用 0 值的連續長度(Length of the Run)和非零值的個數(Level)表示。因此，Run-Level Coding 能把 VLC 產生的 codeword 進一步壓縮。

在儲存檔案或是傳輸資料時，RLC 根據 VLC codeword 組成的 bitstream 中位元的 0 值和 1 值的分佈狀況，用 RLC codeword 取代。對於一連串的 0，尾隨著一個非零的位元(在此意即 1 值)，使用 RLC 後，無需傳輸全部的 0，而只需要傳遞 0 的 run-length。Run-Level Coding 在 quantized DCT coefficients 的高頻部份特別有效。因爲 DCT coefficients 在高頻的強度不高(多數爲 0)，這是圖

片本身的特性。Quantized DCT coefficients 在高頻項的值通常皆爲 0，使用 RLC 進行編碼能更有效壓縮。

Profile (Simple Profile)

Profile 是一種統稱，代表 CODEC 所使用的解碼工具組合。Level 代表 CODEC 支援的解碼程度。MPEG-4 Visual 有許多 Profile，例如 Simple Profile 和 Advanced Simple Profile 等等。多媒體串流系統目前採用 MPEG-4 Visual Simple Profile。

2.1.2 位元流的意義 Bitstream

經過編碼的影像資料，編碼器以 layer 的型式，將其按照次序置入 bitstream 之中。MPEG-4 可以擁有多個 layer。Base layer 是一個低階的 layer，可以單獨進行解碼。而其他的 layer，稱作 enhancement layer，必須和比起他較爲低階的 layer(在 bitstream 的置放位置上也處在 enhancement layer 的前方)一起進行解碼。編碼器將影像資料寫入 bitstream，必須按照一定的句法寫入。以下列舉和 MPEG-4 Visual Simple Profile，以及多媒體串流系統有關的事項。

Start Codes 佔據 32 個位元，其中由 24 個已知位元(start code prefix)和末尾 8 個位元組成。Start code prefix 由 23 個 0 和尾隨的 1 個 1 構成 0000 0000 0000 0000 0000 0001 codeword。末尾的 8 個位元代表了

start code 的種類。大多數種類的 start code 只有一個特定值，而 video_object_start_code 和 video_object_layer_start_code 兩項 start code 則有多個特定值。Bitstream 由一連串的位元組成，需要特定的資訊才能正確的解碼。Start code 在 bitstream 裡承載了兩種資訊，Global Configuration Information 和 Elementary Stream Data，這兩者大略代表了解碼時所使用的環境參數，以及編碼過的影像資料本身。

name	Start code value (hexadecimal)
Video_object_start_code	00 through 1F
Video_object_layer_start_code	20 through 2F
Reserved	30 through AF
Visual object_sequence_start_code	B0
Visual_object_sequence_end_code	B1
User_data_start_code	B2
Group_of_vop_start_code	B3
Video_session_error_code	B4
Visual_object_start_code	B5
Vop_start_code	B6
……	……

Table 2.1 – part of Start Code Values

下面是實際進行編碼/解碼的部份情形

VisualObjectSequence(){	No. of bits
do{	
visual_object_sequence_start_code	32
profile_and_level_indication	8
while(next_bits()==user_data_start_code){	
user_data()	

}	
VisualObject()	
}while(next_bits()!=visual_object_sequence_end_code)	
visual_object_sequence_end_code	32
}	

Table 2.2 - Visual Object Sequence

VisualObject(){	No. of bits
visual_object_start_code	32
is_visual_object_identifier	1
if(is_visual_object_identifier){	
visual_object_verid	4
visual_object_priority	3
}	
visual_object_type	4
if(visual_object_type==”video ID”|| visual_object_type==”still texture ID”){	
video_signal_type()	
}	
next_start_code()	
while(next_bits()==user_data_start_code(){	
……	

Table 2.3 - Visual Object (part)

對於解碼器而言，解碼器所看到的 bitstream 和編碼器所產生的 bitstream 是一致的。若是不一致，將導致解碼失敗，必須等待下一個 start code 或是 resync marker，才有可能繼續進行正確的解碼。

概括而言，Bitstream 的組成可以看作是 Header + Data，Header 以 FLC (Fixed Length Coding)編碼，Data 部份以 VLC (Variable Length Coding)編碼。如果 Header 的某個位元出錯了，那麼解碼器將在不正

確的環境下工作，而 Data 會被捨棄。如果 Data 的某個位元出錯了，那麼在這個錯誤位元後的下一個 start code 有可能挽救剩下的 bitstream data，讓剩下的 bitstream data 能繼續正確的解碼。

2.2 即時傳輸協定簡介 The Real-time Transport Protocol (RTP)

對於一個串流系統而言，若是使用 TCP(Transmission Control Protocol)傳輸影音封包，客戶端將會因爲網路上的封包遺失，而時常面臨難以忍受的畫面延遲。使用 TCP 傳輸影音封包的缺點來自於 TCP 的特性，TCP 會嘗試回復封包，而回復封包的作法就是要求封包來源端重送一次封包。如果網路上的封包遺失狀況嚴重，使用 TCP 傳輸時就會在客戶端發出許多次重送封包的要求。此時的客戶端必須在等待重送的封包來臨之後才能進行進一步的影音服務。等待的時間是漫長的，對於客戶而言這是極度令人不悅的使用經驗。爲了避免這個狀況，多數人選擇使用 UDP 傳送封包。

和 TCP 不一樣的是，UDP(User Datagram Protocol)並不要求重送封包。在容易遺失封包的環境下 UDP 並不是個可靠的協定。UDP 不提供封包遺失偵測的功能，也不支持重送，但在網路擁擠的狀況下，多媒體串流系統使用 UDP 能避免在客戶端過久的等待。

RTP(Real-time Transmission Protocol)是一項能彌補 UDP 不足之處的協定。RTP 在 UDP 的上層工作，而且 RTP 利用了 UDP 的多工(multiplexing)及錯誤位元檢測(checksum service)兩項功能。事實上，在封包封裝時，RTP 的資訊被封裝在 UDP 的封包裡。RTP 能提供即時性的端對端傳輸服務，它也提供了下列功能：content type identification, sequence numbering, timestamping, monitoring QoS of data transmission。之後的章節將會詳述。

RTP 內含了兩項緊密搭配的協定，一項是 RTP 本身，負責資料傳輸；另一項是 RTCP(Real-time Transport Control Protocol)，負責監控資料傳輸的 QoS (Quality of Service)，以及參與對話(Session)的各項控制數據傳遞。

2.2.1 RTP

RTP 提供了端對端的資料傳輸服務，特別適合傳送即時影音資料。事實上，RTP 的主要目的就是要支援即時性的影音資料傳輸，將影音資料切割成數份封包，搭配 RTP header，再交由較底層的協定(如 UDP)傳送。

2.2.1.1 RTP packet format

RTP Header	RTP Payload

Figure 2.4 - RTP packet

如同 figure xxx 所表示，RTP 封包由兩者組成，Header 和 Payload。Payload 可爲 video 或是 audio 資料。Payload specification documents 規範了不同格式的檔案要如何置入 payload 才能使用 RTP 傳輸影音資料；RFC3016 中規範了 3GPP MPEG AV payload，RFC3640 則規範了 MPEG4 AV payload。

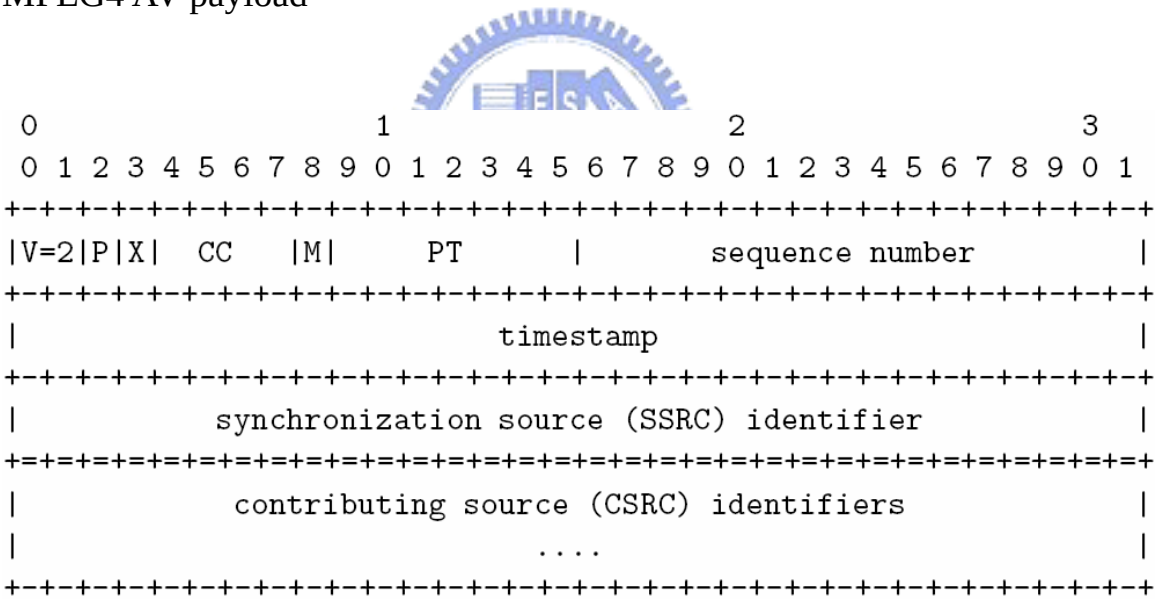

Figure 2.5 - RTP Header Format

- V (Version): 2 bits

  這代表目前 RTP 封包的版本，RFC3550 中，設定 V 值爲 2。

- P (Padding): 1 bit

  表示封包尾巴是否進行 padding。Padding 的原因有兩種，一是爲了進行加密，二是爲了交由較底層的協定進行傳輸。

- X (Extension): 1 bit

  表示是否進行 RTP header extension，在目前的 RTP header 後面，再加上別的 header。

- CC (CSRC Count): 4 bits

  由於串流技術支援多個來源，CSRC(Contributing Source identifier) 代表了參與本次串流通訊的其中一個來源，CC 表示了目前的來源編號是第幾個(0~15)。

- M (Marker): 1 bit

  用來標示重要事件的旗標。

- PT (Payload Type):7 bits

  payload type 代表目前所承載的內容型態。

- Sequence number: 16 bits

  在串流通訊開始時，於 0~65535 中亂數決定一個數字作爲開頭，之後每送一個 RTP 封包便自動加一(monotonically)。在抵達 65535 後，下一次的 sequence number 會由 0 開始繼續增加。在串流通訊開始時亂數決定 sequence number 是爲了避免 know-plaintext 的攻擊。Sequence number 可以在接收端用以偵測封包遺失，或是封包脫序到達的狀況。

- Timestamp: 32 bits

  它紀錄了 payload 第一個位元組的 presentation time for stored video 或是 sampled time for live-captured camera data。Timestamp 重建了串流的時間點，它可以用於進行 jitter 的計算和同步化。

Timestamp 的初始值也是亂數決定的，理由同樣是爲了避免 know-plaintext 的攻擊。對於不同的媒體串流而言，timestamp 可以使用不同的增加率，offset 值也可以不同。

- SSRC: 32 bits

  Synchronization Source(SSRC)代表了串流通訊的來源。SSRC 在任何一個 RTP session 中都是亂數產生，而且是獨一無二的。如果某個傳送者在一個 RTP session 中進行多個串流通訊，那麼這個傳送者將會有多個不同的 SSRC，這能避免只使用網路位址進行來源辨識的問題。

- CSRC list: 32 bits, 0~15 items

  CSRC 的清單，代表了 payload 資料來自於哪些 CSRC。

2.2.1.2 RTP Example

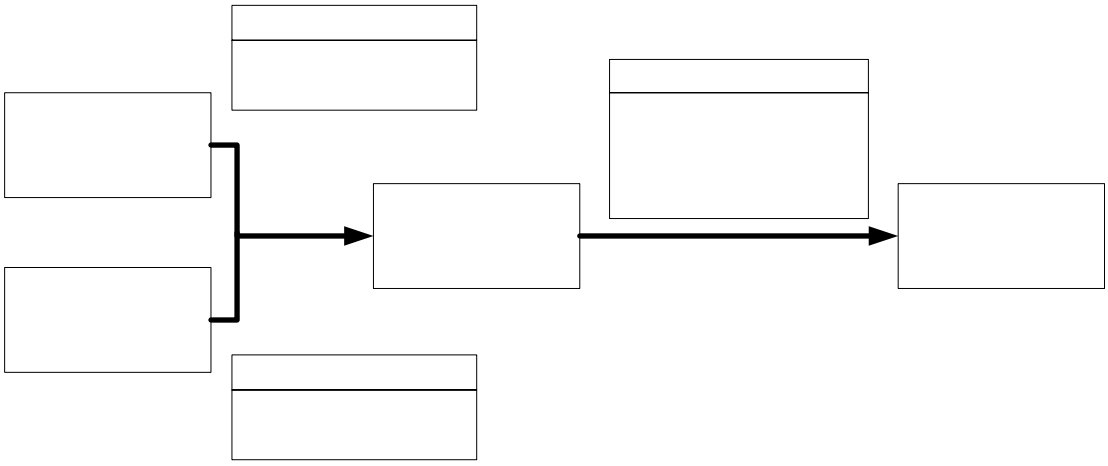

Figure 2.6 - RTP example

Figure xxx 是 RTP example，可以看到 A 的 SSRC 是 357598721，B 的 SSRC 是 234746508，兩者的資料經過 Mixer 的處理後，payload data 是

