國 立 交 通 大 學

電信工程學系

碩 士 論 文

串流平台語音封包遺失的處理

Speech Processing for Packet loss in a Streaming System

研究生　　：　溫志中

指導教授：　　張文鐘 博士

中　華　民　國　95 年　9 月

串流平台語音封包遺失的處理

研究生：溫志中　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

由於互連網路的發達，在網路上即時傳送影像及聲音已經成為了一種趨勢。串流技術(streaming)就是這種一邊下載一邊播放的技術。網路傳輸訊號時難免會有遺失或是錯誤。對「先下載完再播放」的傳統技術而言，遺失握錯誤的封包，只需重新傳輸即可。但是串流技術為了因應即時播放的邀求，沒有時間重新傳輸遺失或錯誤的封包。在聲音方面，過去有人提出用重複上一個收到的片段來填補遺失或錯誤片段的方法。後來有人提出樣本比對(pattern matching)來改進它。本文作者利用 mp3 或 aac 聲音壓縮分頻段的特性，做多頻段樣本比對以填補遺失或錯誤的片段，並將重建的信號與原始訊號比對。結果顯示，多頻段樣本比對在低雜訊的情形下比不上舊有的樣本比對，但是在較高 color noise 時有較佳的效果。

Speech Processing for Packet loss in a Streaming System

Student: Chih-chung Wen

Advisor: Dr. Wen-Thong Chang

The Department of Communication Engineering

National Chiao Tung University

Abstract

It has been a tending to real-time transmit audio-video signals on Internet. Streaming is the technology of playing while downloading. The packet loss and crack are unavoidable in Internet transmission. For the conventional technology of play-after-download, the lost or cracked packets can simply retransmitted. However, there is no time to retransmit them in streaming. In the audio area, it was proposed to replace them by the last received frame. It was then proposed the method of pattern matching to improve. In this thesis, it was proposed further the multi-band pattern matching to make use the multi-band compression of the mp3 and aac audio. The reconstructed signals was then compared wth the original signals. It was found that the results of multi-band patt\ern matching are worse the those of the "old" pattern matching when the noise levels are low, themulti-band pattern matching has better when the color noise are higher.

謝誌

我能夠順利拿到碩士學位，要感謝我的老師張文鐘教授的指導，張教授在我碩士班的學習中指導我，讓我能夠完成研究。感謝在一起學習的學長、同學以及學弟們在生活中的鼓勵。感謝我兩個姊姊的支持與愛護。最後感謝我的父母，謝謝你們。

目錄

圖列表

表列表

1. 簡介

1.1 背景

網路時代以來，人們可以輕易的從網路上取得大量的影片與音樂，同時也佔據了大量的頻寬與儲存空間，造成了巨大的負擔，網路壅塞，儲存空間不足。所以我們會將資料壓縮，使其變得比較小，來減少頻寬與儲存空間的負荷。傳統上，我們會將音樂或影片完整的下載之後，才能夠觀看，在點選到觀看之間，往往需要大量的時間。串流技術的發展，可以不需要儲存空間，就可以觀看影片及收聽音樂，更可以做即時的觀看，不需要先將影片放在伺服器上，而是將即時的聲音、影像做立即的壓縮，傳輸給用戶觀看。為了傳輸這樣的即時資料，舊有的 TCP 顯得過於複雜，而 UDP 則過於簡略，所以 Internet Engineering Task Force (IETF) 在 RFC-3550 中定義了 Real Time Protocol (RTP)[1]來輔助過於簡略的 UDP。

聲音壓縮分為非失真壓縮與失真壓縮兩種，非失真壓縮不會對聲音造成破壞，但是壓縮比有限，所以應用層面不廣。失真壓縮則會丟棄部份資料，所以聲音就不會是原來的樣子，不過壓縮比相當的高，因此各方面都廣為應用。ISO/IEC 11172-3 中定義的 mpeg-1 audio layer3 (mp3)[2]就是一種失真壓縮格式，在這篇論文中利用 mp3 來講解失真壓縮的演算法。ISO/IEC 14496-3 中定義的 Advanced Audio Coding (AAC)[3]則是 mp3 的改良標準。

在串流的過程中，難免會有封包遺失或延遲，特別是現在的無線網路發達，這種情形更為常見。由於串流傳輸即時的特性，在封包遺失時不會有時間重新傳送，單純的封包延遲也可能會趕不上播放，而造成播放的空白。人對這樣的空白非常的敏感，所以需要有機制來處理這個問題，讓人耳感受不到問題。

由於一個封包常常攜帶了數百個甚至上千個取樣，所以普通的內插無法處理這麼長的遺失，必須有其他機制來處理，如說經由錯誤更正碼來恢復遺失或錯誤的部份，像是 Waveform Similarity Overlap-Add(WSOLA)[4][5]，不過這會增加額外的負擔。在不加上任何額外的負擔上面，由於聲音信號在短時間內是類似的波型重複出現，所以常常會直接重複上一個片段[6]。不過敏感的人可能會感覺到聲音重複了一下，所以有人提出了樣本比對(pattern matching)[7]的方法，藉由搜尋之前、或是之後收到的聲音信號，來推測這一段最可能跟之前或之後哪一段相似，來修補遺失的部份。樣本比對也是一種不需要經過任何訓練、所有人的聲音都能夠採用的機制。在[8]中，先將訊號轉換到頻域，然後在頻域上執行樣本比對。[9]則將修補的信號與收到的信號做部份重疊，以試圖解決樣本比對在邊界上產生的不連續現象。我們則想到說，聲音壓縮時將聲音分為許多的頻帶，那麼比對時如果只比對其中一些頻帶，可能會有更好的效果。

1.2 動機

網路傳輸難免會有封包遺失或損毀，特別是在無線網路的情形，所以需要有修復的機制。樣本比對是不錯的機制，如何改進它就成為我們的課題。

1.3 研究目標

- 在之前已經建立的平台上加入聲音抓取的功能
- 在傳輸封包遺失時進行處理
- 樣本比對的改良。

1.4 大綱

在這篇論文的第二章，我將會介紹串流所使用的 RTP 傳輸協定。第三章是聲音壓縮。修補遺失的聲音的方法則在第四章加以討論。第五章則是我們的系統的架構與實作的介紹。執行的結果則放在第六章。

2. 串流技術

在這章中,將會討論關於串流技術的設計與實作,包括了 Real-time Transport Protocol (RTP)與 RTP Control Protocol (RTCP)

2.1 Real-time Transport Protocol (RTP)

我們在傳輸即時的多媒體資料時，假如使用 TCP 的話，將會過於複雜。TCP 具有強大的重傳機制以保證封包的到達，不過同時封包的延遲也會是不可預期的數字，對即時的多媒體資料來說，過期的資料不具任何意義，只是徒增網路的負擔。

而假如使用簡單的 UDP 來傳送封包的話，則因 UDP 太過簡略，甚至連封包的序號都沒有，所以需要一個輔助的傳輸協定，也就是 RTP。

RTP 是為了輔助 UDP 的功能不足而設計的，他具有所沒有的封包序號，不同頻道的傳輸，以及標頭的錯誤檢測。RTP 是專門設計來傳輸即時的多媒體資料，例如說聲音及影像，所以具有許多特別設計的機制。RTP 具有資料內容的分別，封包序號，時間標記及 QoS 的監測。詳情會在下面敘述。RTCP 則是負責監測 RTP 的傳輸品質及監測各個傳輸參數的傳輸協定。

RTP 提供了適合即時影音資料傳送的點對點功能，適用於廣播或是單一播送。多媒體資料會被放在一個或多個 RTP 封包裡，然後再包成 UDP 封包傳送。

2.1.1 RTP 封包格式

Header	Payload

圖 1 RTP 封包格式

RTP 封包是標頭與負載所組成，負載可以是聲音或影像等等。

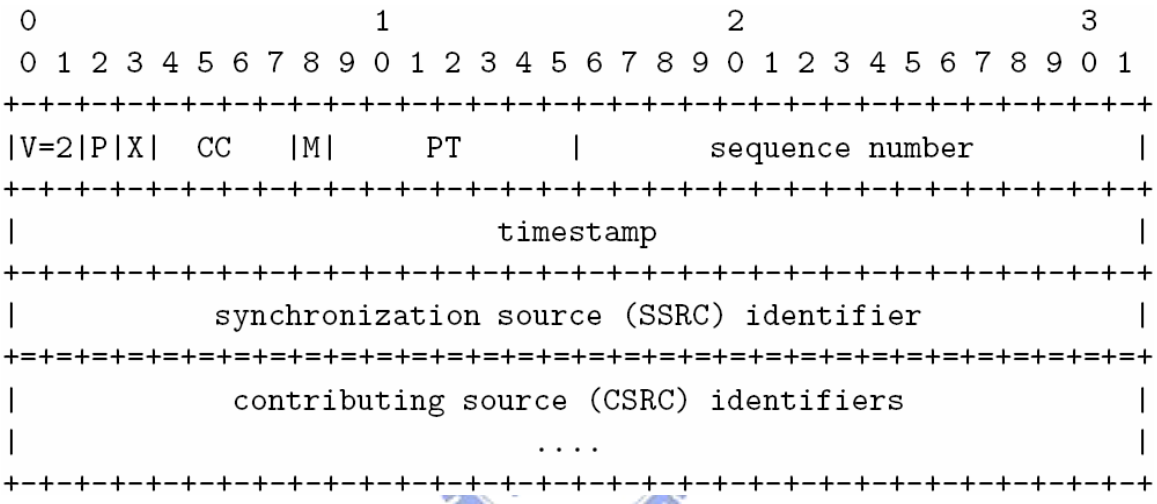


圖 2 RTP 標頭[1]

其中各個欄位定義如下

- Version (V): 2 bits

  這表示 RTP 的版本，RFC3550 中定義的 RTP 版本，此值為 2

- Padding (P): 1 bit

這個欄位表示在封包的最後有沒有填補數量用的位元組，在某些狀況下封包長度不能是任意值的時候，需要加上填補用的位元組。填補用的位元組中最後一個位元組表示了總共有多少位元組需要被忽略。

- Extension (X): 1 bit

這個位元表示了標頭中有沒有用到延伸的旗標。

- CSRC count (CC): 4 bits

Contributing source (CSRC)表示了不同來源的串流。不同來源的串流可以經過

混合器，將多個 RTP 封包重新組成，此時就要靠 CSRC 來進行分別。

- Marker (M): 1 bit

  這個旗標用來表示說有些事件發生了，例如說一個 frame 的邊緣可能就用 marker 來分別。

- Payload type (PT): 7 bits

  用來表示負載的種類。

- Sequence number: 16 bits

  封包的序號，隨著送出封包而增加。一開始先隨機選定一定數字當起始，之後每送出一個封包就加 1。可以用來偵測封包的遺失或是錯序。

- Timestamp: 32 bits

  這表示這個封包所裝的內容所應該被播放的時間。可以用在影音同步，以及播放速度的控制上。在我們的系統中傳輸聲音的部份，每個封包攜帶 1024 個聲音取樣，所以每個封包增加 1024。

- SSRC: 32 bits

  這個 RTP 串流得編號，用來分別哪些風標視同一個串流的。在初始時隨機選定。

- CSRC list: 0 到 15 個，每個 32bits

  這個 RTP 串流中共有哪些 CSRC 的列表。

2.1.2 RTP 範例

在這個範例中，A,B,C 都有各自的 SSRC，然後 A,B 經過混合器合成一個新的 RTP 串流，具有新的 SSRC，A,B 的 SSR 則成為新的 RTP 封包的 CSRC。

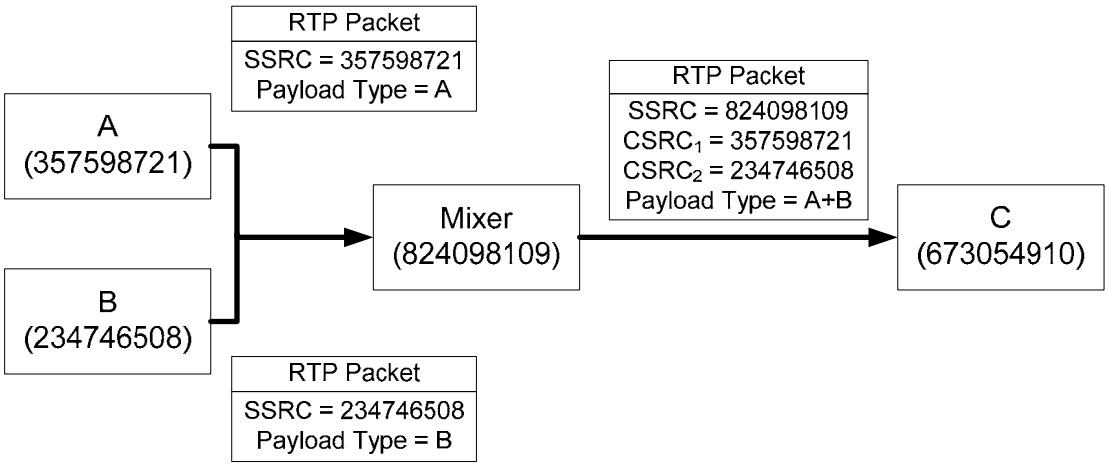


圖 3 RTP 範例

2.2 RTCP

RTCP 負責輔助 RTP，他主要有三個功能

- 監視並回報網路狀況給發送端與接收端
- 使用 canonical name (CNAME) 來描述負載的內容，避免發送端重新啟動造成的 SSRC 改變而無法判斷是同一內容。
- 幫助發送端與接收端調整時鐘差異

2.2.1 RTCP 封包形態

RTCP 具有五種封包形態:

- Sender report(SR):

  讓發送端傳送傳輸與接收的統計資料。

- Receiver report (RR):

  讓接收端傳送傳輸與接收的統計資料，包括封包掉落率、收到的最大封包序號，。

- SDES:

  對來源的描述，其中包含 CNAME

- BYE:

表示結束

- APP:

  由程式自行定義功能

2.2.2 RTCP Sending Rules

為了減少頻寬使用，RTCP 允許一個封包放入多個訊息，成為一個合併的封包，因此每個訊息都要結束在 32 位元的邊緣以方便處理。其中有兩個限制如下:

- 每個合併的封包都要有回報的訊息(RR 或是 SR)，且放在封包的最前面。
- 每個合併的封包都要有包含 CNAME 的 SDES，以讓新的接收者能夠分辨來源以及同步，或是在 SSRC 因為程式重新啟動等原因而變更時，能夠保持連線。

圖似是一個標準的合併的封包的範例，其中包含了 RR，SDES 跟 BYE

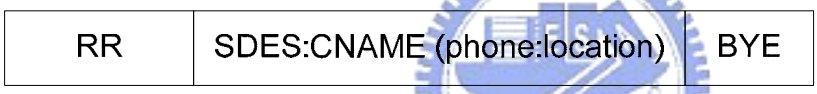


圖 4 標準的合併封包[1]

2.2.3 RTCP 封包格式

Sender Report (SR)

SR 用來提供接收品質的回報以及提供者的資料。封包格式請參考圖五:

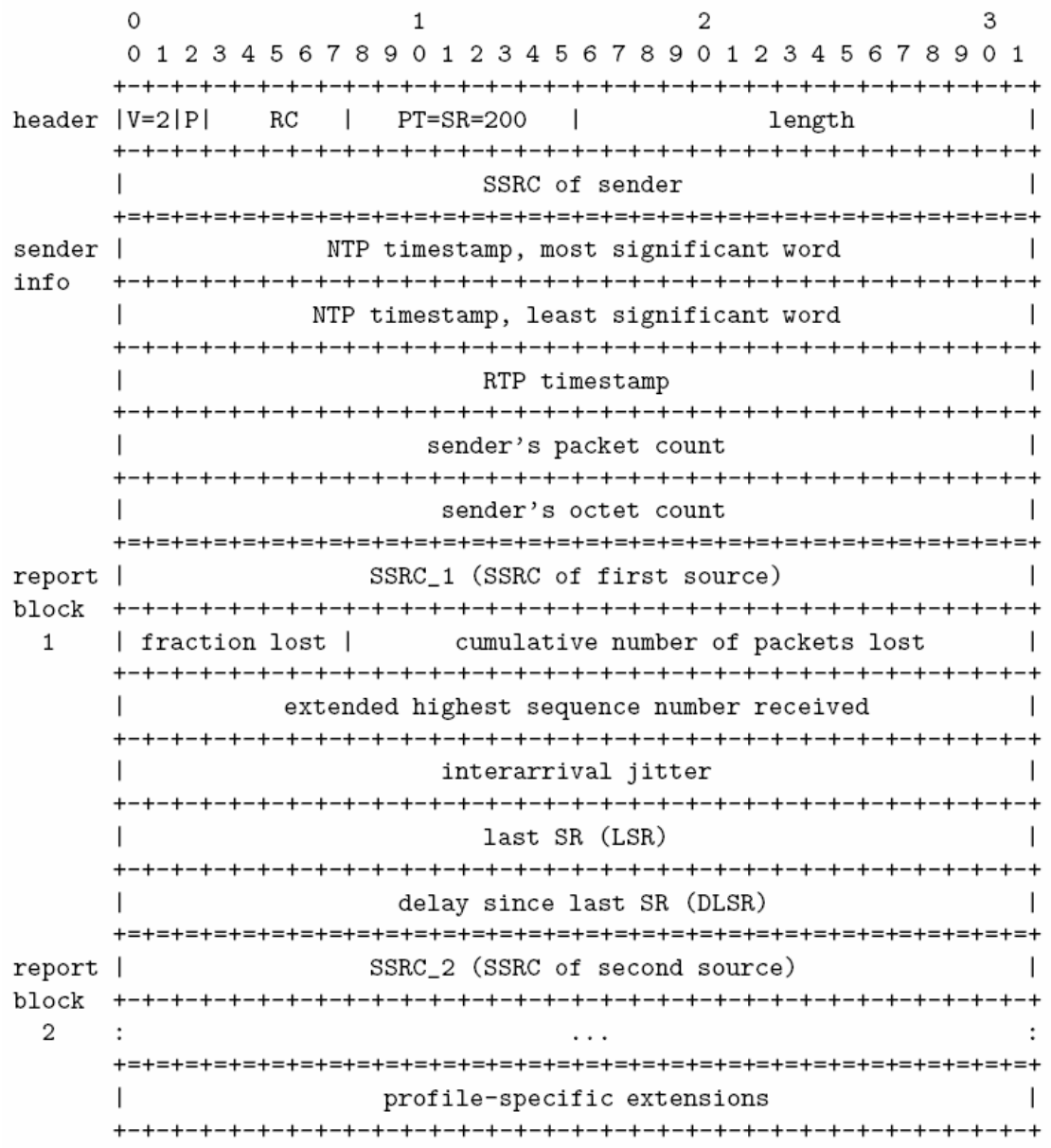


圖 5 SR 封包格式[1]

- Version (V): 2 bits

  表示 RTP 的版本，2 表示是 RFC3550 的 RTP。

- Padding (P): 1 bit

  表示 RTCP 封包最後有沒有填充的位元。

- Reception Report Count (RC): 5 bits

表示在這個合併的 RTCP 封包中，SR 佔了多少個區塊。

- Packet Type (PT): 8 bits

  SR應該要設成200。

- Length: 16 bits

  表示這個RTCP封包的長度。

- SSRC: 32 bits

  表示來源的 SSRC

  接下來的發送者資訊(sender information)區塊包含以下來欄位

- NTP Timestamp: 64 bits

  網路時間協定Network Time Protocol (NTP))格式，表示此封包發送的時間，為從國際標準時間1990年1月1日0:00以來經過的秒數。

- RTP Timestamp: 32 bits

  跟NTP Timestamp同樣是表示時間，不過跟RTP封包裡的timestamp具有同樣的隨機偏移，可以在發送端與接收端的NTP timestamp不同不實作同步之用。

- Sender's Packet Count: 32 bits

  表示從開始到現在一共送了多少各RTP封包

- Sender's Octet Count: 32 bits

  表示從開始到現在RTP封包一共裝了多少位元組的負載。

  第三個區塊包含了0個或更多個回報，各欄位如下:

- SSRC_n (source identifier): 32 bits

  表示這個回報來源的SSRC。

- Fraction Lost: 8 bits

  表示上個SR或RR之後有多少RTP封包遺失。

- Cumulative Number of Packets Lost: 24 bits

表示開始到現在共遺失多少RTP封包

- Extended Highest Sequence Number Received: 32 bits

  前16位元表示從SSRC_n收到的最高的序號，後16位元表示這個序號已經歸零幾次。

- Interarrival Jitter: 32 bits

  這表示了RTP封包的到達間隔時間的變異量的統計。

- Last SR timestamp (LSR): 32 bits

  從SSRC_n來的最新的SR的timestamp。

- Delay since last SR (DLSR): 32 bits

  收到上一個SR到RR發送的時間的間隔。從圖六中可以看出，可以藉由DLSR算出封包來回需要的時間RTT = (S_Clock - LSR) - DLSR，S_Clock是指RR收到的時間。

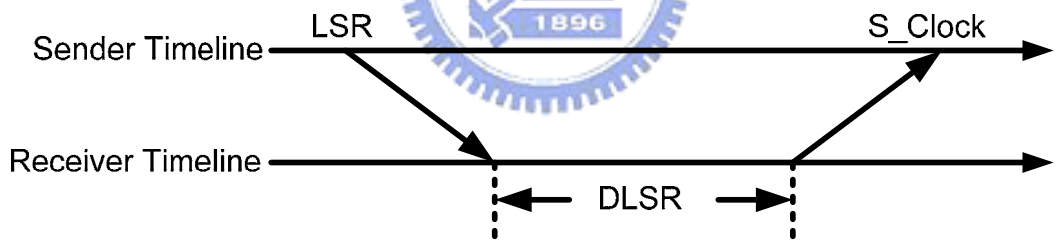


圖 6 計算 RTT

Receiver Report

RR 的目的跟 SR 相同。在各個欄位上都與跟 SR 相同，不過封包格式是 201，而且沒有包含發送者資訊。

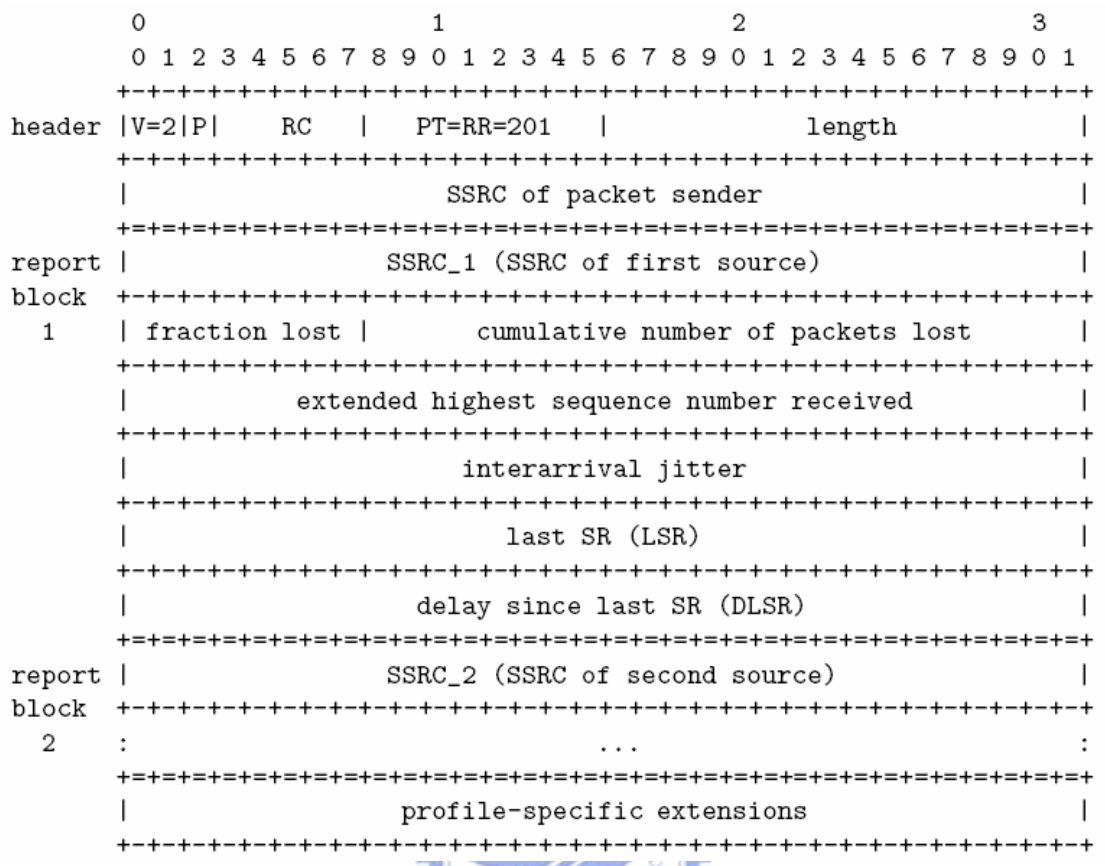


圖 7 RR 封包格式[1]

SDES

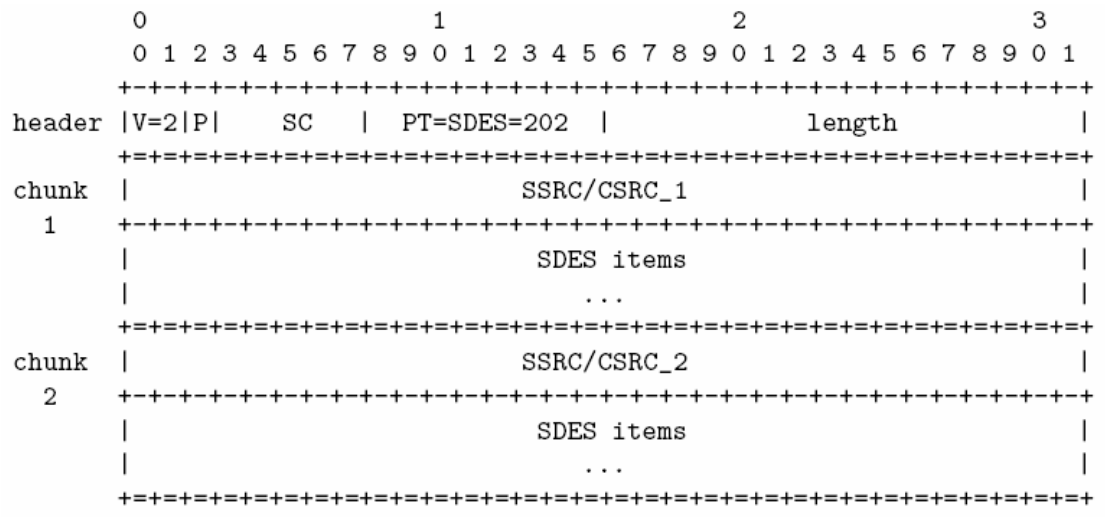


圖 8 SDES 封包格式[1]

圖 8 是 SDES 封包格式。SDES 有標頭跟 0 個以上的 chunk，每個 chunk 都包含了關於物件的描述

- Version (V), Padding (P), Length:

  這三個跟SR中的定義相同

- Packet Type (PT): 8 bits

  SDES應該設為202

- Source count (SC): 5 bits

  表示有多少個chunk在裡面。

  有八種 SDES 物件，每個都有形態跟長度，以及帶的資料

  - CNAME: 標準的辨識物件

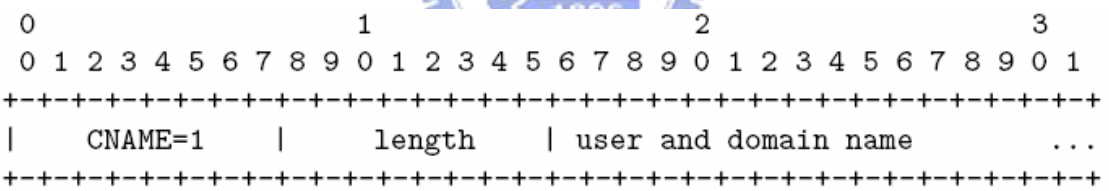


圖 9 CNAME 格式[1]

CNAME 物件格式如圖 9。他用來辨識來源以備 SSRC 因為各種原因而改變的不時之需，所以同一個連線的 CNAME 不該變動。每一個合併的 RTCP 封包都應該要有這個物件>

CNAME可以是”cm9213642@nctu.edu.tw”的樣子，cm9213642 表示使用者名稱，nctu.edu.tw表示主機。

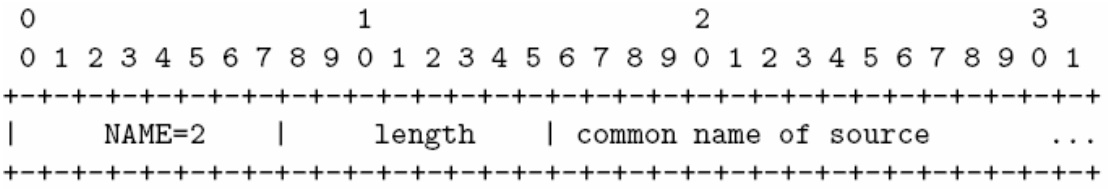


圖 10 NAME 物件格式[1]

■ NAME: 使用者名稱 SDES 物件

NAME 的格式如圖 10，他包含了來源的真實名稱，例如”CM_NCTU”

■ EMAIL: 電子郵件SDES格式

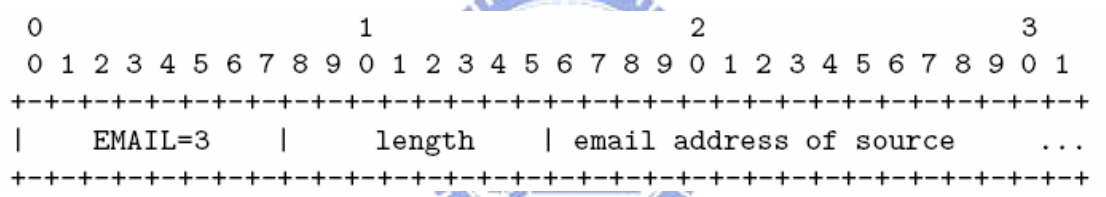


圖 11 EMAIL 物件格式[1]

EMAIL物件格式如圖11，他包玝了來源的電子郵件地址，例

如”fanatic@cm.nctu.edu.tw”

■ PHONE: 電話號碼SDES物件

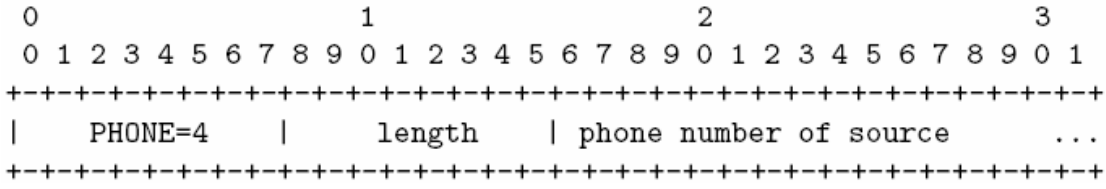


圖 12 PHONE 物件格式[1]

PHONE 物件格式如圖 12，他為來源的電話號碼，例如”035712121”

- LOC: 使用者的地理位置物件

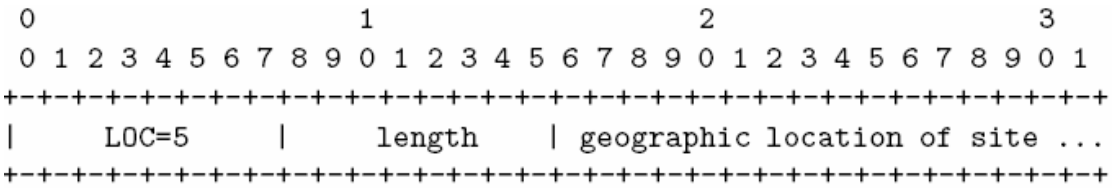


圖 13 LOC 物件格式[1]

LOC 物件格式如圖 13，他為使用者所在位置。

- TOOL: 應用程式或是工具 SDES 物件

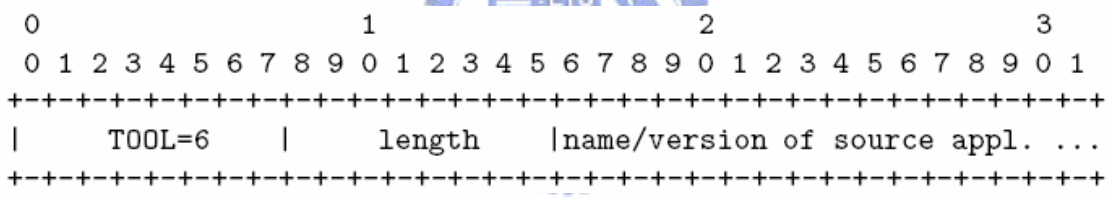


圖 14 TOOL 物件格式[1]