A 和 B 的混合。對 C 而言，他所看到的串流數據，SSRC 是 Mixer 的 824098109，然後 C 看到了兩個 CSRC，分別是 A 的 357598721 和 B 的 234746508。這就是 RTP 的工作情形。

2.2.2 RTCP

RTCP 主要的功能是向 session 中的各個參與者，提供有關接收的服務品質(reception quality of a session)的資料。這份資料對於 flow and congestion control protocol 十分有用，有些 protocol 能夠利用這項資訊，適應性地(adaptively)調整編碼器 encoder 的品質，期待降低網路擁擠程度。

RTCP 的封包格式中帶有 RTP source 的 CNAME(canonical name)，如果在 SSRC identifier 出現了衝突，每位參與者都需要利用 CNAME 鎖定各個來源。RTCP 也能幫助 sender 和 receiver 調整 clock drift(時刻飄移)。

2.2.2.1 RTCP Packet Types

RTCP 有五種封包種類：

- SR: Sender report，由主動傳輸媒體內容者發出，SR 記錄 session 參與者的傳輸和接收各項數據。
- RR: Receiver report，由被動接收媒體內容者發出，RR 記錄 session 與與者的接收相關數據

- SDES: source description items，包含 CNAME。
- BYE: 表示退出及結束通訊。
- APP: 和應用程式有關的功能。

2.2.2.2 RTCP Packet Format

每一種 RTCP 封包，皆以一個近似 RTP 封包且具有固定格式的檔頭開始。接在檔頭之後的是各種資料數據，由於這些資料數據未必能以 32-bit 爲單位收尾，必須視情況在 RTCP 封包的尾巴補 0(padding)。

由於在 RTCP 的檔頭欄位裡記錄了封包長度和補 0 的長度，這使得 RTCP 封包具有「可堆疊性」(make RTCP packets “stackable”)。可參考 figure 如下。

RR	SDES: CNAME	BYE

Figure 2.7 – a typical compound RTCP packet

RTCP 的各個種類封包皆有特定的欄位，以下簡略介紹 Sender Report packet：

- V: 2 bits，version，在 RFC3550 令 V 值爲 2。
- P: 1 bit，padding，表示 RTCP 封包是否含有 padding。
- RC: 5 bits，Reception Report Count，表示在 RTCP 封包中含有多少個 report blocks。

- PT: 8 bits，payload type，設定 PT=200，表示是 Sender Report。
- Length: 16 bits，整個 RTCP 封包，包括檔頭，receiver info，report blocks 和 padding 的總長度。
- SSRC: 32 bits，產生這個 Sender Report 的 synchronization source identifier。
- NTP timestamp: 64 bits，指示封包的 Network Time Protocol(NTP) 的 clock time。
- RTP timestamp: 32 bits，和 NTP timestamp 同樣也是表示時間.
- Sender’s Packet Count: 32 bits，表示從傳輸開始至今，傳送了多少 RTP 封包。
- Sender’s Octet Count: 32 bits，表示從傳輸開始至今，傳送了多少 RTP payload bytes。
- 下略。

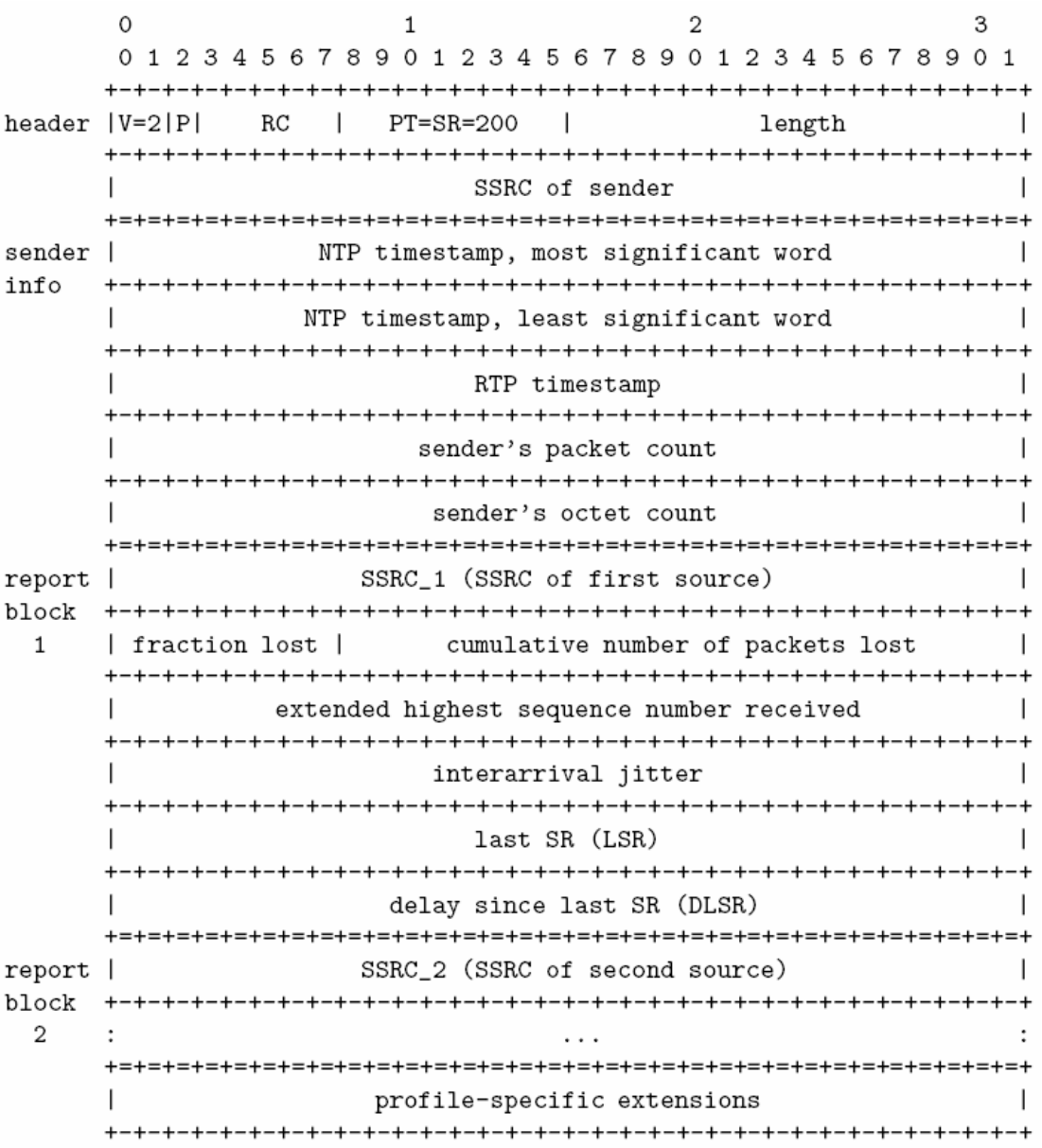

Figure 2.8 – overview of RTCP sender report format

第三章 錯誤隱藏演算法的介紹與設計

3.1 背景介紹

長久以來，通訊系統一直面臨著一個問題，資料在傳輸的途中因爲傳輸通道的噪音或是其他因素導致傳輸錯誤。這代表通訊系統必須承擔資料遺失或是資料內容被改變的風險。

傳輸錯誤大致上可以分成兩類：random bit errors 和 erasure errors。

1. Random bit errors 泛指 bit inverse, bit deletion 和 bit insertion 三種。封包中的任何一個位元皆有可能出錯。
2. Erasure errors 來自封包交換網路中的封包遺失(packet loss)，儲存媒體的錯誤爆發，或是系統當機。

使用者一邊進行下載，一邊觀賞已經接收的影音資料，這是多媒體串流系統強調的服務。由於目前網路的頻寬有限，爲了達到流暢的播放效果，壓縮過的影音資料每秒鐘資料量應等同於下載影音資料時所使用的資料量。影音資料的原本所需容量太大，必須使用高效率的影音壓縮編碼。高效率的影音壓縮編碼必然使用近似於 variable length coding 的技術。

對於使用 variable length coding (VLC)編碼技術的資料而言，

random bit errors 改變了封包中資料的位元流，在客戶端無法使用 VLD 解碼不存在的 codeword，這間接導致無法解碼其他 code word。而 erasure errors 也會造成 VLD 解碼失敗。解碼器在解碼失敗後有不同的處理機制，有些解碼器選擇保留已經解碼的 macroblock 資料，有些解碼器選擇丟棄解碼失敗的資訊。前者的問題是尚未解碼的 macrolbock 會造成之後解碼的 mpeg-4 video frame 畫面的 error propagation 現象；後者的問題來自於之後解碼的 mpeg-4 video frame 指向不應參考的 reference frame。

形成 error propagation 的原因來自於 mpeg 編碼技術所使用的 motion estimation / motion compensation。爲了節省畫面所使用的 bit 數，P frame 和 B frame 皆使用了 motion estimation / motion compensation 兩項技術。由於畫面和畫面之間的相依性(time dependencies)，一張畫面出錯了將連帶影響接下來直接以它爲 reference frame 的畫面，和與其間接相關的畫面。

爲了解決傳輸錯誤所帶來的影響，許多人提出了各種 error concealment 演算法。有些演算法專注於在編碼器端(source coder)和傳輸系統(transport system at server end)上加強，另外一些演算法致力於解碼器端(decoder)和傳輸系統(transport system at client end)的錯誤抵

抗能力，還有一些演算法設計一種能讓編碼器端和解碼器端互動及合作減輕錯誤影響的架構。

3.2 錯誤偵測 Error Detection

在整個多媒體系統進行各種 error concealment 技術之前，客戶端必須先偵測資料傳輸途中是否產生了錯誤，以及錯誤發生的位置。

根據功能[3]，整個多媒體系統大致上分爲三個區塊：編碼區(source coder)，傳輸區(transport coder and transport decoder)和解碼區(decoder)。在傳輸區進行錯誤偵測的最簡單的作法就是加上 header 資訊[3]。Header 中的 sequence number 可以用以偵測 packet loss。

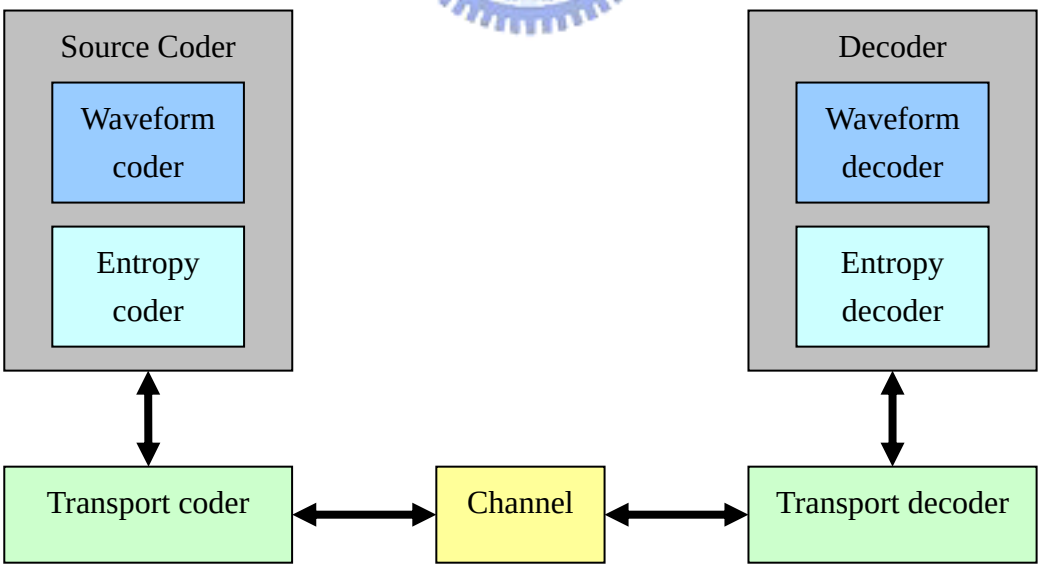

Figure 3.1 - Functional block diagram for media communication system

在解碼區進行錯誤偵測必須由解碼內文(context)的存在與否決。

如果遇上了不屬於 decoding table 的 codeword，這一定是來自於傳輸錯誤。傳輸錯誤會造成像是 quantization step size 的值爲 0 和 DCT 係數超出其所應存在的值域等等錯誤。