TOOL 物件格式如圖 14，為發送這個 RTCP 封包的程式的名稱，例如”skype”

- NOTE: 提醒/狀態SDES物件

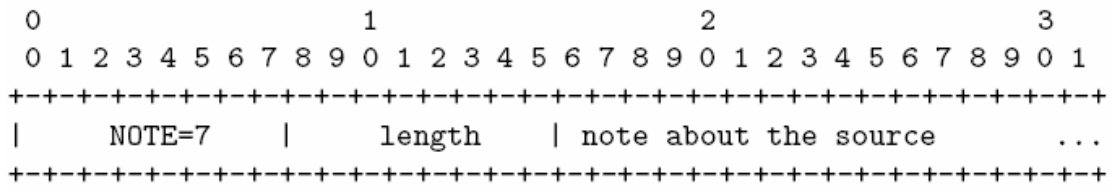


圖 15 NOTE 物件格式[1]

NOTE物件格式如圖15，他用來描述目前來源的狀態，例如說”忙碌”” 閒置”

- PRIV: 私有延伸 SDES 物件

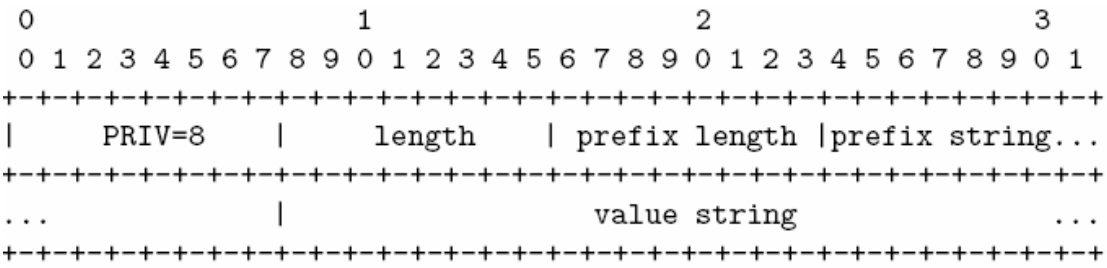


圖 16 PRIV 物件格式[1]

PRIV 物件格式如圖 16，他用來傳輸由程式自行定義的 SDES 功能。Prefix 用來便是使用者要求的功能，而 prefix length 為 prefix 長度。

BYE

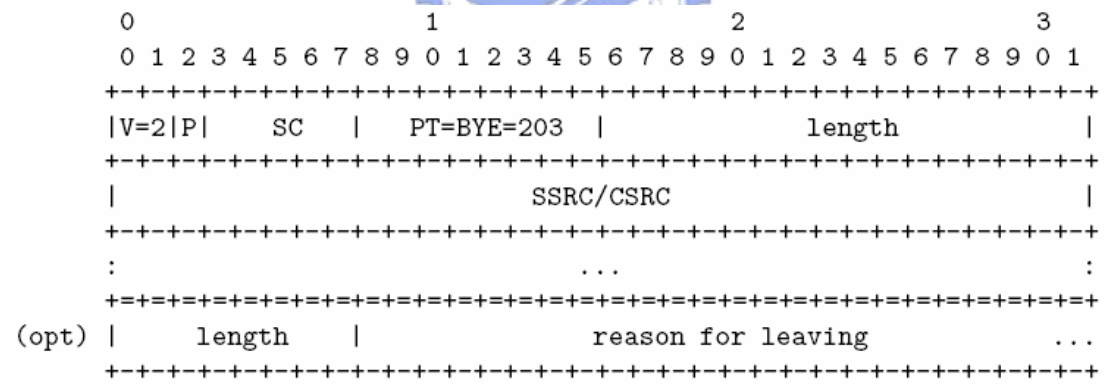


圖 17BYE 格式[1]

BYE RTCP 封包用來終結連線，他包含了以下欄位:

- Version (V), Padding (P), Length:

  和SR定義相同

- Packet Type (PT): 8 bits

  設定為203

- Source Count (SC): 5 bits

  在這個BYE中有幾個SSRC跟CSRC要被終結

APP

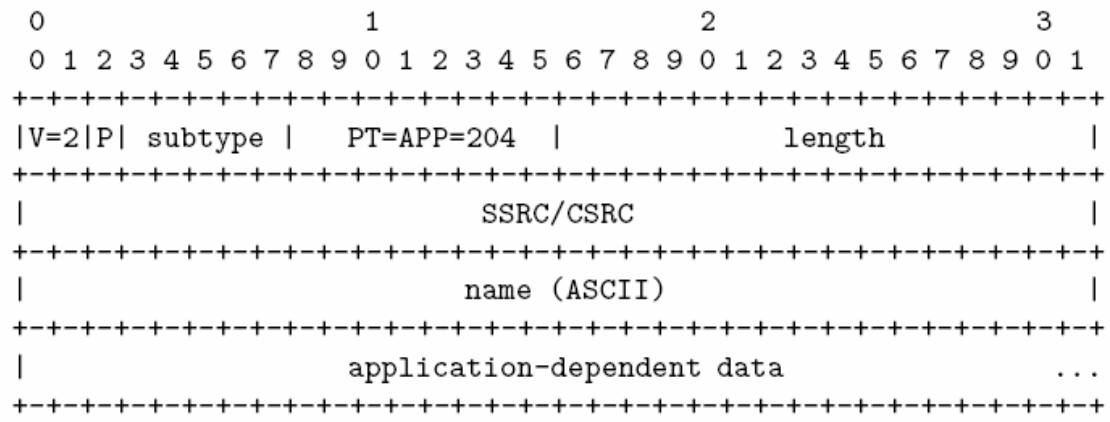


圖 18 APP 封包格式[1]

APP封包是交給應用程式自行定義，格式如圖18:

- Version (V), Padding (P), Length:

  和SR相同

- Subtype: 5 bits

  給應用程式自行定義格式

- Packet Type (PT): 8 bits

  設定為204

- Name: 4bytes

  應用程式的名稱

- Application-dependent Data: 不定長度

  應用程式自行要求的內容，長度應為32位元的倍數。

3. 聲音壓縮

3.1 聲音壓縮的基本

由於聲音儲存需要龐大的空間，而傳輸也需要大量的頻寬，十分的浪費資源，所以就有人想說把聲音壓縮，以減少使用的資源。聲音壓縮主要分為兩大類，非失真壓縮跟失真壓縮。非失真壓縮是純粹以資料的統計特性為基礎，不破壞資料，可以百分之百的還原回原來的聲音，不過壓縮比有限，大概是 1:3 左右。失真壓縮則在壓縮的過程中丟棄了部份資料，雖然無法完全還原回原來的聲音，不過相對的壓縮比就相當的高，可以達到 1:12 以上。要去除哪些信號來讓壓縮比提高，而人耳聽不出來就是各家的技術所在。

3.1.1靜音門檻曲線(absolute threshold)：

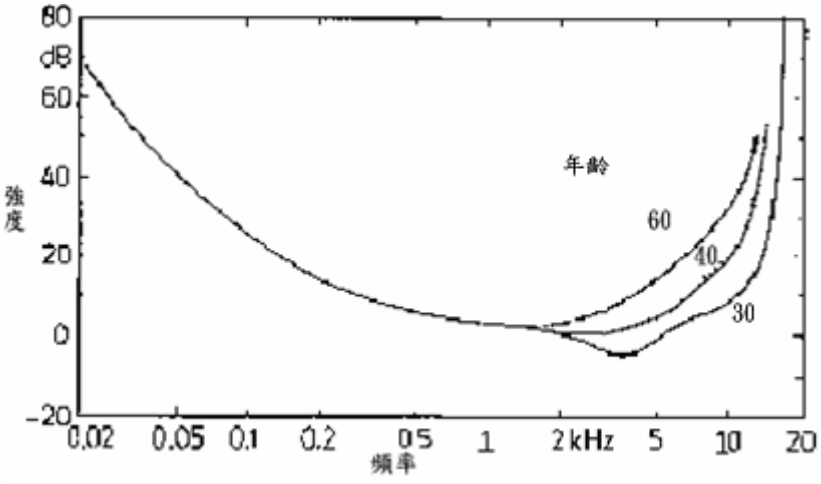


圖 19 靜音門檻曲線

人的聽覺系統對於不同頻率的聲音有著不同的靈敏度，頻率2k~4kHz 為人耳最靈敏的地帶，在此頻率可以輕易的察覺為小的聲音變化，而其它頻率人耳就不那麼靈敏。圖3.1就是人耳的靈敏度的曲線，強度在此曲線下的聲音無法被人耳察覺，所以被稱之為靜音門檻曲線。這條曲線同時也是人耳的敏感度指標，曲線越低的地方，人就能夠察覺越小的聲音變化。在4kHz左右是人耳最靈敏的頻率。

。

3.1.2臨界頻帶(Critical band)：

將所有人耳靈敏度相近的頻率切割後，分出來的各頻率稱為臨界頻帶。這些臨

界頻帶寬度不等，因此在MPEG/Audio 壓縮中將頻率切成32 等分來近似於人耳的臨界頻帶。

3.1.3遮蔽效應(masking effect)：

人類的聽覺效應中，強的聲音會遮蔽弱的聲音，讓若的聲音無法被察覺，例如一段聲音中有一個特別大聲的聲音，同時的其他比較的聲音就可能會聽不到。由頻譜上來看，當某一個頻率的大小特別大時，其鄰近的頻率會被遮蔽，計算遮蔽的臨界值(masking threshold)的模型為精神聽覺模型(後述)

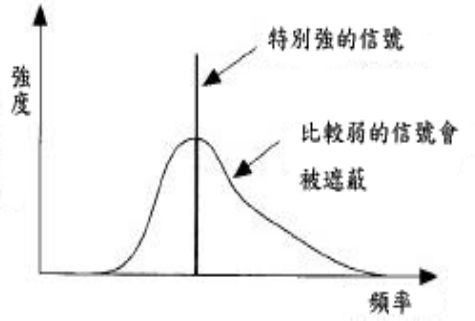


圖 20 遮蔽效應

3.2 MPEG-1 Audio layer3(Mp3)壓縮

由MPEG組織所定義，最普遍的失真壓縮格式之一。在取樣率(Sampling rate)上，可以是32kHz、44.1kHz 或是48kHz，並有單聲道(monophonic)，雙聲道(dual-monophonic)，立體聲(stereo)，聯合立體聲(joint-stereo)等模式可供選擇。Bit rate可以從32kbps到320kbps，有多種可以選擇，也支援變動速率。在我們的系統中傳送的聲音規格如下:

取樣率	聲道	取樣大小	位元率	壓縮比
44100Hz	1	16位元	64kbps	11

表 1 聲音規格

Mp3 壓縮這種失真壓縮的基礎是在於人耳的特性，對哪些頻率的聲音比較敏感，以及遮蔽效應，來決定丟棄哪些資訊，並分配有限的位元在哪些地方。。

Mp3 壓縮經過以下的程序:

- 訊號處理

Polyphase filter bank

MDCT

- 利用聽覺神經模型計算遮蔽曲線

計算靜音門檻曲線

計算遮蔽效應

- 計算 globe gain 跟 scale factor 進行 bit 分配的動作
- 查表填上 bit stream

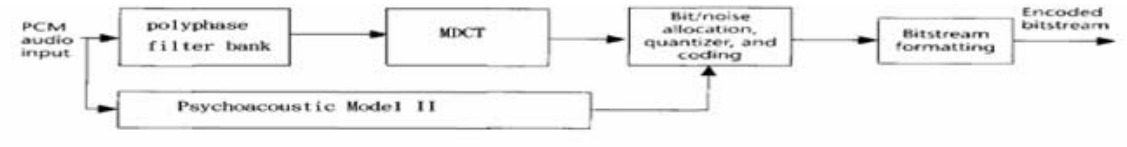


圖 21 mp3 壓縮流程

3.2.1 多相濾波器(Polyphase filter bank)

由於人的耳朵可以分為多個靈敏度相近的頻帶，所以 Polyphase filter bank 的目的就在於將聲音劃分為 32 個頻帶，分別壓縮。將輸入的訊號經過 band pass filter，分成 32 個頻帶，每個頻帶 36 個取樣之後輸出為一個 frame，共 1152 個取樣:

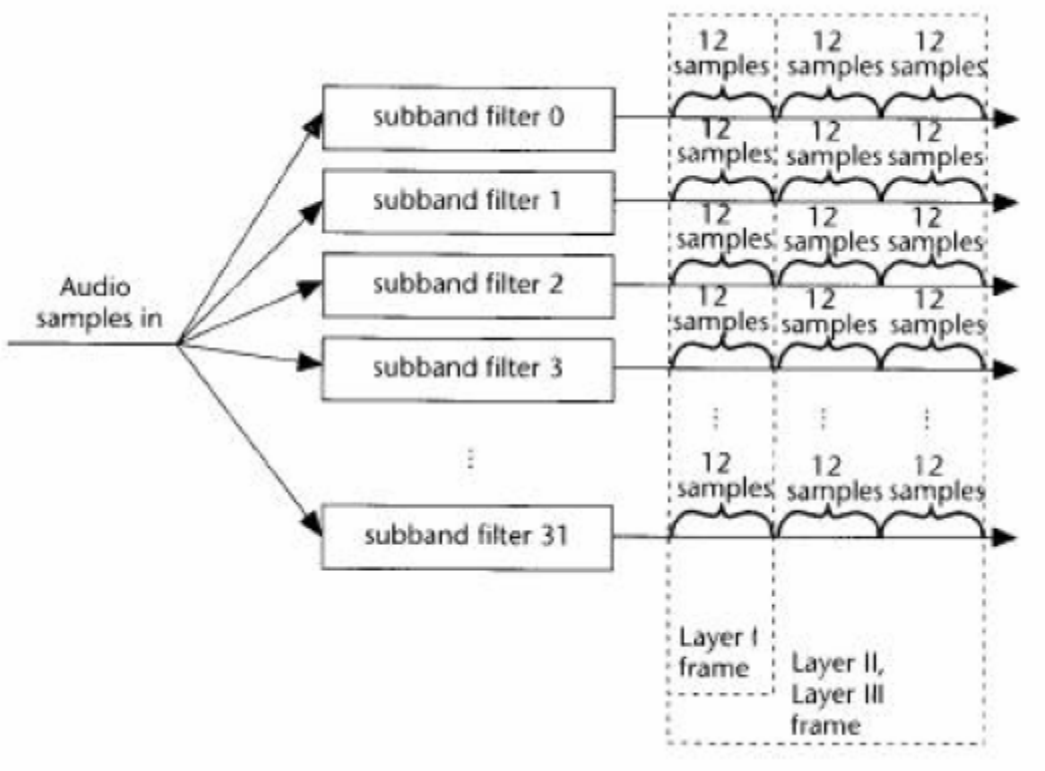


圖 22 多相濾波器[2]

3.2.2 Modified discrete cosine transform(MDCT)

由多相濾波器所得出的結果還要經過MDCT 的處理後才能作量化，主要目的是為了要補償多相濾波器的缺點。MDCT將多相濾波器的輸出(共32 個subbands，每個subband 有36 個取樣)作更細的劃分，因此提供了比原來更好的頻率上的解析度。

為了消除block效應，MDCT會先經過window來減低區塊邊緣的信號大小。MDCT有四種window: long window、short window、由long window 到short window 的轉移window、和由short window 到long window的轉移window。在long window的大小為36個取樣，每次處理時放進18 個新的取樣，和前一次有50%的重疊。short window則是12個取樣，每次處理6個新的取樣，和前一次也是50%的重疊。Long window 提供了較好的頻率解析度，而short window 提供了較好的時間解析度。要用哪種window則由Psychoacoustic Model II的計算結果決定。

MDCT的公式如下：

$$X_k = \sum_{n=0}^{2N-1} x_n \cos[\frac{\pi}{n}(n+\frac{1}{2}+\frac{N}{2})(k+\frac{1}{2})]$$