3.3 編碼器端的錯誤隱藏作法 Forward Error Concealment

目前有許多種 Forward Error Concealment (FEC)作法，主要是在於編碼區塊和傳輸區塊中，增加多餘(redundant)的控制位元。有些作法在編碼區塊中的 waveform coder 和 entropy coder 中加入控制位元降低傳輸錯誤的影響，有些作法則利用網路設備進行傳輸系統中不同層級的 Quality of Service (QoS)。

目前主流的 FEC 作法是在 source coder 端，進行不同層級的編碼，以不同的 layer 承載資料，再使用 transport layer 不同層級的 QoS 對 layer 進行傳輸，這種作法稱作 Layered Coding with Transport Prioritization[4]。由於人眼視覺系統對於畫面有較高的容忍度，偶爾的低解析度的畫面不會對使用者造成太大的困擾。根據這個觀念，基本的和重要的資料在編碼後存放於 base layer，可以忽略的高解析的細節部份在編碼後存放於 enhancement layer。Base layer 的資料在解碼端解碼之後產生可接受的畫面品質，而 enhancement layer 的資料在解碼之後，會附加於 base layer 的解碼後的畫面上，得到更精細的畫

面品質。

在傳輸系統不出錯時，base layer 和 enhancement layer 所解出的資料合在一起，可以順利重建畫面。如果傳輸系統出錯，因爲 base layer 以較高 QoS 層級進行傳輸，可以確保成功傳送；enhancement layer 以較低 QoS 層級傳輸，遺失了也無所謂，因爲 base layer 承載的資訊在解碼成功後可以得到人眼可以接受的畫質。

實現 layer coding 的方法有很多種，可以在 temporal domain, spatial domain 或是 amplitude domain 進行。例如 base layer 以較低的 frame rate 傳輸，或是 base layer 承載的資料的解析度較低，甚或 base layer 承載的資料使用較大的 quantization step size。在能夠得到人眼視覺系統中可接受的品質的保證下，進行剩餘資料的編碼，置入 enhancement layer 中。

Layer coding 必須和 transport prioritization(layer 以不同層級的 QoS 傳輸)合作才能有效實現 error concealment 的目的。

Multiple-Description Coding (MDC)假定伺服器端和客戶端之間存在著多條不重疊的傳輸通道[5]。每條傳輸通道之間發生的事件是獨立的，在機率上，多條傳輸通道同時刻經歷錯誤的機會十分微小。

Layered coding with transport prioritization 則假定了 base layer 使用了一個 error-free(無錯誤)的傳輸通道進行傳輸，這和 MDC 的看法不同。

MDC 將同一項信號來源進行多種編碼(description)，不同的 description 以不同的傳輸通道傳輸。在客戶端的解碼器會根據收受信號的正確與否，決定是否解碼。每一種編碼(description)都能夠得到可接受的畫面品質，解碼器端接收越多種正確傳送的編碼就能得到越好的畫面品質。在任何時間點，只要全部傳輸通道中至少一條通道正確運作，客戶端就不會得到嚴重受損的畫面品質。

另外一種演算法則是在編碼過程中安插了少許控制位元(controlled bits)。由於傳統的編碼設計致力於消減在統計上和視覺上所有的多餘資訊(redundancy)，經過編碼後的 bitstream 只保留解碼時必要的資訊。如果 bitstream 中的位元因爲封包遺失或是遭到通道噪音干擾而出現錯誤，剩下的可用資訊過少，這會增加在解碼器端進行錯誤隱藏的工作難度。如果 source coder 刻意地保留少許在正常編碼模式下被視爲多餘的資訊，這能有效降低在解碼器端進行錯誤隱藏的難度。這樣的技術稱作 Robust Waveform Coding。最簡單的例子就是在 mpeg-4 編碼中，設定 I frame 也接受 motion estimation 和 motion compensation 步驟，編碼過 bitstream 則內含對應的 motion vector 資

訊。[4]

Source coder 也可以在 entropy coding 階段外加控制用的位元，這些位元可以用來偵測錯誤和避免 error propagation。如果在 entropy coder 中指定 VLC codeword table 中的任一個 codeword 爲 synchronization code(同步碼)，解碼器端可以在讀取 synchronization code 時重新獲得同步(resynchronize)[6]。之前提及的 variable length coding 技術在使用 synchronization code 後，codeword 的解碼失敗雖然會造成後續的 codeword 無法正確解碼，但是無法解碼的範圍被限制到下一個 synchronization code 之前。在解碼器端持續讀取 bitstream 直到遇到 synchronization code 之後，解碼器端和 bitstream 重新同步(resynchronize)。Synchronization code 能夠減輕 error propagation 在畫面中造成的影響。在 mpeg-4 video coding 中，video packet header(或稱 slice)就擔任了這種角色。

還有一些在伺服器端進行的 error concealment 技術考量到了傳輸通道的特性，在 bitstream 中加入 ECC(error-correction code)，也是一種增加控制位元的作法。

3.4 解碼器端的錯誤隱藏作法

Error Concealment by Post-processing at the decoder

由於在傳輸過程中連續兩張畫面經歷同樣錯誤事件的機率太小，對於解碼錯誤的畫面而言，前一張畫面通常是正確接收解碼的。根基於此，許多後製處理的影像演算法嘗試利用前後畫面之間的相關性，對於解碼錯誤的畫面進行錯誤隱藏。雖然經由後製處理過的重建畫面和預期正確接收的畫面仍然有著品質上的落差，由於人眼視覺系統能夠容忍視覺畫面中較多的高頻扭曲，經過後製處理重建後的畫面對多數使用者而言仍然是可以接受的。

根據不同的解碼器設計，解碼錯誤時，有些解碼器能保留已經正確解碼的區域(block)，有些則不能。有些則假定 source coder 使用了 synchronization code 之類的技術，解碼錯誤的畫面只有少部份的 block 無法解碼。大部份的 post-processing 技術皆假設編碼器使用 synchronization code，解碼器能有效地限制解碼錯誤的範圍和保留正確解碼的資訊。

對於受損的 block，十分常見而且簡單的錯誤隱藏作法就是將在前一張畫面中相同位置的 block 畫面複製，再貼至目前受損的 block。對於畫面中的物體移動緩慢時，或是物體靜止時，可以得到很好的效果。若是能進一步得到受損 block 的 motion vector，進行 motion compensation，便能夠在前一張畫面找到更適合的 block 畫面複製，

貼至受損 block 得到更好的視覺效果。這項演算法和 layered coding 搭配時能夠得到很好的效果，motion 的資訊記錄在 base layer 之中可以確保錯誤隱藏時的 motion vector 來源。然而這項演算法的問題來自於並非所有情況下都能得到可用的 motion 資料。

不只畫面和畫面間具有關連性，連 block 和 block 之間也有關連性，那就是任一物體在連續兩個 block 之間所構成的線條是連續的。利用影像訊號在大多數畫面的 block 和 block 之間所具有的連續性，名爲 Maximally Smooth Recovery 的演算法嘗試將受損 block 和受損 block 週遭區塊之間的差異值最小化。[7]

假定受損 block 的相鄰區塊爲正確解碼，爲了要重建受損 block，必須要得到在受損 block 中所遺失的 DCT 係數。求得受損 block 的遺失 DCT 係數有各種方式，一是空間上的線性內插，利用相鄰區塊的 block 的 DCT 係數；另一是時間上的線性內插，利用前後畫面的相對應位置 block 的 DCT 係數；最後是頻率上的線性內插。如果受損的 block 遺失了全部的 DCT 係數，那麼 DCT 係數估算作法縮減至空間的線性內插及時間的線性內插兩種。

在求得受損 block 遺失的 DCT 係數後重建 block，計算重建後的 block 的各個 pixel 之間的相差值，將這些相差值平方後加總可得平方

差總合。藉由不同的方法尋找遺失的 DCT 係數，最小化平方差總合能夠盡可能地讓受損 block 的重建畫面與周遭區塊平滑地連接。通常只要得到前 15 項 DCT 係數，就能重建出令人能接受的畫面品質。

根基於平滑性的演算法還有 Projection onto Convex Sets (POCS) 一種[8]。凸面集合(Convex Set)來自於重建畫面中 block 是否等向或是具有某種特定的方向。一開始先讓一組 blocks 接受 edge existence 的測試。Block 在此處被分成 monotone block 和 edge block 兩種。Edge block 中的邊緣具有方向性，此方向的角度指向等分於 0°-180°之間的八個角度。POCS 不斷地對一組 block 進行分類測試，通常在五至八次內可以得到結果，之後進行重建。

Spatial- and frequency- domain interpolation 也是根基於平滑性所提出的演算法[9]。因爲相鄰 block 之間的畫面差異不大，受損 block 的係數，和週遭 block 相對應位置的係數有極大的機率是接近的。首先決定受損 block 的係數，可由周遭四個 block 的係數進行內插計算而得；之後重建受損 block。如果受損 block 中全部的係數皆遺失了，便必須對週遭四個 block 的所有係數進行內插。在取得係數之後，重建的畫面必須和週遭四個 block 進行 boundary pixel 差異值的最小化。能夠最小化的差異值的 block 係數就是我們選擇的重建 block 係

數。

在 post-processing error concealment 中所提及的演算法皆假設解碼器知道受損 block 的 coding mode 和 motion vector 種種資訊[10]。如果 coding mode 資訊遺失了，那麼必須在估計目前的 coding mode 之後才能重建畫面。由於大多數影像畫面在空間上和時間上的平滑性，coding mode 和 motion vector 可以經由內插相鄰的 block 資訊而得。

估計 coding mode 最簡易的方法就是將受損 block 視爲 intra-coded，再利用四周 block 的資訊進行重建。如果假設爲 inter-coded，就必須估計 motion vector。估計 motion vector 的作法大致上有幾種：將 motion vector 設定爲 0，設定爲上一張畫面相對應位置的 motion vector，設定爲四周 block 的 motion vector 的平均數，或是設定爲四周 block 的 motion vector 的中位數種種。決定使用哪種估計 motion vector 方法同樣地須由重建 block 和週遭四周 block 的連續性而定。

3.5 編碼器和解碼器互動式的錯誤隱藏作法

Encoder and decoder interactive error concealment

如果客戶端能夠經由一條傳輸通道向伺服器端交流影音資料傳

輸的網路交通數據，客戶端和伺服器端雙方可以合作進行錯誤隱藏。雙方合作的層級可以在雙方的傳輸系統端進行或是在雙方的編碼解碼層級進行。根據解碼器回傳的數據，編碼器中的編碼參數可以自行調整。在傳輸系統中，重新傳送或是使用 FEC 所佔用的頻寬能根據客戶端的回傳數據而調整。[11]

最直覺的概念，解碼器在解碼錯誤後，將解碼錯誤後的分析資訊回傳編碼器，編碼器因此適應性地調整編碼參數，有機會在解碼器端減輕或是消除 error propagation 的影響[12]。本演算法最簡單的作法就是當解碼器偵測到錯誤，即要求編碼器以 intra coding 的模式對下一張畫面進行編碼，error propagation 可以在約一個 round-trip time 後因爲 intra-coded frame 的來到而消除。

某些多媒體即時串流系統採用另一個概念，客戶端遇到傳輸錯誤時，由解碼器端向伺服器端發出重新傳送的要求。然而，同時間解碼器端並不等待重新傳送後的資料，反而選擇繼續解碼[13]。關於傳輸錯誤造成的畫面品質降低和 error propagation 造成的影響，選用另一種錯誤隱藏演算法進行補救。解碼器端同時記錄與受損畫面有關的 block 資訊和 error propagation 所造成的影響區域。當重新傳送的資訊正確地抵達客戶端時，將會一一更正這些被影響的 block。