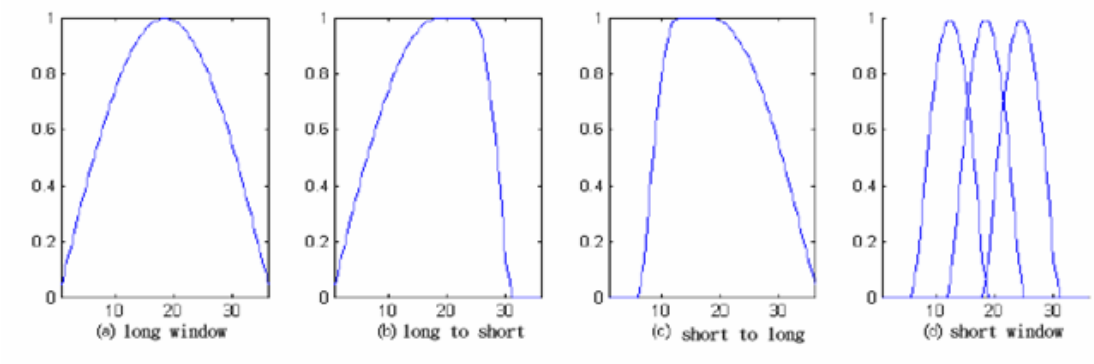


圖 23 MDCT window

在這邊列出實際上跑出來的 MDCT 數據作為範例

short windows 的 MDCT，前 6 個 sample 是舊的資料，後 6 個 sample 是新輸入的資料

1.640647e-010
1.027373e-010
-5.350009e-012
4.873280e-010
-4.335368e-010
1.370018e-010
5.539137e-010
-1.722703e-009
-1.627872e-010
-1.895251e-010
2.202607e-010
-1.094534e-011

輸出 6 個 MDCT 資料

1.670587e-020
-2.611324e-020
7.662660e-022
-1.626481e-022

3.381129e-020
1.332636e-021

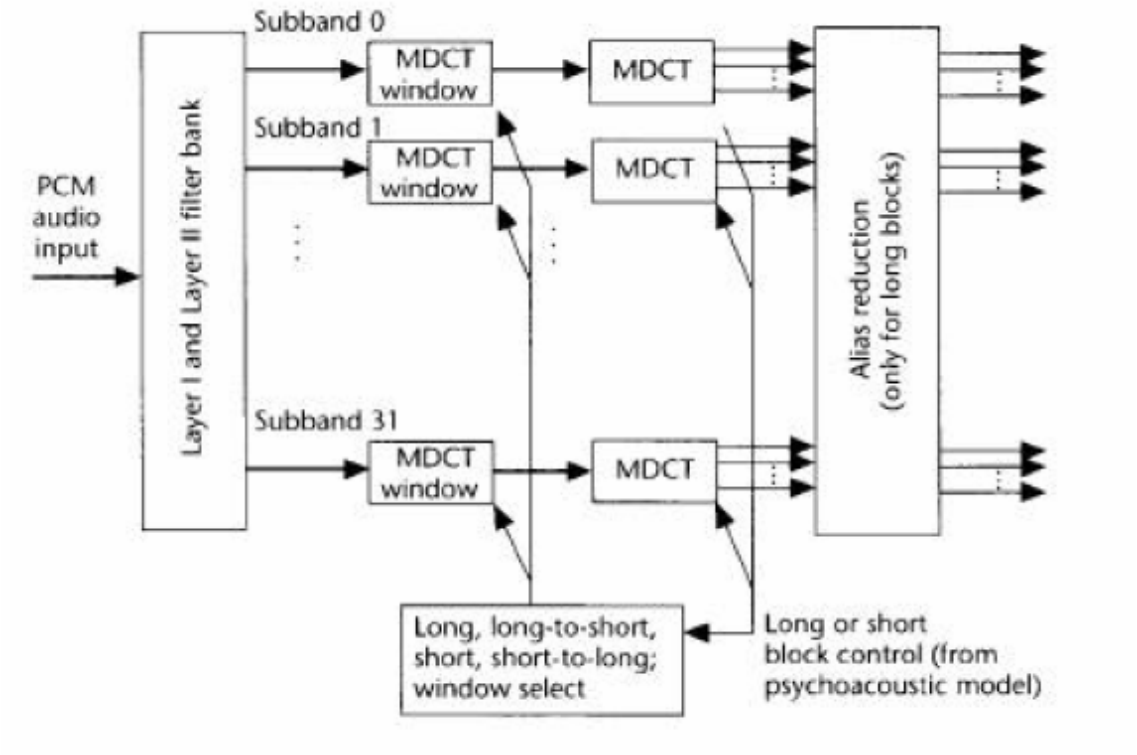


圖 24 MDCT[2]

3.2.3 計算遮蔽曲線

遮蔽曲線是在綜合信號的各個遮蔽效應，與人耳的靜音門檻之後，算出來的一條聽的到的聲音大小的下限的曲線。如果信號在這條曲線之下，就表示人耳無法察覺，所以可以直接丟棄不需要編碼。同時也當作位元分配的依據。

3.2.3.1 靜音門檻曲線的模型

如同前述，這是利用一個人耳對聲音有多少敏感度，計算一個只要不到這個程度，人耳就聽不到的底限。採用的 model 如下:

$$ATH(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4$$

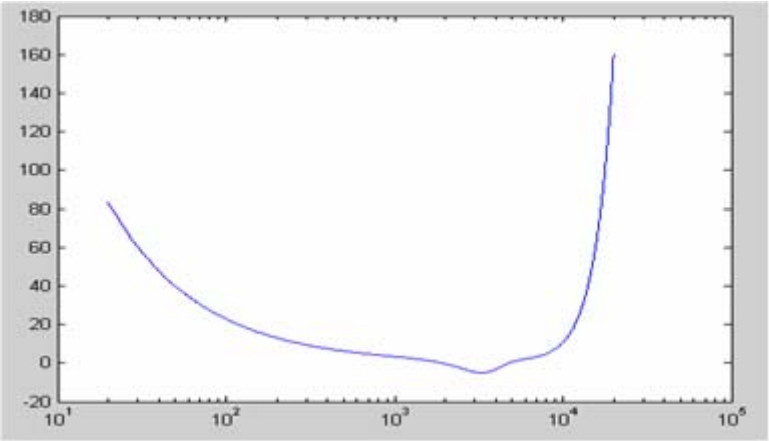


圖 25 ATH

3.2.2.3 精神聽覺模型 II(Psychoacoustic Model II)

這是一個用來模擬人耳敏感度的模型，他計算了在這個 frmae 中有多少特別強的信號，然後計算他會在頻率上遮蔽掉多少信號:

- Psychoacoustic Model II 是用來計算遮蔽曲線的模型，它是選出 frame 中特別強的單音(tone)，然後再加上其他頻帶的 tone 的影響後，算出 tone 在頻帶中的能量比率，稱之為音調度。而非 tone 的就當成背景雜訊處理。
- 然後從音調度算出會遮蓋的信號，當作遮蓋曲線。
- 由於模型中使用的頻率區段的分割跟壓縮使用的不同，所以需要做映射的動作。

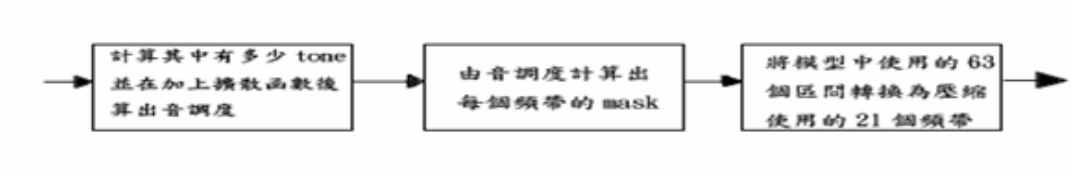


圖 26 聽覺神經模型計算流程

計算音調度

- tone 的識別法是將信號經過 FFT 之後，去看過去到現在的振幅與相角是否連續，如果連續的話，就表示這邊是 tone。

(1) 首先先要將輸入取樣值 si 乘上一個 Hann window 得到 swi

(2) 再將計算所得 swi 的值作 FFT，可以得到大小 rw 和相角 fw

由前二次的大小和相角值計算預測的大小$\hat{r}_w$

和預測的相角$\hat{f}_w$

$$\hat{r}_w = 2r_w(t-1) - r_w(t-2)$$

$$\hat{f}_w = 2f_w(t-1) - f_w(t-2)$$

- 計算不可預測的量度(unpredictability measure) cw，這表示了這個頻帶中的 tone 的能量的比率

$$c_w = \frac{\sqrt{(r_w \cos f_w - \hat{r}_w \cos \hat{f}_w)^2 + (r_w \sin f_w - \hat{r}_w \sin \hat{f}_w)^2}}{r_w + |\hat{r}_w|}$$

計算能量 eb 和不可預測性 cb

$$e_b = \sum_{wlow_b}^{whigh_b} r_w^2$$

$$c_b = \sum_{wlow_b}^{whigh_b} r_w^2 \cdot c_w$$

- 接下來是要將計算出這個 window 裡 tone 總共佔的比率，稱之為音調度（tonality）。
- Spreading function

Spreading function 是用來計算其他頻帶的 tone 會影響到其他頻帶的模型:

- i 為要 spread 的信號的 bark，j 為被 spread 進的頻帶的 bard

$$tempx = 1.5(j - i)$$

$$x = 8\min\{(tempx - 0.5)^2 - 2(tempx - 0.5), 0\}$$

$$tempy = 15.811389 + 7.5(tempx + 0.474) - 17.5(1.0 + (tempx + 0.474)^2)^{0.5}$$

$$sprdngf(i, j) = \begin{cases} 0 \text{ if}(\text{tempy} < 100) \\ 10 - \dfrac{x + tempy}{10} \text{ else} \end{cases}$$

算出音調度

- 計算 partitioned energy 與 unpredictability with spreading function

$$ecb(b) = \sum_{bb=1}^{\text{b max}} e_{bb} * sprdngf(bval_{bb}, bval_b)$$

$$ct_b = \sum_{bb=1}^{\text{b max}} c_{bb} * sprdngf(bval_{bb}, bval_b)$$

計算每一個 partition 的音調度(tonality) tbb

$$tb_b = -0.299 - 0.43\log_e \frac{ct_b}{ecb_b}$$

- 由音調度進行簡單的運算，就可以得到 masking。
- Tone 比 noise 強一定程度後會遮蔽掉 noise，稱之為 tone masking noise
- 而 noise 比 tone 強一定程度後會遮蔽掉 tone，稱之為 noise masking tone

- 計算每一個 partition 的 Signal-to-Masking ratio (SMR)

$$SMR_b = TMN_b \cdot tb_b + NMT_b(1 - tb_b)$$

TMN_b is tone - masking - noise, 標準中指定爲18dB

NMT_b is noise - masking - tone, 標準中指定爲6dB

- 映射

在計算出每一個 partition 的 masking 後，要將之換算到實際上使用的各個 band 的 masking，所以要經過正規化的動作

計算正規化能量(normalized energy) enb

$$en_b = \frac{ecb_b}{\sum_{bb=0}^{b\max} sprdngf(bval_{bb}, bval_b)}$$

計算功率比(power ratio) bcb

$$bc_b = 10^{\frac{-SNR}{10}}$$

- 計算能量臨界(energy threshold) nbb
- nbb=enb・bcb
- 計算每條頻譜的能量臨界 nbw

$$nb_w = \frac{nb_b}{whigh_b - wlow_b + 1}$$

- 計算絕對臨界(absolute threshold) thrw

- thrw=max(nbw ,abstrw)
- 其中 absthrw 是在標準中指定

計算各 subband 之能量 epartn

$$epart_n = \sum_{w=wlow_n}^{whigh_v} r_w^2$$

- 根據標準中指定的各個 subband 之 widthn 值來計算雜訊階(noise level)npartn

$$npart_n = n \sum_{w=wlow_n}^{whigh} thr_w$$

- 最後可以計算得到各個 subband 之 SMR

$$SMR = 10\log_{10}\frac{epart_n}{npart_n}$$

從圖 3.8 中，可以簡單的看出 SNR、SMR、NMR 的關係

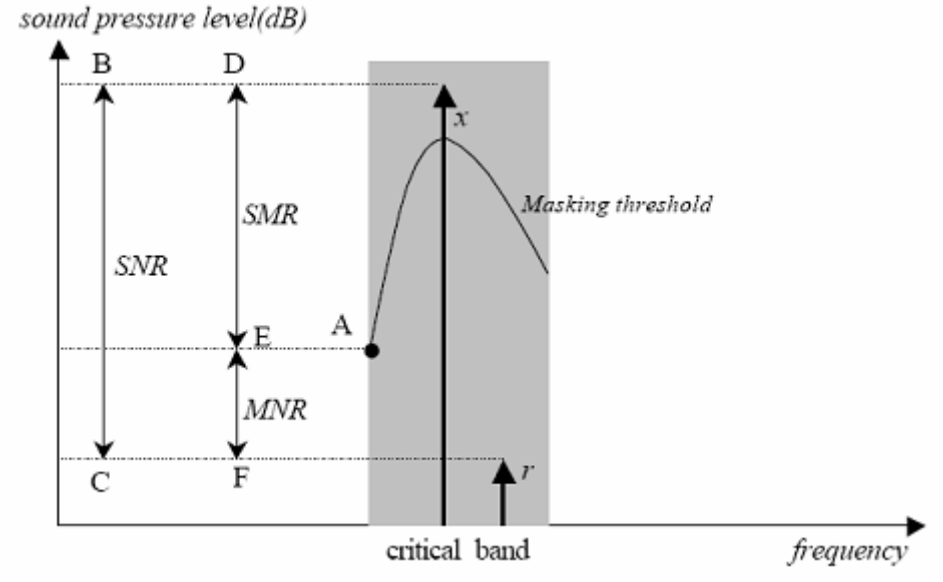


圖 27 SNR SMR MNR

如此一來，就可以得到遮蔽曲線。

3.3 位元分配(Bit allocation)

- 位元分配的時候，跟遮蔽曲線息息相關，遮蔽曲線不單是人耳聽覺的極限，也是敏感度。所以可以在人耳聽覺敏感的地方增加位元，以確保壓縮品質。
- 位元分配的主要動作就是量化，將信號除以一個 step 以讓數字變小，每個頻帶都有自己的 step。Step 越大，所佔的位元就越少。

3.3.1 量化

一個信號 X 量化的公式如下

$$S_i = \mathrm{int}\left(\frac{X_i^{\frac{3}{4}}}{\Delta_q}\right)$$

$$\Delta_q = 2^{\frac{1}{4}(g-210)-(scale\_fac\_multiplier \times c_q)}$$

Xi 為輸入信號。受底下三個參數而成：globe gain (g), scale factor (cq)，還有單位 cq 所代表的 gain scale_fac_multiplier 。g 為所有頻帶共用，越大整體所需位元就越少。Cq 為每個頻帶專有，越大分配在這個頻帶的位元就越多。下面就介紹其中一個分配法。

3.3.2 計算適當的 g 與 cq

a. 一開始在 cq 皆為 0 的情形下，用二分逼近法計算出合於 bit rate 的 g 並計算此時的 noise

b. 進行放大 cq 的工作

放大 cq 時看使用者要求來決定放大哪些頻段:

- 放大所有超過 masking 的 band
- 放大失真大於 50%最大失真的 band
- 只放大最大失真的頻段

失真為 noise 跟該頻段信號平均大小的比值

c. 計算傳輸 cq 所需的 bit，如果過大，就放大 cq 的 scale(增加 scale_fac_multiplier)，然後減少 cq 的數值

- 計算在此 cq 下合於 bit rate 的 g
- 根據 block type，做以下的比較:

超過 masking 的頻帶數目

總共的 noise
noise 的 variance
最大的 noise

- 來決定是否要留下新的 g 跟 cq
-

d. 終止條件:

- 沒有 band 超過 masking 且不需要再調整 noise 分布
- 全部的 band 的 cq 同時被放大
- g 大於 255
- 調整失敗超過一定次數

就物理意義來看，這個方法是將 bit 分配在 noise 跟信號相比比較強的地方，而如果 bit 分配太多的時候，就增加整體的 step size 來減少 bit。

4. 聲音修補

跟先將整個檔下載到硬碟的情形不同，串流是即時撥出剛剛接收到的東西，所以一但遇到封包延遲，或是掉落，沒有時間可以等待重新傳送這個封包，所以就會出現問題。遇到遺失封包時，由於播放的時候會出現沒有東西可以播放的情形，所以我們聽到的聲音就會斷斷續續，對人耳來說，這樣斷斷續續的聲音極度明顯，對於辨識內容也造成了極大的困擾，讓人非常難以忍受。這極度的影響傳輸的聲音品質