根據重新傳送的資料進行更正是項極爲複雜的工作，複雜程度由 motion compensation 的 pixel prediction 精確度而定。

爲了克服網路上的高資料遺失率，也有一種演算法採用 layered coding，並且在伺服器端同時傳送多份內容一樣的資料(multi-copy)。同時傳送多份內容一樣的資料除了提高重新傳送的成功性，也降低了要求重新傳送的次數，這暗指由於多次的重新傳送要求所造成的網路延遲(delay)將被縮短。每層 layer 所應傳送的內容份數必須由該層 layer 的重要性而定。考慮到封包在網路上的 gateway 被丟棄的情況，演算法必須有效控制各層 layer 封包的份數，避免超過網路的傳輸能力，另外也避免造成網路堵塞。

3.6 在 compressed domain 進行錯誤隱藏的概念與設計

根據之前的認識，我們可以理解 MPEG-4 錯誤隱藏的多數作法是在 pixel domain 操作。接下來要介紹的是利用 compressed domain 的資訊而發展的 Error Concealment 演算法。

由於目前這套多媒體串流系統所使用的 MPEG-4 CODEC, SP_Encore.lib 和 SP_Decore.lib 並不支持 Video Packet，在解碼器遇到解碼錯誤時，是以整張畫面爲單位進行捨棄。此時使用 compressed

domain 的 error concealment algorithm 是有必要的。Pixel domain 的 error concealment algorithm 難以即時地進行補救，它需要大量的運算量和運算時間。由於目前中央處理器的能力已到達極限，無法縮短運算時間，這意謂在多媒體串流系統上使用 compressed domain error concealment algorithm，客戶端有較大的機會提供一個重建畫面。MPEG 和 H.26x 編碼系統的特色是藉由 motion estimation 和 motion compensation 等技術進行影像壓縮。在影像壓縮的過程中，motion vector 保留了物體移動的資訊。這些 motion vector 資訊就是所謂的 compressed domain 的資訊。

2^{nd}-order motion field image processing:

由於最新畫面的解碼失敗，不進行重建的結果是後續來到的 MPEG-4 video frame 會因爲對應到不屬於自己的 reference frame，造成 motion compensation 時，motion vector 指向的 macroblock 或 block 不是對應於自己的 macroblock 或 block，這會造成畫面的混亂。

我的作法是重建一張照片(create a new image)。在 new image 上，以 block by block 的方式重建。Block 的來源是根據已解碼的畫面而得，new image 的 block 和已解碼的畫面關係最密切的就是最新解碼完的畫面，在此稱作 previous frame。New image 和 previous frame 之

間的關係就是 motion vector，因此要重建畫面，首先就是要找到 new image 的 block 的 motion vector。

[14]中的構想：如果解碼最新的畫面失敗，便利用最後一次解碼成功的圖片(previous frame)所持有的 motion vector，進行 motion forward projection。

Forward motion projection 的意義是利用 motion vector 的 magnitude，和相反的方向(inverse direction)，進行 block projection 的動作。

只使用 previous frame 的 motion vector 進行 forward motion projection 可能還不足夠。如果能利用兩張以上的 frame 的 motion vector 資訊，經過處理後再進行 forward motion projection，得到的重建畫面應該更能讓大多數使用者感到滿意。

由於 motion vector 在 MPEG 之類的影像編碼技術中代表的意義是 block 在 frame 之間移動的位移，若是把每一個 block 都視作一個 object(物體)，或是將 object 視作 block，那麼 motion vector 就是 object 的 velocity。Object 的移動不只是看當時 velocity 的計算結果，還需要加上 acceleration 的影響。在這個演算法裡，object 就是 block，velocity

就是 motion vector，那麼 acceleration 就是相鄰畫面間的同一 object 的 motion vector difference。

考慮到 object 在連續畫面之間的連續性，在 new image 之前的兩張畫面 previous frame 和 2nd previous frame 必然擁有著 new image 內的 object。使用 previous frame 和 2nd previous frame 的 motion vector 可以估計 object trajectory(物體軌跡)。根據 object trajectory，我可以推算出 new image 的特定 object 的位置。目前的 object 是以 block 爲單位出發，那麼找出相鄰 frame 的同一 object 的作法就是找出相鄰 frame 的同一 block，這稱作 block correspondence for object association。更精確地講，要找出相鄰 frame 的同一 object，就是要找出相鄰 frame 的同一 block 的位置。能夠找到同一個 block 的位置就能找到它的 motion vector，這也意謂我可以計算 object 的 acceleration。

如果在 previous frame 中得到 block correspondence，意謂在 2nd previous frame 中能得到一個 corresponding block 代表同一個物體 (object association)。Corresponding block 不一定座落於以 8 爲倍數的座標上，無法持有 motion vector，但考慮到 block 所攜帶的 motion vector 就是 object 的移動資訊，如果 corresponding block 和 2nd previous frame 中某一塊 adjacent block 的有很高程度的重疊，那麼這一塊

adjacent block 極有可能攜帶了 corresponding block 的資訊，也就是 motion vector，這一塊 adjacent block 的 motion vector 可以視作 corresponding block 的 motion vector。圖示請參考 figure xx。

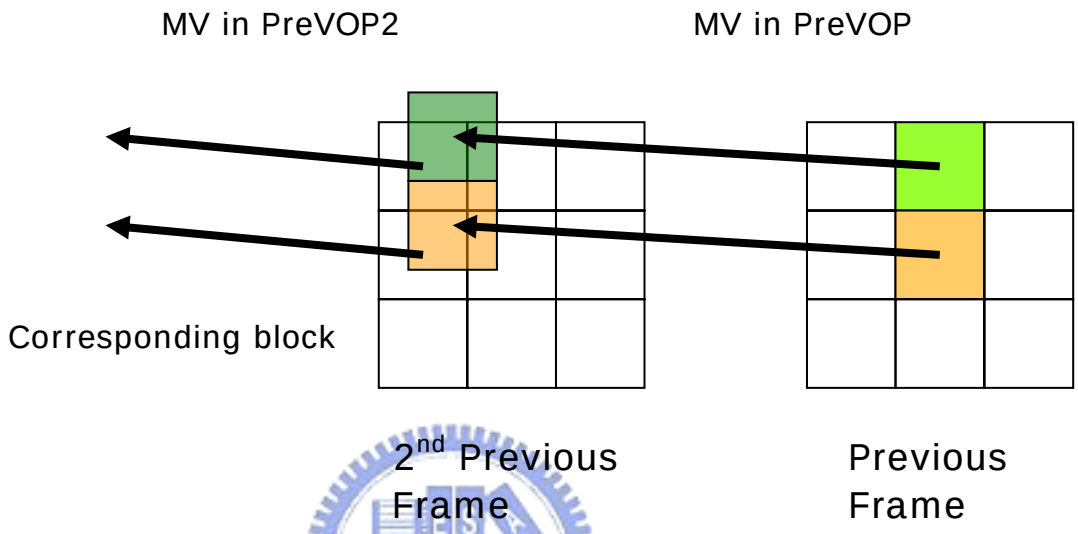

Figure 3.2 – Block Correspondence

得到 adjacent block 的 motion vector 便能進行 acceleration 計算，也就是能得到 object 在 new image 中會以什麼 motion vector 出現。Object 在 new image 中的 motion vector 就是 previous frame 中的 motion vector 加上 acceleration。根據這個計算結果，將 motion vector 進行 forward motion projection。每一個投射的 block 並不會精準地落在以 8 爲倍數的座標上。New image 上的 block 投影通常會佔據四個 adjacent blocks。如同之前的概念，block 的 motion vector 承載的是 object 的位移，如果它和 object 的重疊程度很高，那麼它所承載的 motion vector

就會是 object 的 velocity。進行 forward motion projection 之後，每個 new image 中的 block 會決定它要承載哪個 block 的 motion vector，決定的基準是面積重疊度。New image 中的大部份 block 藉此成爲 new image 中的 corresponding block。圖示請參考 figure xx。

由於從 previous frame 進行 forward motion projection 時，有些 new image 的 block 並沒有得到 object 的投影，意思是它們無法代表任何 object，這些 new image 的 block 不屬於 new image 的 corresponding blocks，我將這些 block 當作是不存在於 previous frame 的 static background(背景)。因爲，當這些 block 和物體有關時，它們必然會是 corresponding block 中的一員，當這些 block 屬於 previous frame 的 background，它們得到的 motion vector 是 0。造成這種現象的主要原因是因爲 previous frame 中，這些 block 的位置存在著一個 moving object，當這個 moving object 在 new image 中移動到新的地點時，被遮蔽的背景(background)就顯露出來。

對於這些 new image 中不屬於 corresponding block 的 background，直覺的處理方式就是從 2nd previous frame 中複製同樣位置的內容(content)。因此，new image 中，被投射的 block 和沒被投射的 block 都得到了解決。

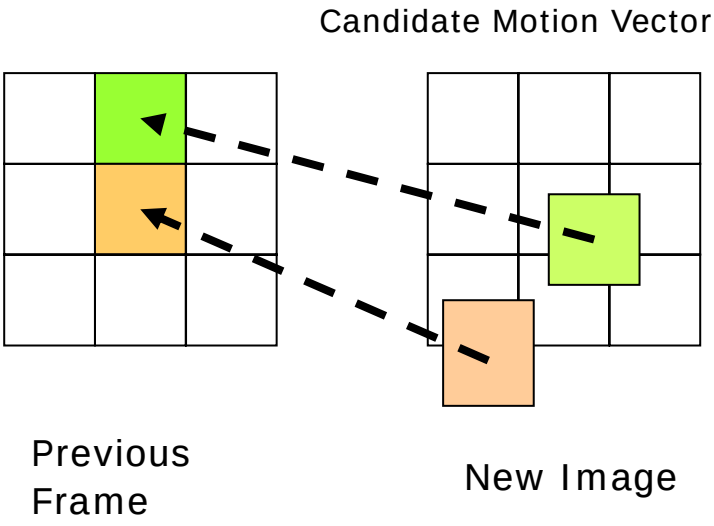

Figure 3.3 – forward motion projection

藉由 motion vector 的建立，和 2nd previous frame 的 copied content，new image 便能順利的建立。

第四章 多媒體串流系統與錯誤隱藏演算法

4.1 多媒體串流系統開發環境

由於 Error Concealment 操作於 client 端，因此我將專注於 client 端的功能實現和操作環境。

4.1.1 硬體要求 Hardware Requirements

- 一部與 IBM 相容的標準個人電腦。
- 一台 BUFFALO Air Station WHR-G54S 無線區域網路路由器，支援 IEEE 802.11b/ 802.11g 規格，並附上 4 個內建 port。
- 一個 D-Link DWL-G520+ 採用 PCI 插槽式的無線網路卡，支援 IEEE 802.11b/ 802.11g 規格。

4.1.2 軟體要求 Software Requirements

- 安裝 Microsoft® Windows® 2000/NT/XP
- 安裝 Microsoft® Visual C++ 6.0
- 安裝 MPEG-4 decoder library

4.1.3 工作環境 Working Environment

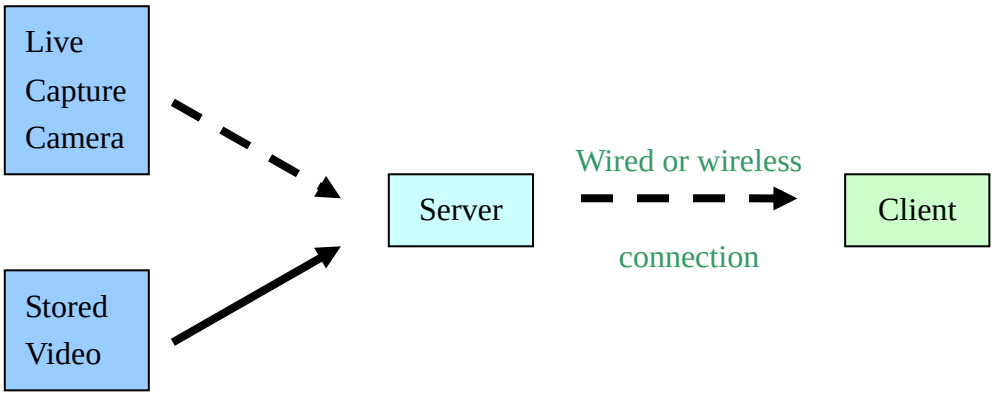

Figure 4.1 – basic description of streaming server-client working environment

4.1.4 Expected Capabilities at Client End

1. 客戶端能接受及回應使用者的輸入。
2. 客戶端能與伺服器端建立連線。
3. 客戶端能向伺服器端發出 RTSP 指令。
4. 客戶端能夠根據系統實際運作情形更改 RTSP state。
5. 客戶端能藉由 TCP 接收來自伺服器的 RTP 封包。
6. 客戶端能藉由 UDP 接收來自伺服器的 RTP 封包。
7. 客戶端能夠接收來自伺服器的 RTP 封包，並且 de-packetize (解開封包) RTP 封包。
8. 客戶端能夠解碼 RTP 封包的 payload 資料(MPEG-4 audiovisual data)。
9. 客戶端能夠在解碼失敗時進行 MPEG-4 Error Concealment。

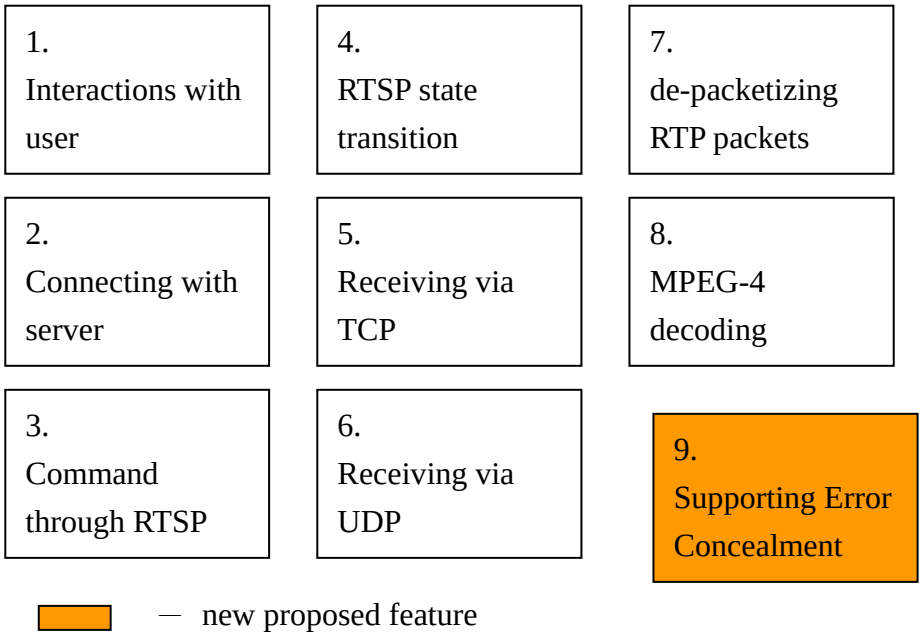

Figure 4.2 – tasks that the client must cope with

4.2 多媒體串流系統客戶端架構

Multimedia Streaming Client Architecture

多媒體串流系統的客戶端，主要分爲三個部份，一個部份負責傳輸系統，另一個部份負責將封包所組成的 mpeg-4 video frame 解碼，最後將解碼後重建成功的圖片顯示於使用者介面上。

傳輸系統可再細分爲三個部份，一個是負責接收封包，另一個是將封包收入暫存記憶體後進行排列，排列過的封包就是 mpeg-4 video frame 的 bitstream，再交由解碼器解碼。第三個部份是負責接收和處理 RTSP 命令。

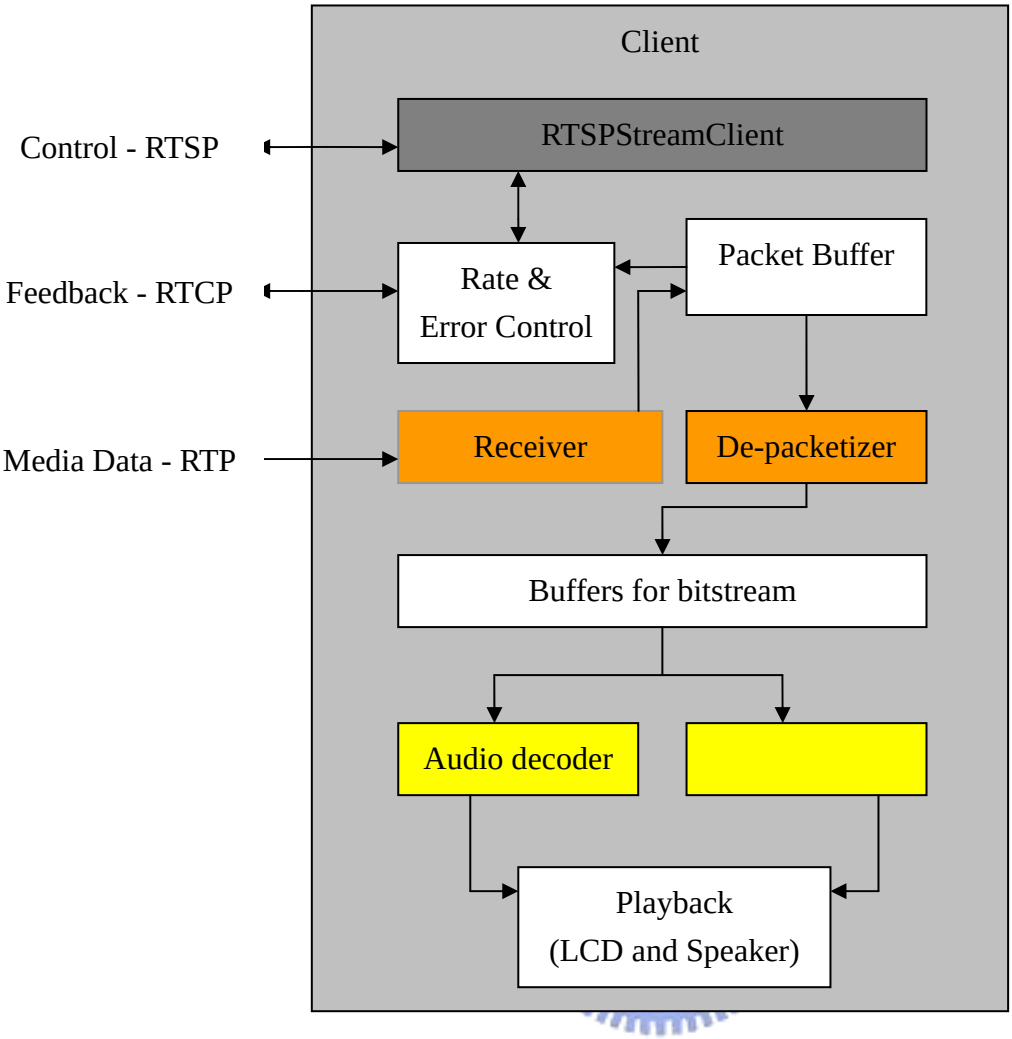

Figure 4.3 – Client Architecture in Multimedia Streaming System

4.2.1 多媒體串流系統客戶端工作模組

Modules in Streaming Client End

多媒體串流系統是以「模組化」爲原則進行開發的。以「與客戶進行互動的 Dialogue」、「負責傳輸資料命令事務」和「進行 MPEG-4 編碼/解碼」三個方向進行模組化。

多媒體串流系統在客戶端使用了三個模組。

1. Client_Dialogue 模組：

- 負責建立使用者介面以及定義各項輸入和輸出，和使用者互動。
- 以 figure xx 而言，負責 figure xx 中的第一項事務。
- 由於新加入了 Error Concealment 功能，在收到 SP_decore.lib 解碼失敗的訊息後，會開啓 Error Concealment。
- 以 figure xx 而言，負責 figure xx 中的第九項事務。

2. SP_decore.lib 模組：

- 負責解碼
- 以 figure xx 而言，負責 figure xx 中的第八項事務。

3. Sub_Client 模組：

- 使用多媒體串流系統客戶端中，所有有關傳輸的事務。包含與伺服器端要求連線，設定連線參數和建立連線。關於接收 RTP 封包和收發 RTSP 命令也是由 sub_client 模組負責。
- 以 figure xx 而言，負責 figure xx 中第二項到第七項事務。

4.2.2 多媒體串流系統客戶端執行緒

Running Threads in Streaming Client End

在軟體開發的過程中，除了引入模組化的概念，還必須使用執行緒 (threads) 的觀念。由於在 Operating System 內，程式是以 single process 的型態存在，無法保證能即時處理程式數據。因此程式有必要從 process 型態進入 thread 型態。

在開發多媒體串流系統客戶端時，現有的三個模組進行了更精細的切割。計有 GUI thread，VideoStatusTimeFunc thread，AudioStatusTimeFunc thread，RTSPProcessorThread thread 和 DePktThreadIsK thread + RecvThreadIsK thread 或是 TunnelMain thread。

- GUI thread
  建立使用者和程式之間的互動圖形介面，定義各種按鍵的功能，以及將影像展示在監視器上。
- RTSPProcessorThread( )
  接收來自於 GUI thread 的訊息，RTSPProcessorThread( ) 能夠處理各種 RTSP 指令，及其中夾帶的訊息。
  RTSPProcessorThread( )同時也負責維護多媒體串流系統客戶端的串流狀態。

上述兩個 thread 在多媒體串流系統客戶端一開啓時即產生。

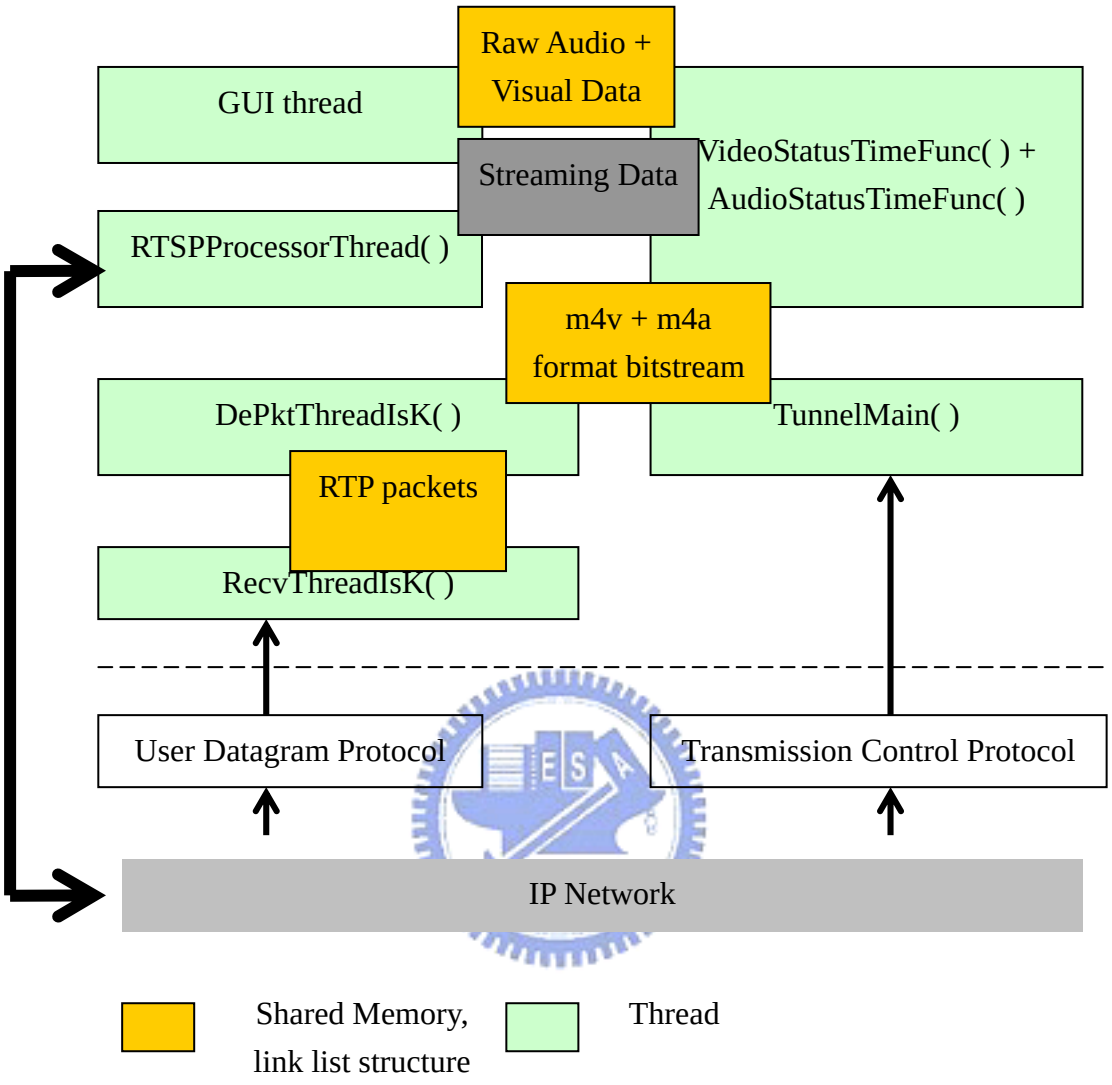

Figure 4.4 – Threads in Multimedia Streaming Client

- VideoStatusTimeFunc( ) 和 AudioStatusTimeFunc( ) 負責呼叫 SP_decore.lib 模組，將存放在記憶體中的 MPEG-4 格式影像/聲音資料進行解碼。此外，VideoStatusTimeFunc( )和 AudioStatusTimeFunc( )也同時負責監視多媒體串流系統客戶端的封包接收情形。