4.1 直接重複上一個片段

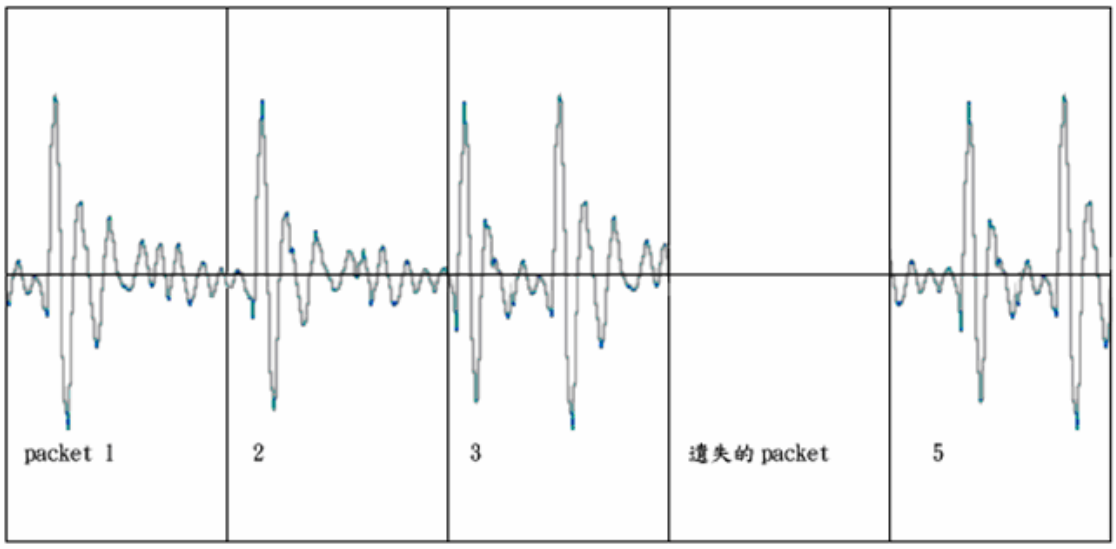
在我們的系統中，每秒鐘共有 43 個封包傳輸，每一個封包包含了 1024 個取樣，相當於 23ms，這樣長的時間無法用簡單的內插功能來修補。

由於聲音在短時間內的波形是極度類似的，所以傳統上可以直接重複上一個封包的片段，這樣子就不會出現斷斷續續的情形，不過同時也會因為單純重複上一個片段，而被人耳察覺到。

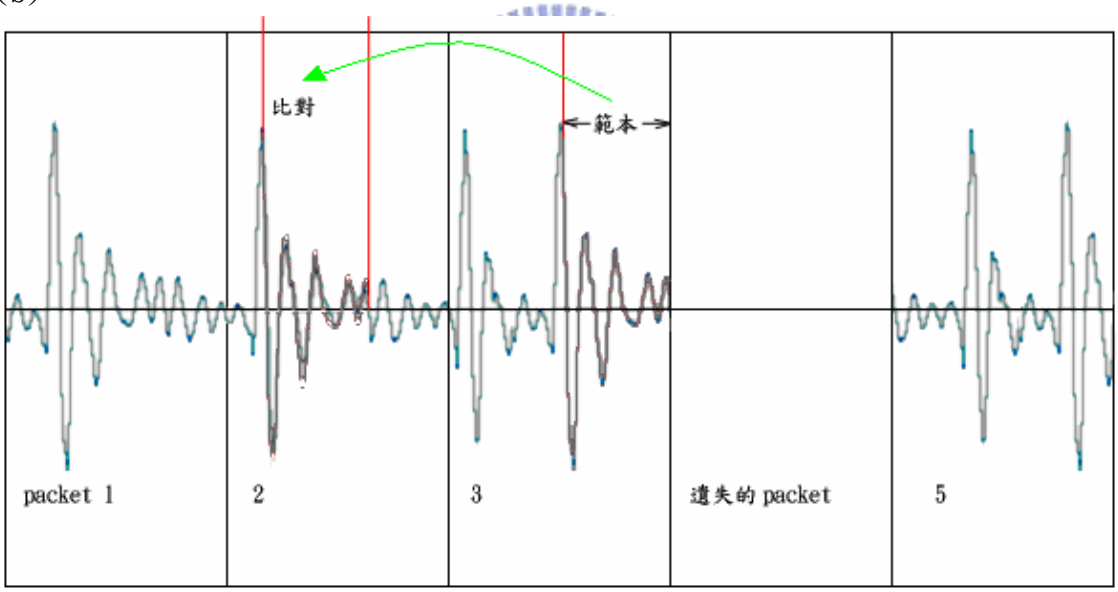
4.2 樣本比對(pattern matching)

樣本比對就是遇到掉的封包的時候，修補的技巧之一，他同樣是利用聲音在短時間內波形類似的特性，不過改善了單純重複上一個片段容易被察覺的缺點。它的精神就是將遺失的片段前後的取樣作為樣本，到其他的地方去尋找跟這個樣本接近的部份，然後將相鄰的片段作為修補之用，複製到空白處。

(a)



(b)



(c)

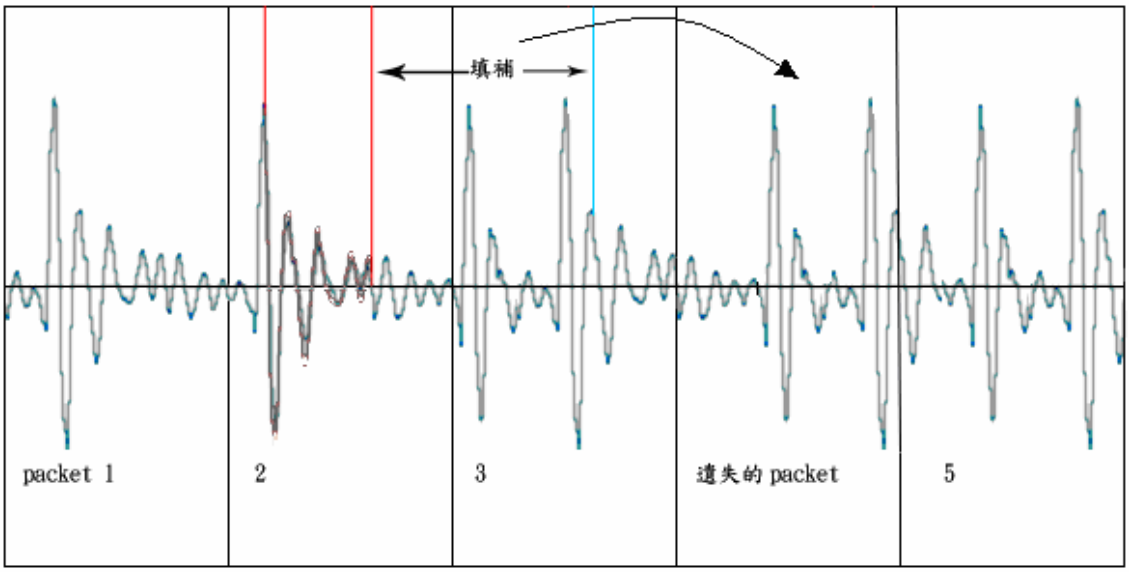


圖 28 樣本比對

4.3 多頻帶的樣本比對

樣本比對可以在原來的時域上面進行，不同的聲音通常在某些的頻段會比較明顯，而在其他的頻段不顯著，所以可以將樣本比對施行在這些特別明顯的頻段上。

剛好失真壓縮都是以頻段壓縮為主，正好可以拿來利用在樣本比對之上。所以可以把解碼的中間信號儲存下來作比對之用。頻段比對可以讓比對選擇在雜訊比較弱的頻帶，來對抗雜訊的干擾。

5. 語音傳輸系統

在這章我會解釋我們整個系統關於語音傳輸的部份
從系統語音部份的架構，語音封包的形式， 以及封包掉落時的處理

5.1 語音傳輸系統的架構

5.1.1 伺服器部份

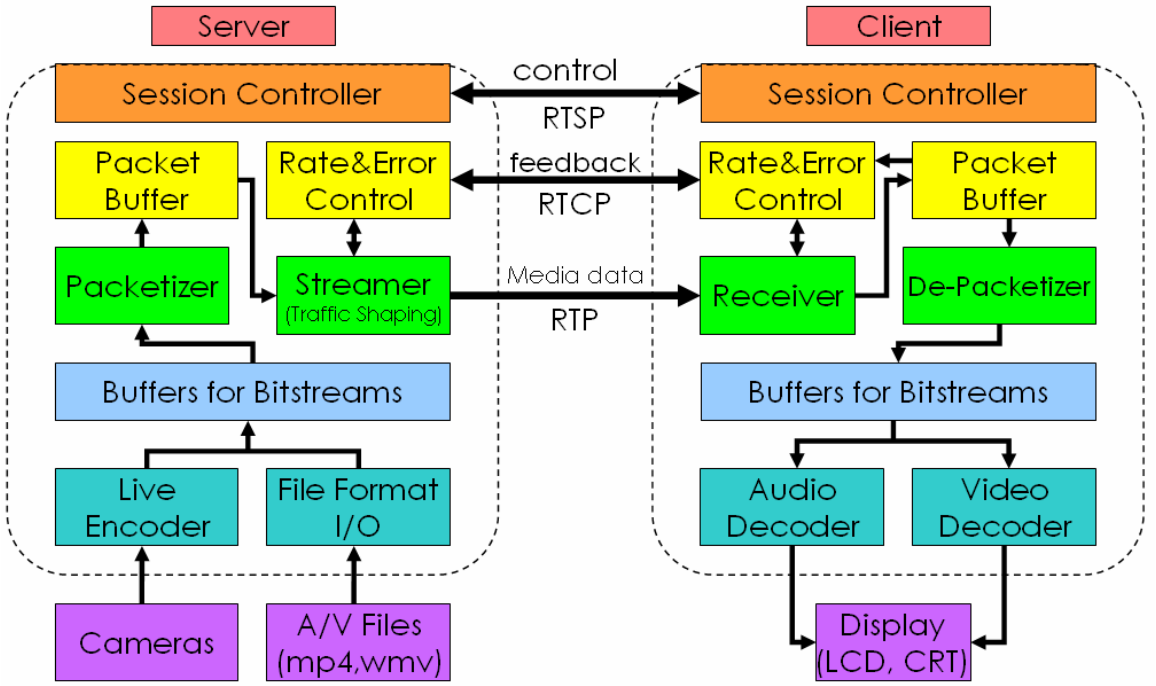


圖 29 典型 RTP RTSP 系統架構

5.2.1 伺服端

伺服端含有數個模組

1. 3GP Parser模組

   - 負責讀取並解析MPEG-4檔案

2. 3GP Creator模組

   - 負責製作MPEG-4(.mp4)檔案跟即時串流時給客戶端使用的虛擬檔案(.3gp)

3. mpeg4影像壓縮模組

   - 負責將影像壓縮成mpeg4格式

4. AAC聲音壓縮模組

   - 負責將聲音壓縮成AAC格式

5. Sub Server 模組

   - 負責伺服器端的串流傳輸，包括連線的初始、連接、建立，RTP、RTSP

的封封處理等等。

6. Server Dialog模組

- 使用者界面

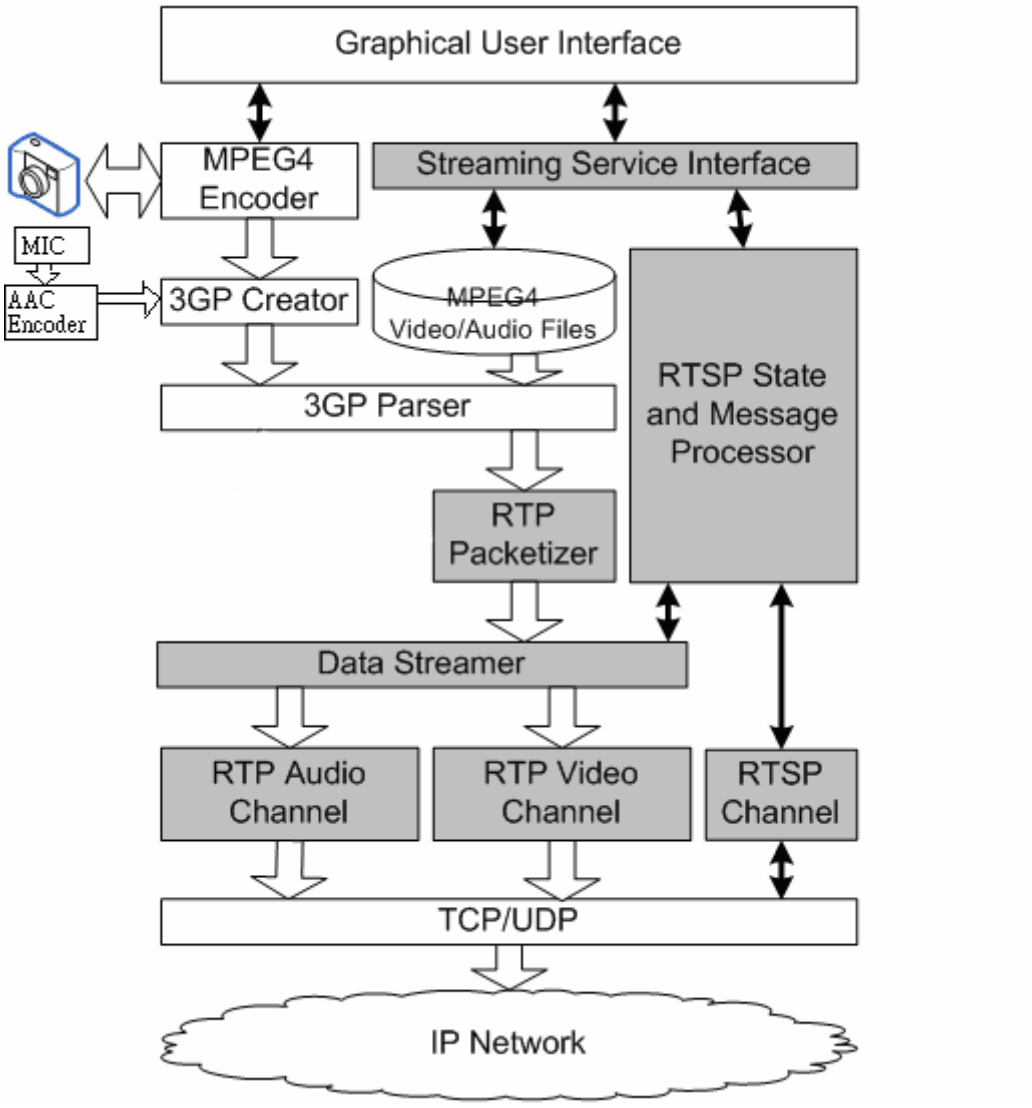


圖 30 伺服端架構

5.1.2 客戶端部份

1. Sub Client 模組

- 負責處理所有網路傳輸的部份，包括連線初始、連接、建立，RTP與RTSP封包的處理等等。

2. Client Dialog 模組

   - 負責使用者界面

3. MPEG-4解壓縮模組

   - 負責解壓縮MPEG-4影像

4. AAC解壓縮模組

   - 負責解壓縮AAC聲音

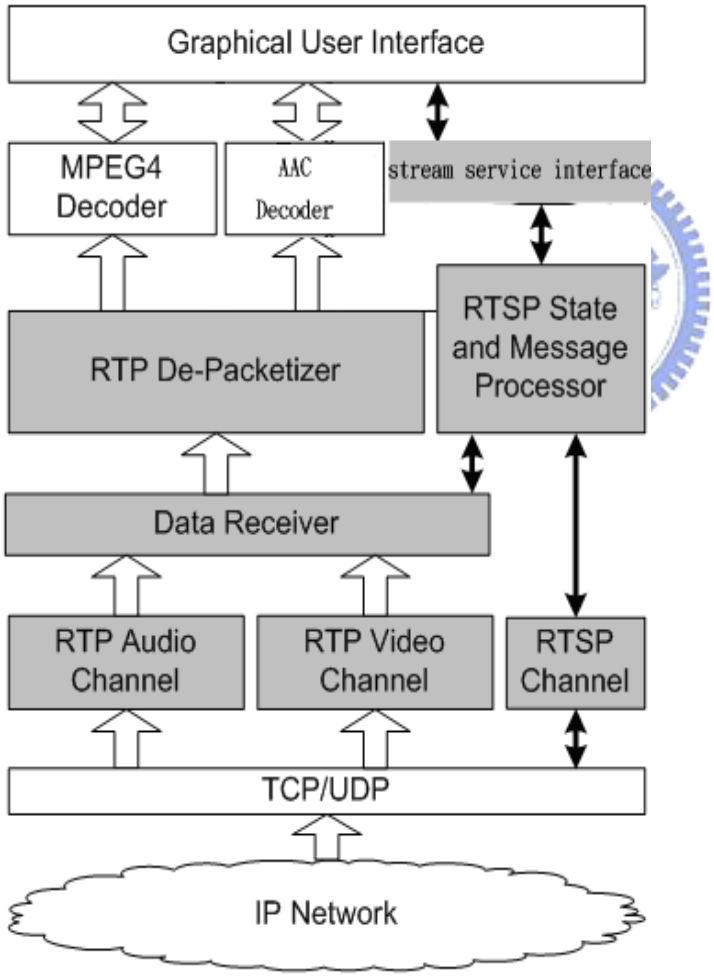


圖 31 客戶端架構

5.2 聲音的抓取與播放

聲音的抓取與播放是採用 VC6 內建的函式控制音效卡，將聲音抓取進來處理，或

是將解壓縮出來的聲音播放出去

5.2.1 聲音抓取用的函式

- UINT waveInGetNumDevs(VOID);

回報有多少個聲音輸入裝置

- MMRESULT waveInGetDevCaps(UINT uDeviceID, LPWAVEINCAPS pwic, UINT cbwic);

將第 uDeviceID 個聲音輸入裝置照著 pwic 設定,pwic 是 WAVEINCAPS 格式,記載著裝置要設定成什麼格式， cbwic 是 pwic 的大小

- MMRESULT waveInOpen(LPHWAVEIN phwi, UINT uDeviceID, LPWAVEFORMATEX pwfx, DWORD dwCallback, DWORD dwCallbackInstance, DWORD fdwOpen);

開啟聲音輸入。將第 uDeviceID 個聲音輸入裝置連到 phwi 這個 handle 上，pwfx 為所要求的格式(後敘)， dwCallback 設定 callback 使用的函式，可以在這邊進行抓取進來的資料的處理；在我們的系統中，這邊負責將抓進來的訊號喂給 AAC Encoder 壓縮成 AAC 格式。 fdwOpen 設定 callback 是 function 還是 thread

- MMRESULT waveInReset(HWAVEIN hwi);

停止裝置 hwi 的聲音輸入，並將 buffer 清空

- MMRESULT waveInAddBuffer(HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh);

從裝置 hwi 抓取一筆資料存到 pwh 中所指向的 buffer，cbwh 為 pwh 指向的 buffer 的長度。在抓取資料後，會自動執行 callback 函式一次。所以只要 callback 函式裡自行呼叫此函式，就會自動的連續不斷抓取聲音。

- MMRESULT waveInStart(HWAVEIN hwi);

啟動裝置 hwi 的聲音輸入

- MMRESULT waveInClose(HWAVEIN hwi);

停止聲音輸入裝置 hwi

5.2.2 聲音輸出用的函式

- UINT waveOutGetNumDevs(VOID);

回報有多少個聲音輸出裝置

- MMRESULT waveOutGetDevCaps(UINT uDeviceID, LPWAVEOUTCAPS pwoc, UINT cbwoc);

將第 uDeviceID 個聲音輸出裝置照著 pwoc 設定，pwoc 是 WAVEINCAPS 格式，記載著裝置要設定成什麼格式， cbwoc 是 pwoc 的大小

- MMRESULT waveOutOpen(LPHWAVEOUT phwo, UINT uDeviceID,

LPWAVEFORMATEX pwfx, DWORD dwCallback, DWORD dwCallbackInstance, DWORD fdwOpen);

開啟聲音輸出。將第 uDeviceID 個聲音輸出裝置連到 phwo 這個 handle 上，pwfx 為所要求的格式， dwCallback 設定 callback 使用的函式， fdwOpen 設定 callback 是 function 還是 thread

- MMRESULT waveOutWrite(HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh);

將 pwh 所指向的長度為 cbwh 的 data block 送到輸出裝置 hwo 上播放。

- MMRESULT waveOutReset(HWAVEOUT hwo);

停止聲音輸出裝置 hwo 的輸出，並且將 buffer 清空

- MMRESULT waveOutClose(HWAVEOUT hwo);

關閉聲音輸出裝置 hwo

5.2.3 WAVEFORMATEX 格式

```
typedef struct {
WORD wFormatTag;
WORD nChannels;
DWORD nSamplesPerSec;
DWORD nAvgBytesPerSec;
WORD nBlockAlign;
WORD wBitsPerSample;
WORD cbSize; }
WAVEFORMATEX;
```

- wFormatTag 為聲音的格式，1 為 WAVE_FORMAT_PCM
- nChannels 是聲道數，1 為單聲道 2 為雙聲道
- nSamplesPerSec 表示取樣頻率
- nAvgBytesPerSec 表示一秒有多少個 bytes
- nBlockAlign 表示最小的 block 的長度，為 channel 數乘以每個 sample 的 byte 數
- wBitsPerSample 表示一個 sample 用幾個 bits 表示
- cbSize 為 user define 標頭的長度

5.3 聲音的傳輸

在聲音的傳輸上，伺服端將壓好的 AAC frame 包裝成 RTP 封包，然後傳送出去。客戶端則在接收之後，將 RTP 封包拆開成為 AAC frame，然後交給 AAC decoder 解壓縮成為 pcm 之後播放。

聲音傳輸有一些特性，它只要一延遲就會造成聲音播放的不順，然後就會被人耳所察覺到，更不用說有封包遺失了。因此在聲音的傳輸上，要特別注意播放的連續性，避免播放的時候出現斷斷續續的問題。接收端則使用 buffer，控制播放的速度一致。

5.3.1 伺服端

在AAC frame壓縮出來之後,程式會將它加進mp4檔案結構裡,然後交給上層 server 傳輸。

Server 負責傳輸的 thread 如下:

StreamerletIs:負責將資料打包成 RTP 封包

PacketScheduler:負責將封包送出去給網路卡傳輸。由於 UDP 沒有控制網路流量的功能，所以這個 thread 會負責傳輸速率的控制，避免速率過高客戶端無法消化造成 buffer 被填滿，或是速率過低，客戶端客戶端沒有足夠的東西可以播放而造成中斷。

5.3.2 客戶端

客戶端會從網路上抓取 RTP 封包，將其解開成為 AAC frame，然後解壓縮成為聲音播放。相關的 thread 如下

- RecvThreadIsK()
  - 負責接收 RTP 封包
- DePktThreadIsK()
  - 負責將接收的 RTP 封包解開成為 AAC frame
- AudioStatusTimeFunc()
  - 負責解壓縮 AAC frame 成為聲音，並且播放。如果要對聲音信號做什麼處理，例如說上面提到的修補機制，也是放在這邊。

5.4 語音封包的格式

在我們的系統中，語音封包是將 MPEG-4 AAC 的的 frame 加上 RTP 標頭封裝而成，header 部份則是使用了 mp4 檔案格式中關於聲音的檔頭 elementary stream

descriptors (ESDs)，格式如下:

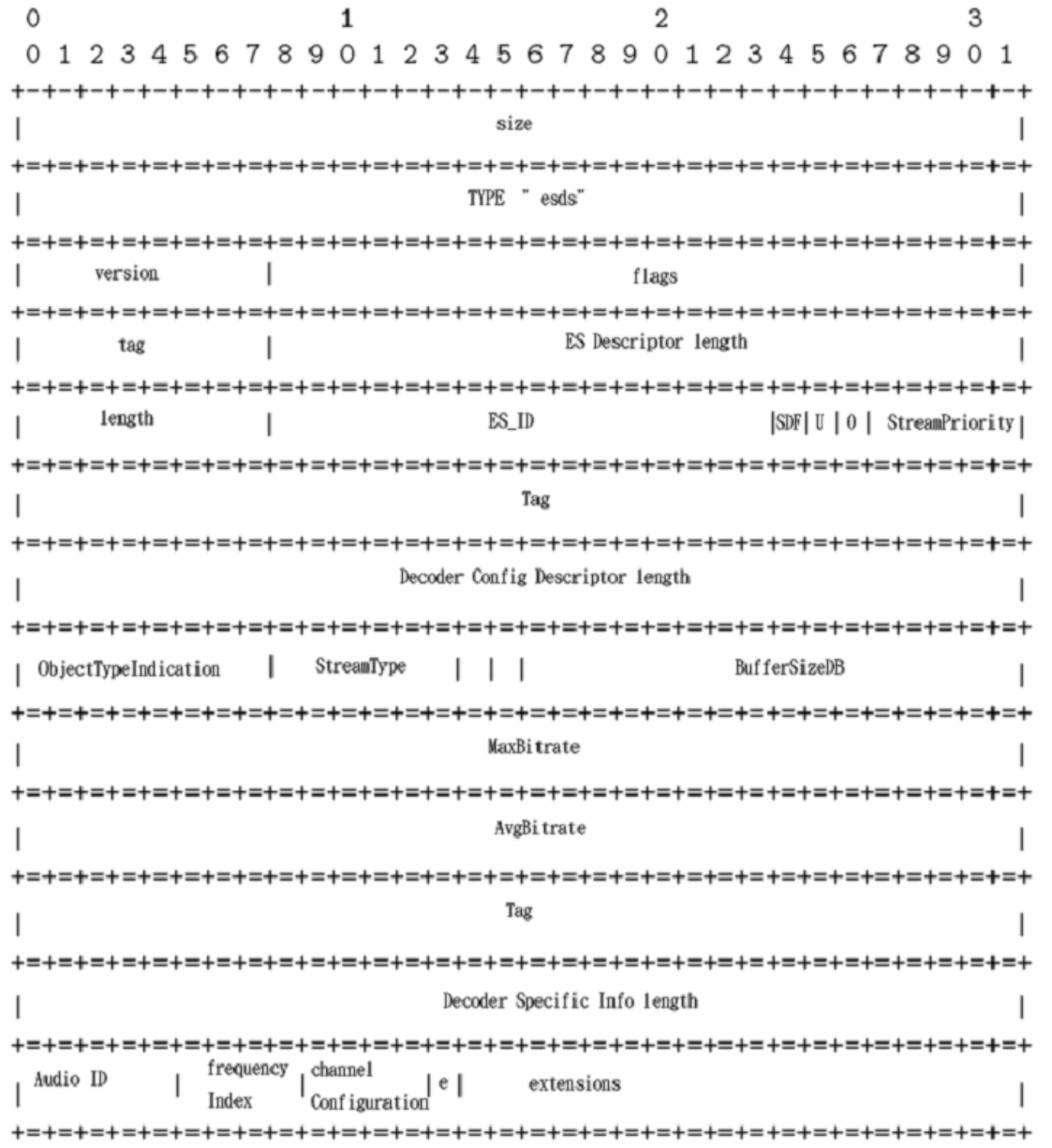


圖 32 ESDS 封包標頭[3]

跟系統相關的有

object type:5 位元，表示物件內容的形式

有如下的選擇

0: NULL

1: AAC Main

2: AAC Low complexity

3: AAC SSR

4: AAC Long term prediction
5: AAC High efficiency
6: Scalable
7: TwinVQ
8: CELP
9: HVXC
10: Reserved
11: Reserved
12: TTSI
13: Main synthetic
14: Wavetable synthesis
15: General MIDI
16: Algorithmic Synthesis and Audio FX
17: AAC Low complexity with error recovery
18: Reserved
19: AAC Long term prediction with error recovery
20: AAC scalable with error recovery
21: TwinVQ with error recovery
22: BSAC with error recovery
23: AAC LD with error recovery
24: CELP with error recovery
25: HXVC with error recovery
26: HILN with error recovery
27: Parametric with error recovery
28: Reserved
29: Reserved
30: Reserved
31: Reserved

MaxBitrate:32 位元，表示最高的位元率
AvgBitrate:32 位元，表示平均的位元率
headerlength:8 位元，表示標頭的總共長度
AudioID:5 位元，用來作為識別
frequencyIndex:4 位元，用來表示聲音的取樣頻率
0: 96000 Hz
1: 88200 Hz

2: 64000 Hz
3: 48000 Hz
4: 44100 Hz
5: 32000 Hz
6: 24000 Hz
7: 22050 Hz
8: 16000 Hz
9: 12000 Hz
10: 11025 Hz
11: 8000 Hz
12: 7350 Hz
15: 使用者自行定義，接下來 24 位元表示真正的取樣頻率

Channel configuration:4 位元，用來識別有多少個聲道
0: 使用者自行定義
1: 一聲道:前聲道
2: 二聲道:左前，右前
3: 三聲道:左前，右前，前方中置
4: 四聲道:左前，右前，前方中置，後方中置
5: 五聲道:左前，右前，前方中置，左後，右後
6: 六聲道:左前，右前，前方中置，左後，右後，重低音
7: 八聲道:左前，右前，前方中置，左後，右後，重低音，左側，右側

frame length flag:1 位元，設成 0 表示一個 frame 有 1024 個取樣，1 表示有 960 個
dependsOnCoreCoder:1 位元，由使用者自行定義
extensionFlag:1 位元，有沒有延伸的檔頭

5.5 封包遺失的處理

在遺失封包的處理上面，我們分別做了完整的聲音片段的樣本比對，以及分成 32 個頻帶的樣本比對。分成 32 個頻帶的處理又分為一階段與兩階段兩種。

5.5.1 完整的聲音片段樣本比對

為了準備修補遺失的聲音片段，所以我們保留了之前收到的五個聲音片段作為樣本比對之用。在發現有遺失的情形的時候，就會利用之前收到最後一個片段的後 512 點作為樣本，在更之前的四個片段中，參考公式 5.1 及圖 33，以 MSE 方式尋找最接近的波形，然後將後面 1024 點複製到遺失的地方播放。找到的片段在

前面四個片段中的位置，這邊稱之為回朔取樣數。如果找到的片段正好是最後一個封包時，回朔取樣數為 0。

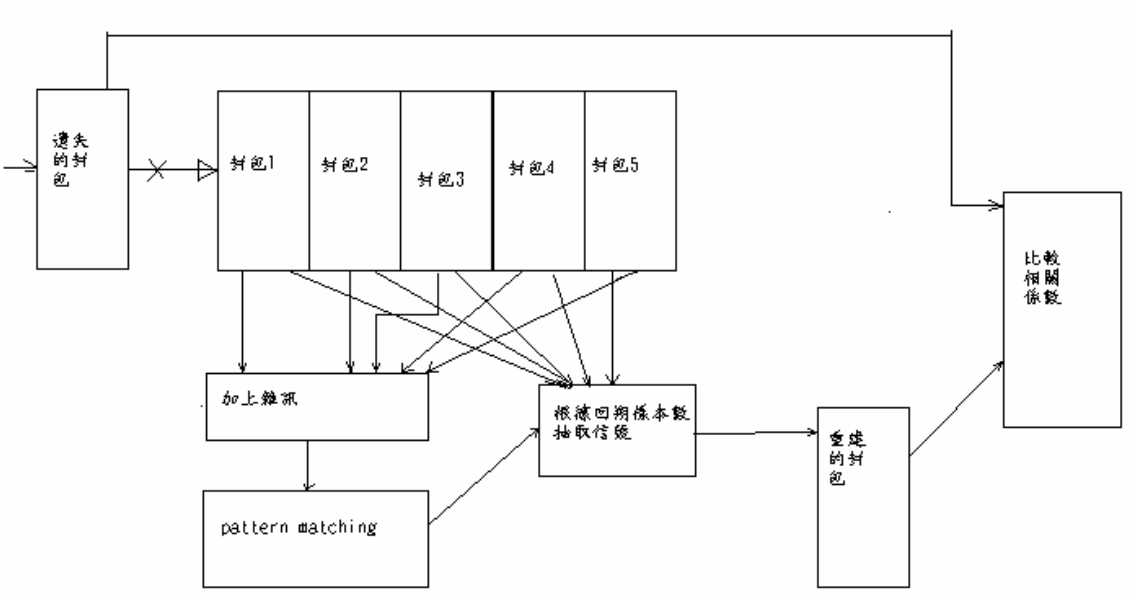


圖 33 全頻帶 pattern matching

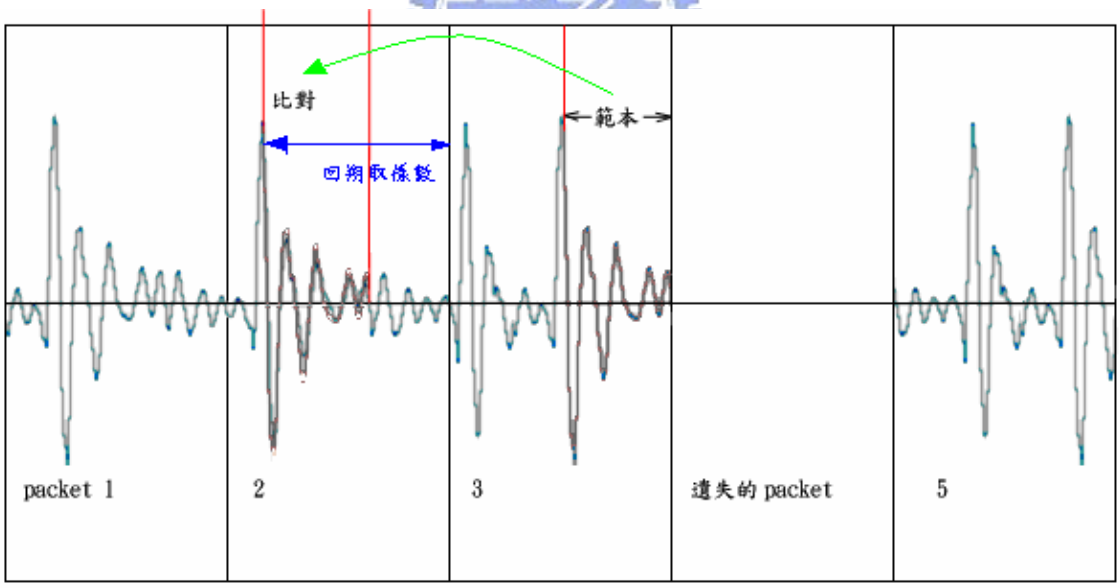


圖 34 回朔取樣數

$\min_{k=0\sim3784}\sum_{i=0}^{511}(X_k[i]-P[i])^2$ ，k 為回朔取樣數，Xk[i]為對應位置的取樣，P[i]為樣本 (5-1)