上述兩個 thread 在多媒體串流系統客戶端要求連線時產生。

當我們選擇使用 UDP 爲傳送多媒體影音資料的方法時，多媒體串流系統客戶端會爲了影像和聲音，建立下面的兩項執行緒共兩套，包含影像的 RTP 封包使用一套，包含聲音的 RTP 封包使用另一套。

- RecvThreadIsK( )

  負責接收經由 UDP 模式傳送的 RTP 封包。將這些 RTP 封包置於 DePktThreadIsK( )也能同時存取的記憶體區塊中。

- DePktThreadIsK( )

  負責將 RecvThreadIsK( )所接收的 RTP 封包，進行封包拆解。從 RTP 封包中所拆解的內容，進行排列之後，置於 VideoStatusTimeFunc( )和 AudioStatusTimeFunc( )也能同時存取的記憶體區塊中。

如果我們選擇使用 TCP 爲傳送多媒體影音資料的方法，那麼多媒體串流系統客戶端會建立下面一個執行緒。

- TunnelMain( )

  負責接收經由 TCP 傳送的 RTP 封包，並且將 RTP 封包的內容拆解，被拆解的內容將放在 VideoStatusTimeFunc( )和 AudioStatusTimeFunc( )也能同時存取的記憶體區塊中。

4.2.3 RecvThreadIsK( )處理封包機制

RecvThreadIsK( )使用 WINsocket 程式，接收網路卡 memory 中的 RTP 封包。這些封包之後被置於 DePktThreadIsK( )所能存取的記憶體之中，以 link-list 的結構管理，稱作 RTP packet link-list。詳細情形參考 figure.xx。

RecvThreadIsK( )在動態記憶體管理中所扮演的角色是爲接收的每個 RTP 封包皆分配記憶體，加入 link-list 中。

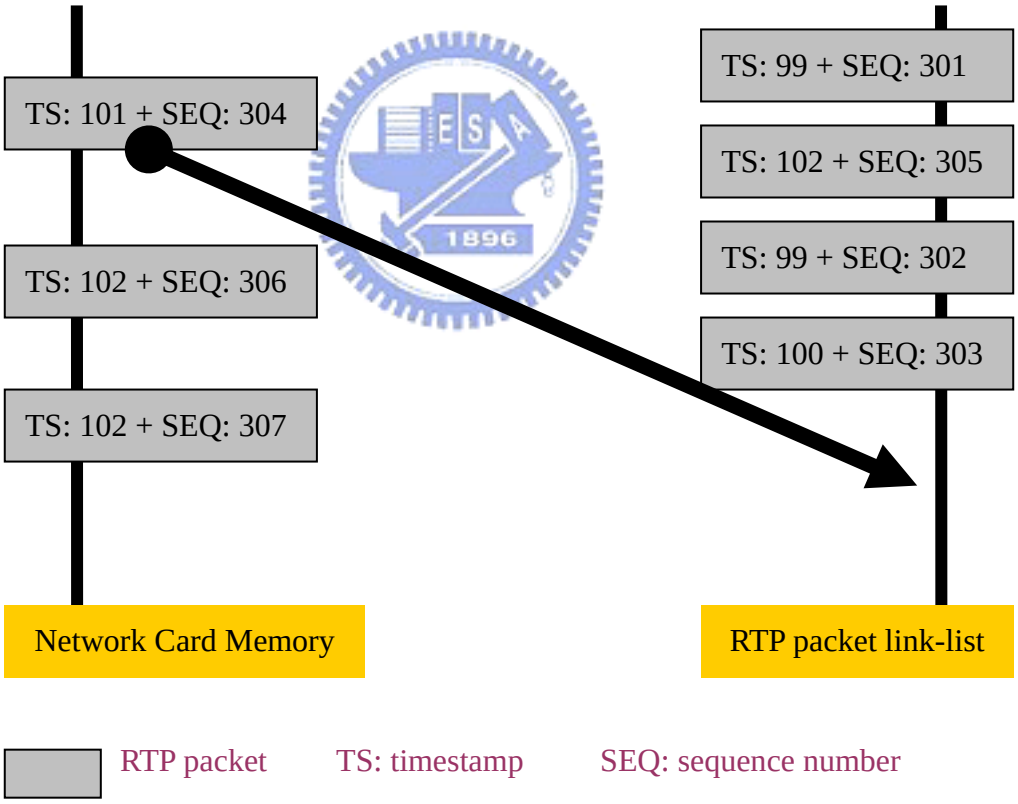

Figure 4.5 - RecvThreadIsK( )接收封包機制

4.2.4 DePktThreadIsK( )處理封包機制

DePktThreadIsK( )和 RecvThreadIsK( )能夠共同存取 RTP packet link-list。DePktThreadIsK( )每一次只讀取一個 RTP 封包，也就是 RTP packet link-list 的 head。

讀取 head 的 RTP header 後，得到 timestamp 和 sequence number。MPEG-4 video frame 的記憶體區塊也是以 link-list 的結構維護，稱作 MPEG-4 video frame link-list。MPEG-4 video frame link-list 根據 timestamp 對 video frame 排序。如果新讀取的 RTP 封包的 timestamp 在 video frame link-list 中找不到，意謂這個 RTP 封包屬於新的 video frame，DePktThreadIsK( )需要動態分配記憶體。如果新讀取的 RTP 封包的 timestamp 可以在 video frame link-list 中找到，代表 RTP 封包所屬的 video frame 已經分配過記憶體區塊，無需再分配一次。

每一個 video frame 所需的空間不同，RTP 封包中必須提供 video frame 所需空間的資訊。然而，因爲 RTP 封包 header 本身的限制，我們在伺服器端進行 video frame packitization 的工作時，在每一個 RTP 封包的 header 之後，payload 之前皆加入了一段特定的 AVI_header。AVI_header 裡儲存有四個欄位，(1)是否加

密 (encrypt)，(2)是否爲I frame(flags_tot)，(3)RTP封包所屬的video frame 的 frame size(chunk size)，(4)RTP 封包在所屬的 video frame 中的相對位置(offset)。AVI_header 能夠幫助我們處理 MPEG-4 video frame link-list 的排序和分配記憶體。

AVI_header 的 chunk size 代表本 RTP 封包所屬的 video frame 在動態分配記憶體時需要多少空間。在 DePktThreadIsK( )分配 video frame 的記憶體空間後，將 RTP 封包中的 payload，也就是真正有關 MPEG-4 video bitstream 的部份複製到 video frame 中。關於這個和 RTP 封包 timestamp 一樣的 video frame，frame 的記憶體大小就是之前分配的大小，而根據 AVI_header 的 offset，可以得知 RTP 封包的 payload，MPEG-4 video bitstream，要複製到這個 frame 裡的相對位移位置。

DePktThreadIsK( )在動態記憶體管理中所扮演的角色釋放已被讀取過的 RTP 封包記憶體區塊，另外將需要新空間的 MPEG-4 video frame 分配新的 video frame 記憶體區塊，加入 link-list 中。

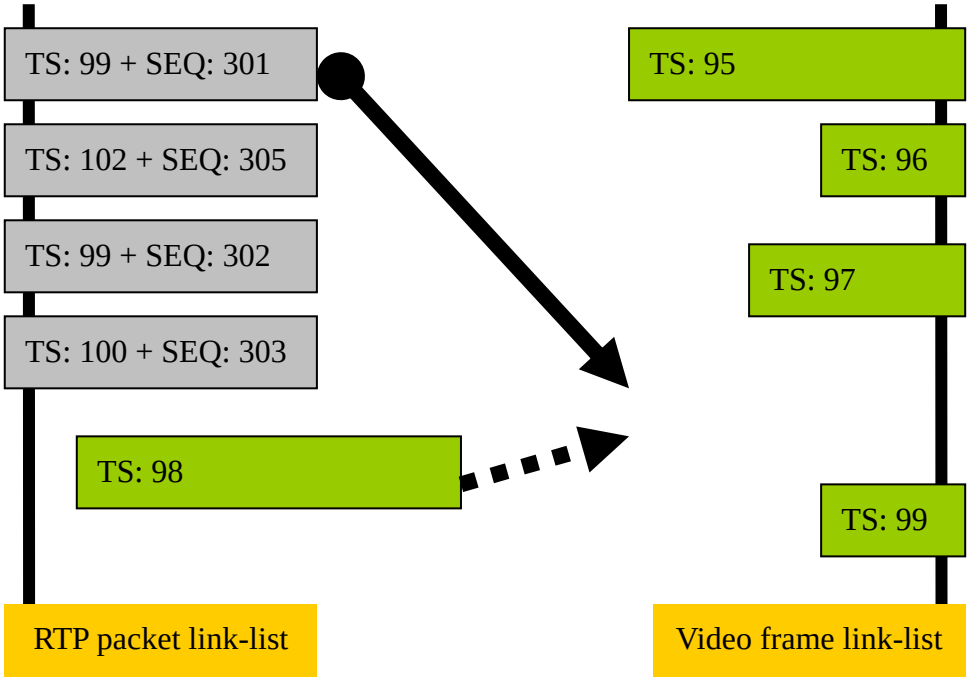

Figure 4.6 – DePktThreadIsK( )處理封包和填塞 frame 機制

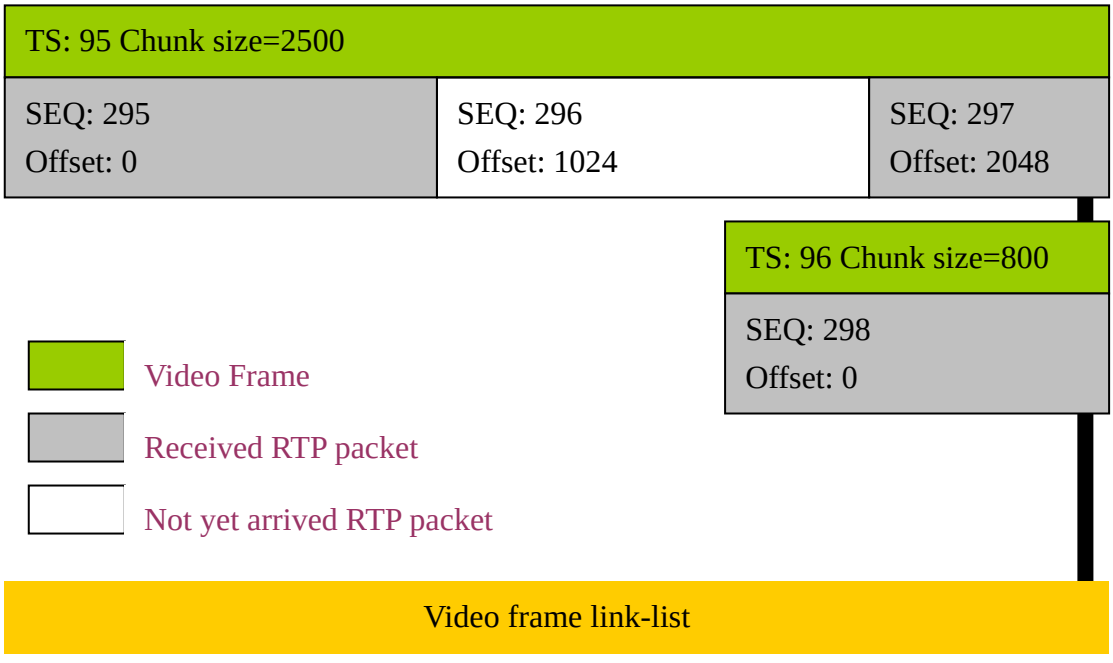

Figure 4.7 – structure of video frame in MPEG-4 video frame link list

4.2.5 VideoStatusTimeFunc( )處理 frame 機制

VideoStatusTimeFunc( )和 DePktThreadIsK( )共同存取存放 mpeg-4 video frame 的記憶體。

VideoStatusTimeFunc( )利用 GetStreamVideo( )每次只取出一張 mpeg-4 video frame，存放在 ucBuffer 暫存記憶體之中，之後將 video frame 交由 SP_decore.lib 進行 mpeg-4 解碼。GetSteramVideo( ) 並不負責檢查 video frame 的 timestamp，對於 1 RTP packet=1 mpeg-4 video frame 的狀況，假設 RTP packet 遺失了，那麼此張 mpeg-4 video frame 也不存在，對於之後的 mpeg-4 decoding 而言，將使用不正確的 reference frame。在 1 RTP packet = 1 mpeg-4 video frame 的狀況下，必須利用 ucBuffer 暫存記憶體的資訊，ucBuffer 中可以取得 timestamp。利用 frame 和 frame 之間的 timestamp 差值，可以了解 mpeg-4 video frame 之間有無 frame loss。這個特性可以克服 1 RTP packet = 1 mpeg-4 video frame 狀況下的 packet loss 情形。