5.5.2 32 個頻帶的樣本比對

在 32 個頻帶的樣本比對中，同樣是記住前五個聲音片段的資料，再加上已經分成 32 個頻帶的資料，每個片段中每個頻帶有 32 個取樣。在比對的時候，同樣是採取最後收到的片段作為樣本，來跟之前做 MSE 計算找出回朔取樣數。由於聲音已經降頻 32 被，所以回朔取樣數在適用到波形複製的時候，要乘以 32 才是真正的回朔取樣數。這樣會在每個頻帶各找出一個回朔取樣數，但是最後能各使用的只有一個，所以還要經過比較：在這邊我們是採取將每個頻帶 MSE 的錯誤跟該頻帶樣本的能量做比值，取具有最小的比值的頻帶的回朔取樣數來使用。

$$\min_{k=0\sim31}\frac{MSE_k}{\sum_{i=0}^{31}P_k^2[i]}$$ ，MSE_k 為第 k 個頻帶的 MSE 錯誤，$P_k[i]$ 為第 k 個頻帶的樣本值 （5-2）

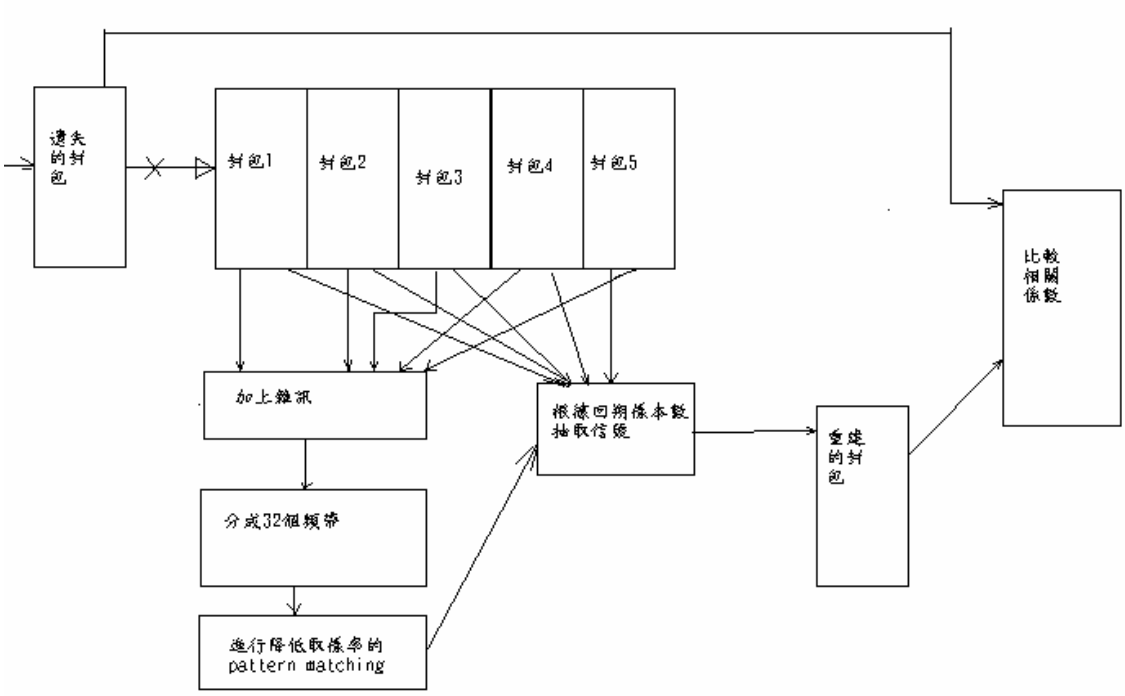


圖 35 多頻帶樣本比對

5.5.3 兩階段的多頻帶比對

由於上面的做法會讓回朔取樣數固定為 32 的倍數，在比對上相當的粗糙，所以我們另外又做了兩階段(2pass)的樣本比對，而上面的做法就稱之為一階段(1pass)的樣本比對。兩階段的樣本比對的做法是經由一階段的比對之後，在一階段的比對找出的最佳頻帶做重新檢視。將過去收到的信號特別抽出這個頻帶，在不降低取樣頻率的情形下進行 1024 點的樣本比對，這樣就不會有一階段固定是 32 的倍數的問題。

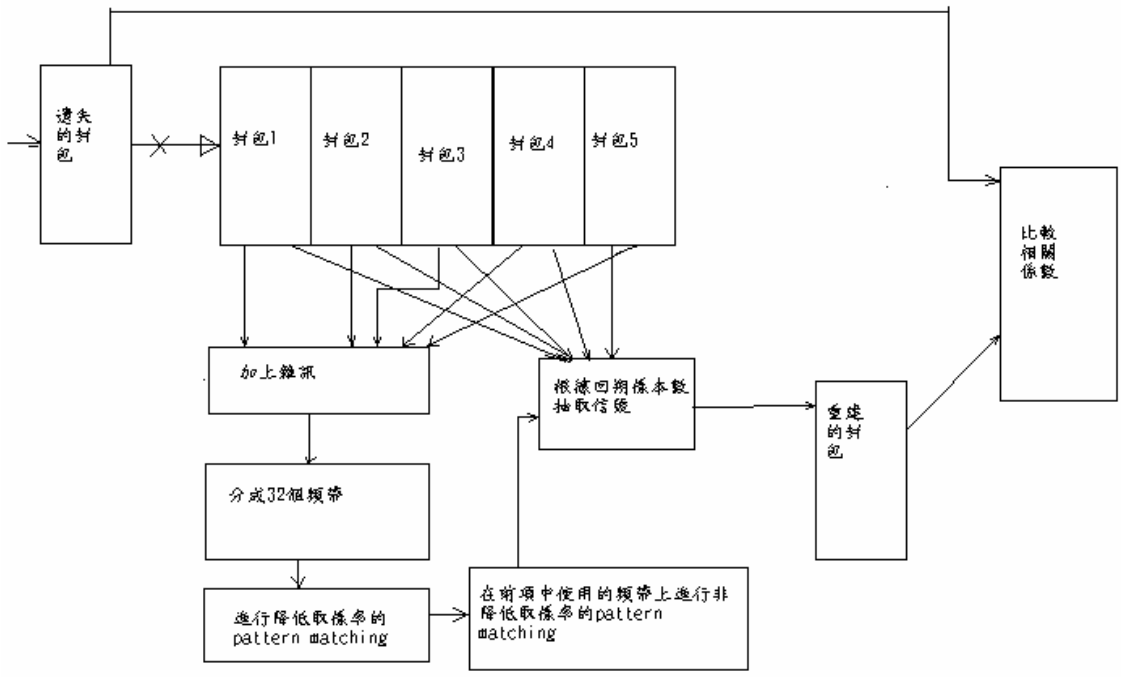


圖 36 兩階段的多頻帶樣本比對

5.5.4 效能比較基準

在比較處修補的性能部份，是採用相關係數的算法，將比對產生的修補片段，跟原來的遺失片段做相關係數的計算，作為修補的性能依據：

$c = \frac{\sum_{m=0}^{M} x[m]y[m]}{y^2[m]}$，x 為比對產生的修補片段，y 為原來的片段，M 為總長度。

6. 系統執行結果

在這一章中，會寫出系統實際執行的結果。

6.1 伺服端介面

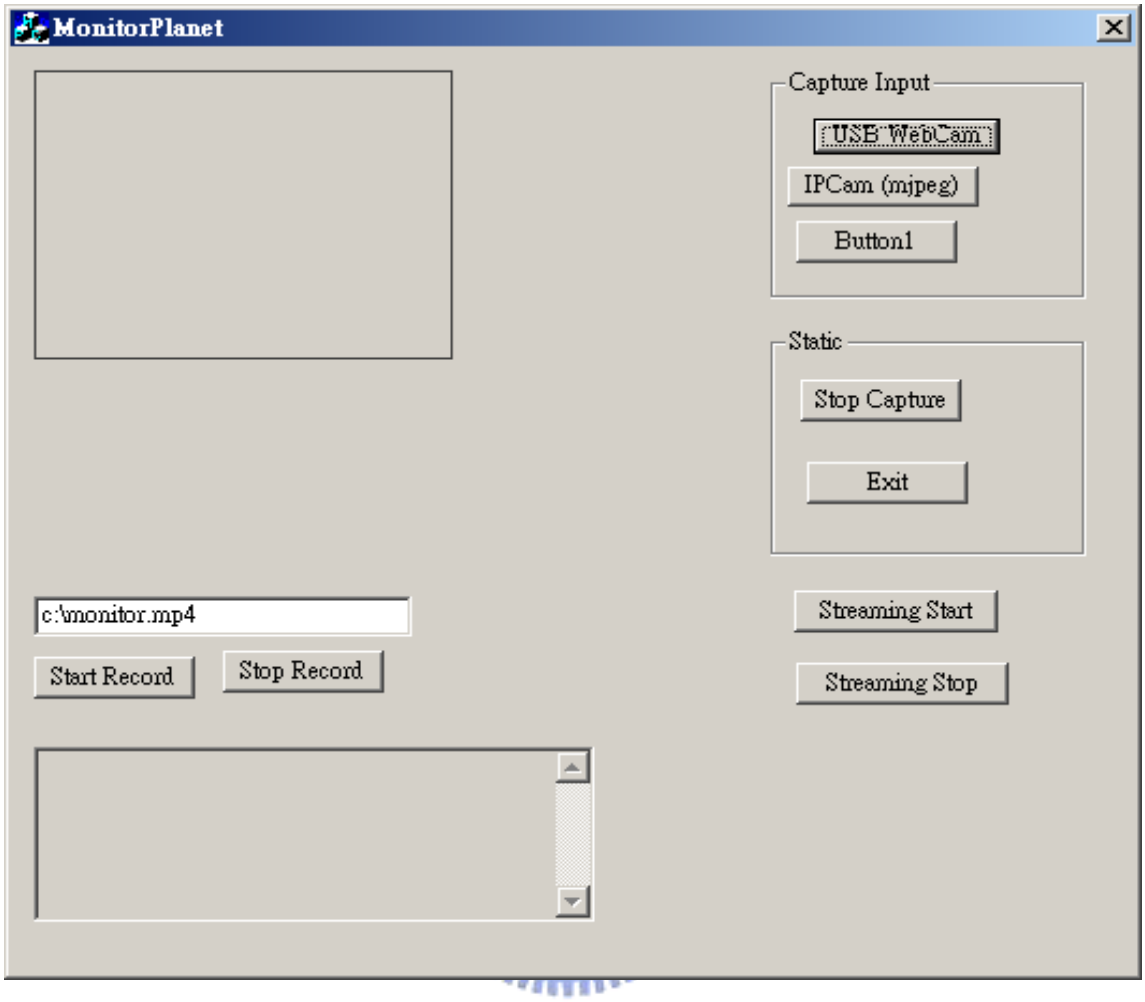


圖 37 伺服端介面

在這個介面中，在按下 USB webcam 按鈕後，攝影機拍攝到的畫像會出現在左上方，之後可以按下 start record 按鈕將影音儲存成.mp4 放在硬碟裡，按下右邊的 Streaming Start 按鈕之後則可以接受客戶端的連入。

6.2 客戶端介面

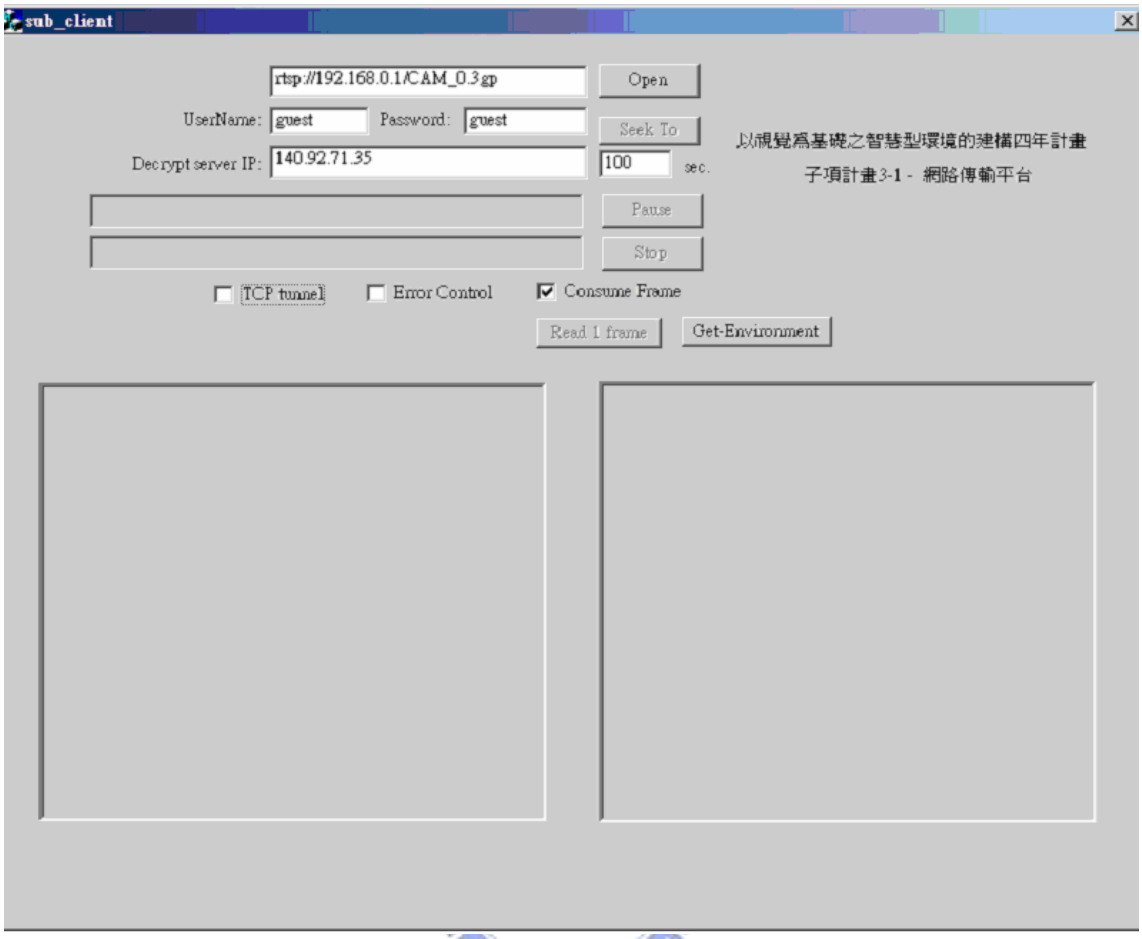


圖 38 客戶端介面

在客戶端介面中，上方是要連線的伺服端的 url，再按下 open 之後就可以連線，收到的影像會播放在左下角，聲音則從音效卡輸出。右下角是影像處理的結果，目前是單純的顯示跟第一張圖片不同的部份。