由於 SP_encore.lib 的編碼特性，在 mpeg-4 video frame 並未切割成 video packet 型式(類似於 slice)，因此在 mpeg-4 video bitstream 之中，並沒有 resynchronization codeword 的存在。這導致了 SP_decore 在遇到解碼錯誤時，例如 mpeg-4 video frame

bitstream 中 bit inverse、bit insertion 和 bit deletion 等等錯誤，SP_decore 會放棄解碼之後的 bitstream。

當 SP_decore 完成解碼程序後，SP_decore 會負責將 YUV space 係數轉換爲 RGB space 係數，重建畫面。緊接著是更新 reference frame 的指標。如果 SP_decore 失敗，放棄解碼程序，那麼也無法更新 reference frame。接下來的畫面解碼皆會以「正確 reference frame 的前一張 frame」爲參考畫面進行解碼。詳細情形可以參考 figure xx。在 SP_decore 順利解碼後，通知 GUI thread 顯示重建過後的畫面。

VideoStatusTimeFunc( )在動態記憶體管理中負責釋放 mpeg-4 video frame 的記憶體區塊，維護 mpeg-4 video frame 的 link-list。

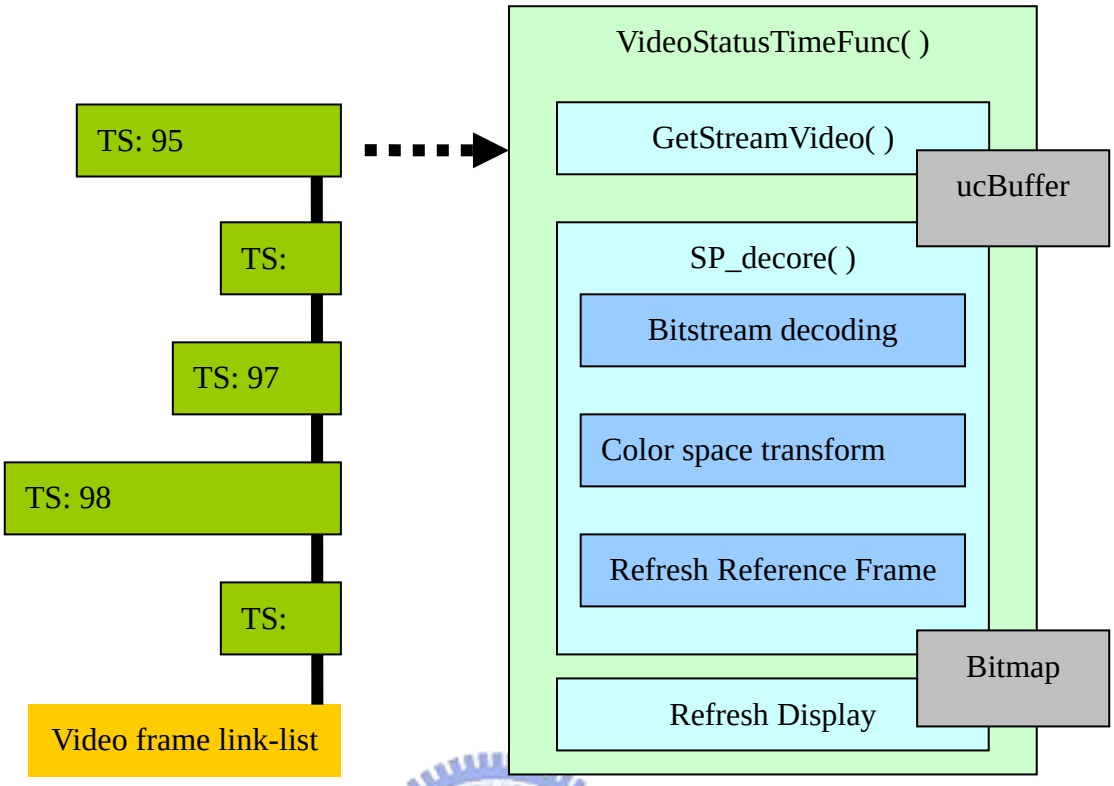

Figure 4.8 – VideoStatusFimteFun( )處理畫面機制

4.3 多媒體串流系統客戶端搭配錯誤隱藏演算法實作細節

實際作法如下：

以 block (8x8) 爲單位，紀錄每一次解碼成功的畫面的 motion vector。將這些 motion vector 存放在兩塊記憶體區塊中。一塊存放 Previous Frame 的 motion vector，簡稱 PreVOP；另一塊存放 2nd Previous Frame 的 motion vector ，簡稱 Pre2VOP。因爲每一次解碼都無法預知當作輸入的 bitstream 是否會造成解碼錯誤，PreVOP 儲存的 motion vector 是源自目前解碼成功的畫面的 motion vector，而

Pre2VOP 儲存的 motion vector 是目前解碼成功的畫面的上一張畫面的 motion vector。以解碼成功的畫面而言，PreVOP 存取倒數第一張畫面的 motion vector，Pre2VOP 存取倒數第二張畫面的 motion vector。

儲存兩個 frame 的 motion vector，是爲了計算 object motion acceleration。Motion vectors in Pre2VOP 和 motion vectors in PreVOP 的差，就是加速度(acceleration)。

new image 的重建是來自於 PreVOP 的投射，PreVOP 的投射必須根基於每個 block 投射的 candidate motion vector for projection。這個 candidate motion vector for projection 就是來自於 Pre2VOP 和 PreVOP 所算出的 acceleration，加上 PreVOP 的 velocity (motion vector)所得出的新 velocity。因此，這個 candidate motion vector for projection 是來自於 2nd-order motion vector extrapolation。

經過 2nd-order motion vector extrapolation 之後，在 PreVOP 中的每個 block 進行投射，在 new image 上形成許多投射區塊。如果畫面的尺寸是 320 pixel x 240 pixel，那麼以 8x8 爲大小的 block，可以在 new image 上投射出 1200 塊區域。在 new image 上的 block 會收到來自 PreVOP 中不同 block 的投射，決定 PreVOP 中哪個 block 能成爲 new image 中的 block 的選擇，憑藉於其投射區塊在 new image 上的

投射面積。以 pixel 爲單位，如果投射區塊 b_1 和 block X 重疊了 24 pixels，那麼 b_1 得到了 24 分；如果投射區塊 b_2 和 block X 重疊了 35 pixels，那麼 b_2 得到了 35 分，而 b_1 出局。這使用了權重因子(weighting factor)的概念。

每個 block 一開始的 weighting factor 設定爲 0，若有被投射，設定權重因子等於投射的面積。如果之後再次被投射，根據權重因子的大小，判定新的投射是否能取代之前的投射。在 new image 中，少數的 block 有可能沒被投射，此時設定它們的 motion vector 爲 0，在進行 motion compensation 之後，對於這些沒被投射的 block 的 YUV/RGB 值直接進行修改，修改的內容是複製(重貼)Pre2VOP 畫面(也就是 2nd previous frame)的相同位置的 block。

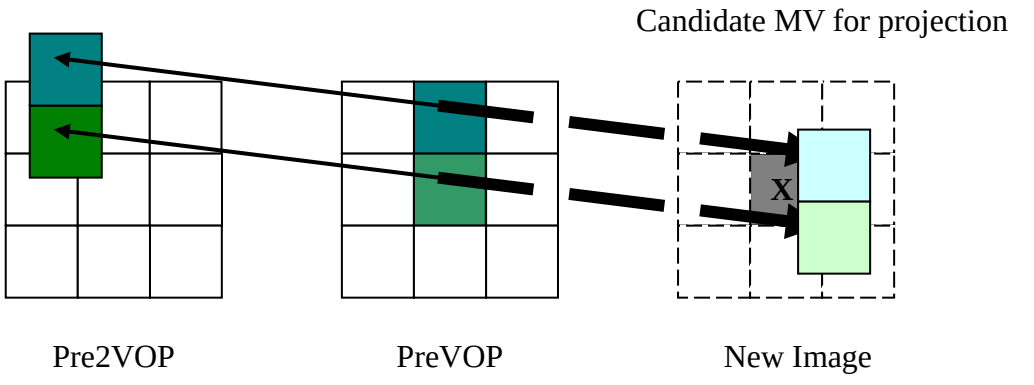

Figure 4.9 – Block X in New Image receiving various forward motion projection from blocks in Previous Frames

new image 中的每個 block 都設定了 motion vector，這時候可以進

行 motion compensation。以 MODE_INTER4V 爲參數，輸入 SP_Decore.lib 中的 motion compensation。這樣可以確保全部以 block 爲單位進行 motion compensation。

由於 bitstream 的錯誤導致 MPEG-4 解碼失敗，我們無法取得 DC/AC 係數，compressed domain error concealment 的畫面重建至此爲止。下一張進行 MPEG-4 解碼的畫面，將以這張重建的畫面爲參考畫面(reference frame)。

4.4 錯誤隱藏演算法中所使用的各項 API

爲了進行錯誤隱藏，VideoStatusTimeFunc( )必須對於內部流程進行新的安排。除了解碼成功時的

- NCTU_error_concealment( ):

進行error concealment的介面函式。首先，它呼叫了 NCTU_mv_extrapolate_v2( )，這支函式能夠進行 2nd-order motion vector外插計算。計算完的數據，將交由NCTU_mv_assign( )函式處理和判斷。之後，NCTU_error_concealment( )呼叫 NCTU_motion_compensation( )，進行motion compensation。最後，NCTU_error_concealment( )呼叫NCTU_write_frame( )將數據以RGB

格式輸出。

- NCTU_mv_extrapolate_v2( ):

進行 2nd-order的motion vector 外插計算(extrapolation)。它計算著Pre2VOP和PreVOP兩個畫面間的motion vector的差值，藉以當作加速度(acceleration)。 此這個加速度爲準，加上PreVOP原本的motion vector，得到投影用的候選motion vector (candidate motion vector for projection)。候選motion vector投射在new image上，得到的x軸和y軸座標，將傳進NCTU_mv_weight_factor( )之中，決定這個候選motion vector的權重因子，也同時決定這個block的motion vector。

- NCTU_mv_weight_factor( ):

藉著 NCTU_mv_extrapolate_v2( )傳入的 motion vector MVx, MVy 和傳入的座標(x, y)，以 block 爲基準，計算每一個 block 所得到的投影面積。這個投影面積等同權重因子，如果 block 之前未被投影過，則採用這次投影的權重因子，並且紀錄目前投影的 motion vector。如果 block 已經被投射過，則比較之前和本次投影的權重因子，如果本次的權重因子大於或等於之前的，那麼紀錄本次投影的 motion vector；反之，保留之前的 motion vector。

- NCTU_mv_assign( ): 在 NCTU_mv_weight_factor 結束後執行。由於 PreVOP 進行的投影，並無法保證在 new image 中的每個 block 皆

能被投射到。在此處除了將之前計算過的投射 motion vector 寫入解碼物件中，還要將未被投影的 block，設定其 predicted motion vector 爲 0，再寫入解碼物件中。

- NCTU_motion_compensation( ): 在完成了 motion vector assignment 之後，以前一張成功解碼的畫面爲參考畫面(reference frame)，進行 motion compensation。而 motion compensation 所使用的模式是以 block 爲單位，而不是常見的以 Macroblock 爲單位。

- NCTU_write_frame( ): 由於沒有辦法取得 DC/AC 等係數，在進行 NCTU_motion_compensation 之後，便將這份資料當作重建後的最後資料。但這份資料是以 YUV 的格式儲存，NCTU_write_frame( )試著把 YUV 格式轉換成 RGB 格式。

- NCTU_mv_refresh( ): 被 VideoStatusTimeFunc( )呼叫，在解碼正確無誤時，進行 motion vector 的紀錄。如果 motion vector 是以 Macroblock 爲單位進行解碼，在 NCTU_mv_refresh( )中，便將其複製成四份，以供四個 block 使用。

- NCTU_avoid_occlusion( ): 在執行完 NCTU_motion_compensation 之後，針對那些在 NCTU_mv_extrapolate_v2( )之中沒有被投射過的

block，進行 YUV value 的處理。由於沒有被投射過的 block 隱含著原來位置的物體離開的意義，被這物體遮蔽的畫面需要用別的方式重建。這裡選用的是以前二張畫面的相同位置 block 取代。

//

VideoStatusTimeFunc( ){
…
result = **StreamGetVideoFrame(…)**;
if(result==RESULT_PACKET_LOST_FRAME_ERROR)
NCTU_error_concealment(…);
else
decore(…);
…
NCTU_buffer_swap(…);
NCTU_mv_refresh(…);
}

Table 4.1 – overview of VideoStatusTimeFunc( )

NCTU_error_concealment( ){
…
NCTU_mv_extrapolate_v2(…);
NCTU_mv_assign(…);
…
for(each block)

NCTU_motion_compensation(…);
NCTU_avoid_occlusion(…);
NCTU_write_frame(…);
}

Table 4.2 – overview of NCTU_error_concealment( )

NCTU_mv_extrapolate_v2( ){
int predict_mvx, predict_mvy, prev_mvx, prev_mvy, pre2_mvx, pre2_mvy;
int predict_x, predict_y, pre2_x, pre2_y;
int acce_x, acce_y;
…
for(each block){
由 **PreVOP** 出發，根據 **motion vector**，往 **Pre2VOP** 尋找對應的 **block**
得到 **pre2_x, pre2_y**，得到 **pre2_mvx, pre2_mvy**。
得到 **acce_x, acce_y**。
得到 **predict_mvx, predict_mvy**。
得到 **predict_x, predict_y**。
NCTU_mv_weight_factor(…);
}
}

Table 4.3 – overview of NCTU_mv_extrapolate_v2( )

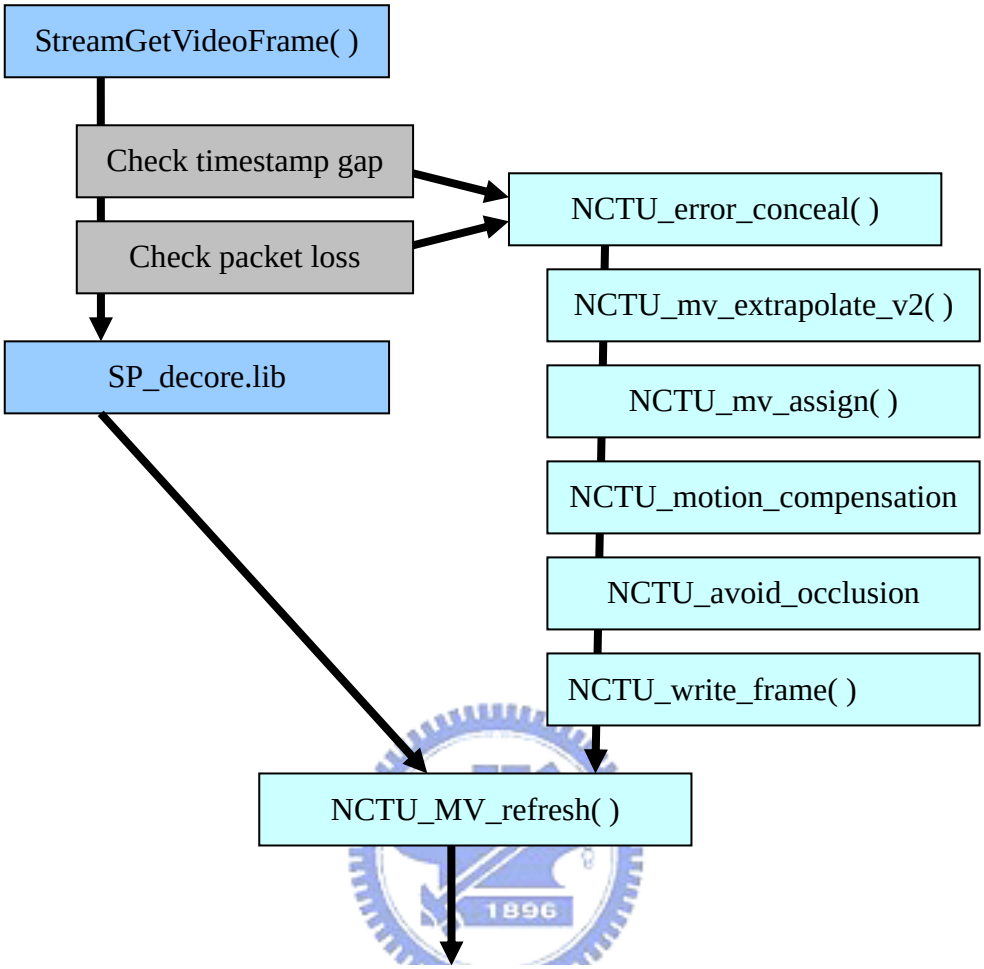

Figure 4.10 – modified flow in VideoStatusTimeFunc( ) of Error Concealment version

Chapter 5 Experimental Results

5.1 和 1st-order motion field image processing 第一次比較

Figure 5.1 – previous frame

Figure 5.2 – new image with 2^{nd}-order motion field image processing

Figure 5.3 – new image with 1^{st}-order motion field image processing

可以見到 2nd-order motion field image processing 在處理 motion vector 時有較好的預測，能夠得到比較好的畫面品質。

5.2 和 1st-order motion field image processing 第二次比較

Figure 5.4 – 2nd previous frame

Figure 5.5 – previous frame

左邊的畫面是 2nd previous frame，右邊的畫面是pervious frame。

Figure 5.6 – new image with 1st-order motion field image processing

Figure 5.7 – new image with 2nd-order motion field image processing

左邊的畫面是以 1st-order motion field image processing 處理，右邊的畫面是以 2nd-order motion field image processing 處理，可以發現兩張修補過的圖在品質已經很接近。但我們仍可觀察到左圖裡女生的

右手週圍稍有品質下降情形。

5.3 在接近靜態時與原圖比較

Figure 5.8 – original correct image

Figure 5.9 – new image

可以觀察到，兩圖幾乎沒有差別。

5.4 和不進行錯誤隱藏的照片比較

在進行了手動丟除 frame 的行動之後，比較「進行錯誤隱藏演算法」和「不進行錯誤隱藏演算」的比較。

Figure 5.10 - Previous frame

Figure 5.11 - New image (2^{nd}-order motion field image processing)

Figure 5.12 – #A_EC, referencing to new image
Figure 5.13 – #A_NEC, referencing to previous frame

左圖是在 new image 之後的後續照片#A_EC，右圖是不使用 error concealment 而直接參考 previous frame 的照片#A_NEC。

Figure 5.14 – picture referencing to #A_EC

Figure 5.15 – picture reference to #A_NEC

左圖是參考#A_EC 的後續照片，右圖是參考#A_NEC 的後續照片。

可以觀察到，當 new image 和 previous frame 得到的結果很相近時，他們後續的畫面之間並不存在明顯差異。甚至可以得到「進行 error concealment」是多餘的影像處理步驟之類的結論。

5.5 單純看錯誤隱藏演算法的表現

Figure 5.16 – previous frame

Figure 5.17 – new image

左圖是 previous frame，右圖是經過 2nd-order motion field image processing 的重建畫面。

Figure 5.18 – next frame

Figure 5.19 – 2nd next frame

左圖是在new image後第一張畫面next frame，右圖是在new image後第二張畫面 2nd next frame。觀察畫面發現，2nd next frame的motion vector指向的next frame區域大部份都不符合 2nd next frame的期待。又觀察next frame和new image之間的相似度極高，和next frame有關的推論可以適用於new image。所以，我可以大膽推論new image並不符合new image後續畫面的期待，也就是說，new image以一種可以接受的畫面品質呈現，但是這種畫面品質卻無法提升後續畫面的品質，這也是錯誤隱藏演算法的一種缺陷。

以下附上 new image, next frame 和 2nd next frame 的原圖

Figure 5.20 – new image 位置原圖

Figure 5.21 – next frame 位置原圖

Figure 5.22 – 2nd next frame 位置原圖

Chap 6 Conclusion

隨著無線網路設備的產品售價降低和易於安裝，越來越多家庭和小型辦公室使用無線網路設備上網。由於在無線網路的環境下，通道噪音極容易造成封包遺失或是封包延遲到達，以 UDP 爲傳輸協定的應用服務必然遭受服務品質下降的威脅。

多媒體串流系統的下層傳輸協定採用 UDP，在面對通道噪音時，使用者端必然會感受到聲音和畫面的品質降低。本篇論文提出的演算法，以類似於追蹤物體的想法，進一步地在利用 compressed domain 的 motion vector 資訊。進行了二階 motion vector 外插後，再嘗試避免 occlusion effect。

多媒體串流系統強調即時性，應用於多媒體串流系統的 error concealment algorithm 必須能即時地而且有效率地對於整張畫面進行彌補。根基於 pixel domain 的 error concealment algorithm 在多媒體串流系統方面有其局限性，演算法的運算量過於龐大，耗費的計算時間過久。

2^{nd}-order motion filed image processing的計算量和解碼一張正確影像相比差距不大，以 2^{nd}-order motion field image processing 進行錯

誤隱藏演算法對於客戶端而言是可以接受的。根據實驗結果發現，2nd-order motion field image processing比起單純 1st-order motion field image processing，能得到較好的影像品質。在可以實現的情況中，2nd-order motion field image processing是值得採用的作法。

在網際網路頻寬仍嫌不足的現在，許多以提供多媒體串流服務的網站，它們提供的內容以低解析度低品質的影片爲主。使用 compressed domain error concealment algorithm 而得到的彌補畫面，品質和這些網站所提供的影片品質十分接近。Compressed domain error concealment algorithm 的問題出在於無法修補 DCT DC/AC 係數，彌補過的畫面沒有細節，blocking effect 稍嫌明顯。由於 post-processing error concealment 本身的缺憾，在修補過的畫面中仍有 occlusion effect 的存在。來自於 previous frame 和 2nd previous frame 的 motion vector 無法有意義地傳達物體的移動軌跡，motion vector 的值只是單純的 motion estimation 計算結果，error concealment in compressed domain 的這些缺點還需要更進一步的影像處理技術和影像內容理解。

Reference List

[1] Iain E. G. Richardson, “H.264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia”, Wiley, 2003.

[2] H. Schulzrinne, S. Sasner, R. Frederick, and V. Jacobson, „RTP: A Transport Protocol for Real-Time Applications“, Audio Visual Working Group Request for Comment RFC 3550, IETF, July 2003.

[3] Y. Wang and Q. F. Zhu, “Error Control and Concealment for Video Communication: A review”, Proc. IEEE, vol.86, pp.974-994, May 1998

[4] “Information technology – Generic coding of moving pictures and associated audio information – Part 2: Video”, ISO/IEC DIS 13818-2, 1994.

[5] J. K. Wof, A. Wyner, and J. Ziv, “Source coding for multiple descriptions”, Bell Syst. Tech. J., vol. 59, pp. 1417-1429, Oct. 1980.

[6] J. C. Maxted and J. P. Robinson, “Error recovery for variable length codes”, IEEE Trans. Inform. Theory, vol. IT-31, pp. 794-801, 1985.

[7] Y. Wang, Q. F. Zhu, and L. Shaw, “Maximally smooth image recovery in transform coding“, IEEE Trans. Commun., vol.41, pp. 1544-1551, Oct. 1993.

[8] H. Sun and W. Kwok, “Concealment of damaged block transform coded images using projections onto convex sets”, IEEE Trans. Image Processing, vol.4, pp. 470-477, Apr. 1995.

[9] S. S. Hemami and T. H.-Y. Meng, “Transform coded iamge reconstruction exploiting interblock correlation”, IEEE Trans. Image Processing, vol.4, pp.1023-1027, July 1995.

[10] Q. F. Zhu, Y. Wang, and L. Shaw, “Coding and cell loss recovery for DCT-based packet video”, IEEE Trans. Circuits Syst. Video Technol., vol.3, pp.248-258, June 1993.

[11] W. Wada, “selective recovery of video packet loss using error concealment”, IEEE J. Select. Areas Commun., vol. COM-32, pp.280-287, Mar. 1984.

[12] “Terminal for low bitrate multimedia communication”, International Telecommunications Union, Geneva, Switzerland, ITU-T Draft Recommendation H.324, Dec. 1997.

[13] Q. F. Zhu, “Device and method of signal loss recovery for real-time and/or interactive communications”, U.S. Patent 5 550 847, Aug. 1996.

[14] Q. Peng, T. W. Yang and C. Q. Zhu, “Block-Based Temporal Error Concealment for Video Packet Using Motion Vector Extrapolation”, IEEE 2002

作者簡歷

邱培哲，民國七十一年生於台北縣。民國九十三年畢業於國立台灣大學電機工程學系，同年進入國立交通大學電信研究所，從事影音多媒體相關研究。民國九十五年取得碩士學位，論文題目是「壓縮領域中的錯誤隱藏演算法」。研究興趣是 C/C++程式開發。