6.3 執行結果

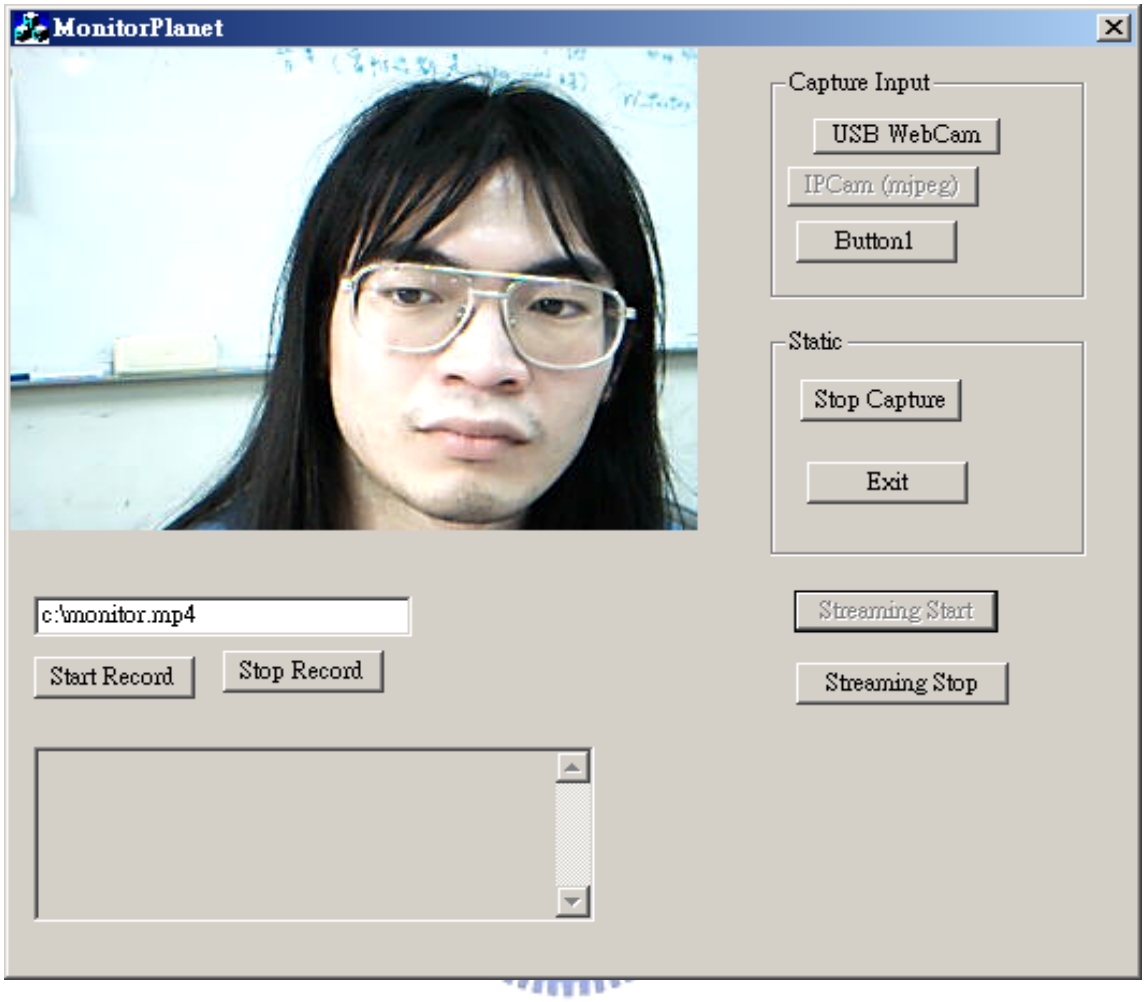

MonitorPlanet
Capture Input
USB WebCam
IPCam (mjpeg)
Button1
Static
Stop Capture
Exit
c:\monitor.mp4
Start Record
Stop Record
Streaming Start
Streaming Stop

圖 39 伺服端執行結果

圖 40 客戶端執行結果

中間第一行是表示聲音現在的狀況，收到了 303 個封包，另外遺失了 52 個封包，遺失率是 0.146，上一個遺失的片段是回朔多少個取樣修補的。回朔 0 個 sample，就是表示將前一個 frame 拿來做修補，回朔 295 是表示拿前一個 frame 再往前 295 個取樣之後的 1024 個取樣修補。

6.4 數據

在這邊我們做了全頻帶與多頻帶樣本比對的比較，多頻帶又分作全部 0 – 31 頻帶、0 – 9 頻帶，以及 0 – 4 頻帶三種。雜訊分為 matlab 7.0 產生的 AWGN 雜訊，以及 AWGN 雜訊再通過下圖的濾波器所產生的 color noise。雜訊強度則是根據每個片段的能量，將 SNR 正規化為 10dB，0dB，-10dB。

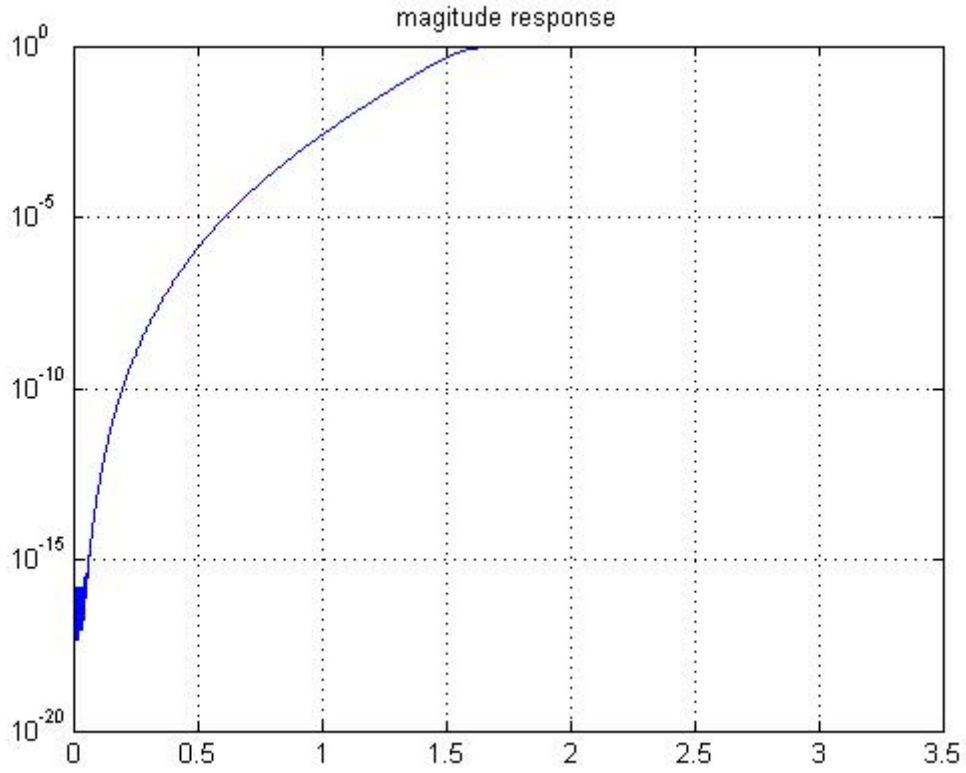
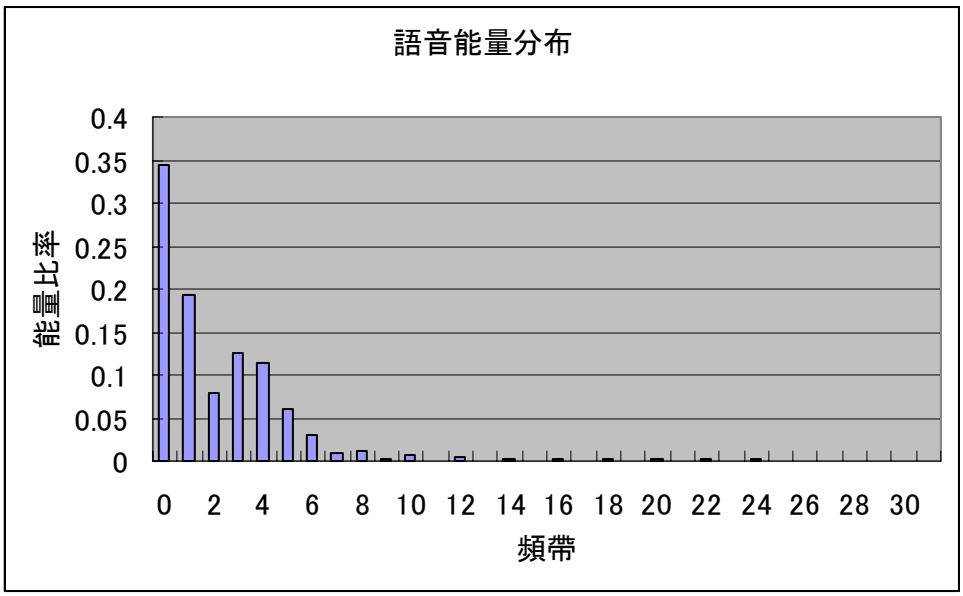


圖 41 color noise 產生用濾波器

(a)



(b)

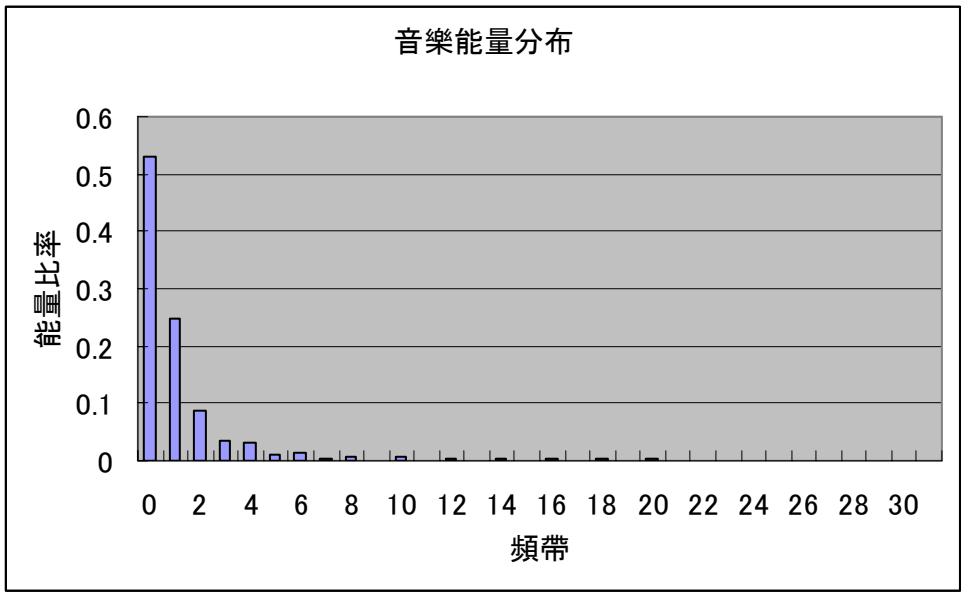


圖 42 實驗片段的能量分布

(a)

無雜訊													
聲音類型	語音							音樂					
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.38	0.36	0.34	0.37	0.35	0.37	0.34	0.61	0.43	0.31	0.42	0.30	0.

(b)

White 雜訊 SNR 10dB		
聲音類型	語音	音樂

比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.37	0.36	0.34	0.36	0.34	0.36	0.34	0.60	0.44	0.29	0.43	0.29	0.

(c)

White 雜訊 SNR 0dB													
聲音類型	語音							音樂					
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.37	0.30	0.33	0.31	0.33	0.32	0.30	0.45	0.39	0.24	0.35	0.20	0.

(d)

White 雜訊 SNR -10dB													
聲音類型	語音							音樂					
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p

相關係數	0.30	0.28	0.30	0.26	0.30	0.27	0.29	0.18	0.23	0.14	0.13	0.08	0.

(e)

color 雜訊 SNR 10dB													
聲音類型	語音							音樂					
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.37	0.37	0.34	0.37	0.35	0.37	0.34	0.60	0.44	0.31	0.44	0.31	0.

(f)

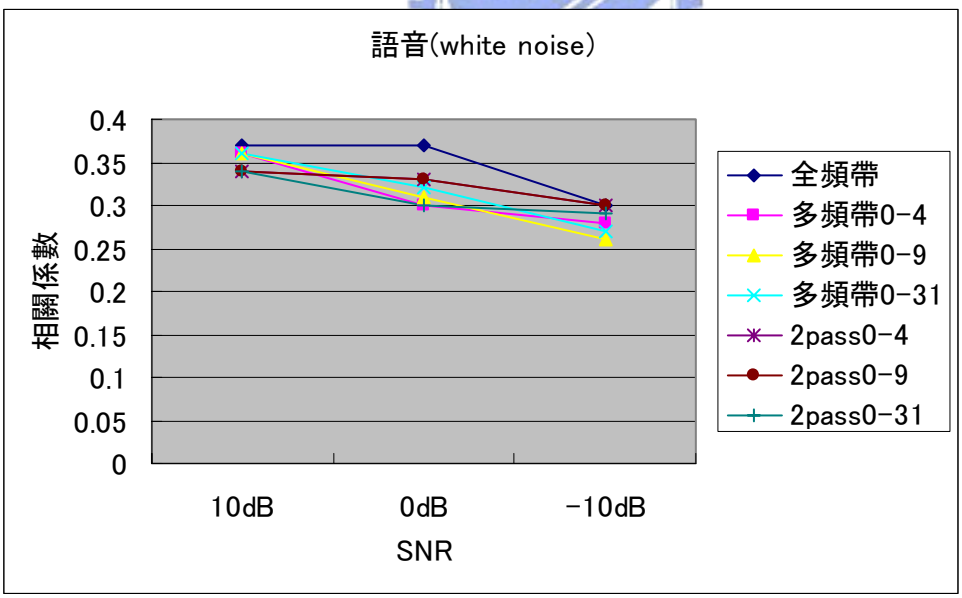
color 雜訊 SNR 0dB													
聲音類型	語音							音樂					
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.38	0.33	0.33	0.34	0.33	0.30	0.32	0.51	0.44	0.30	0.43	0.29	0.

(g)

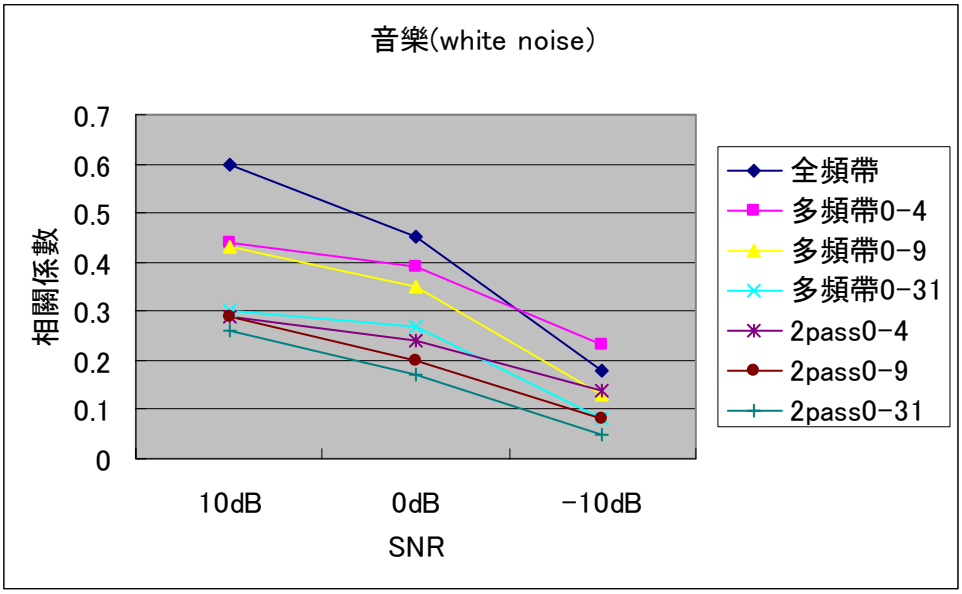
color 雜訊 SNR -10dB												
聲音類型	語音							音樂				
比對類型	全頻帶	多頻帶						全頻帶	多頻帶				
		0-4 頻帶		0-9 頻帶		0-31 頻帶			0-4 頻帶		0-9 頻帶		0-
		1pass	2pass	1pass	2pass	1pass	2pass		1pass	2pass	1pass	2pass	1p
相關係數	0.29	0.32	0.25	0.30	0.25	0.30	0.28	0.20	0.45	0.32	0.43	0.31	0.

表 2 修補的效果

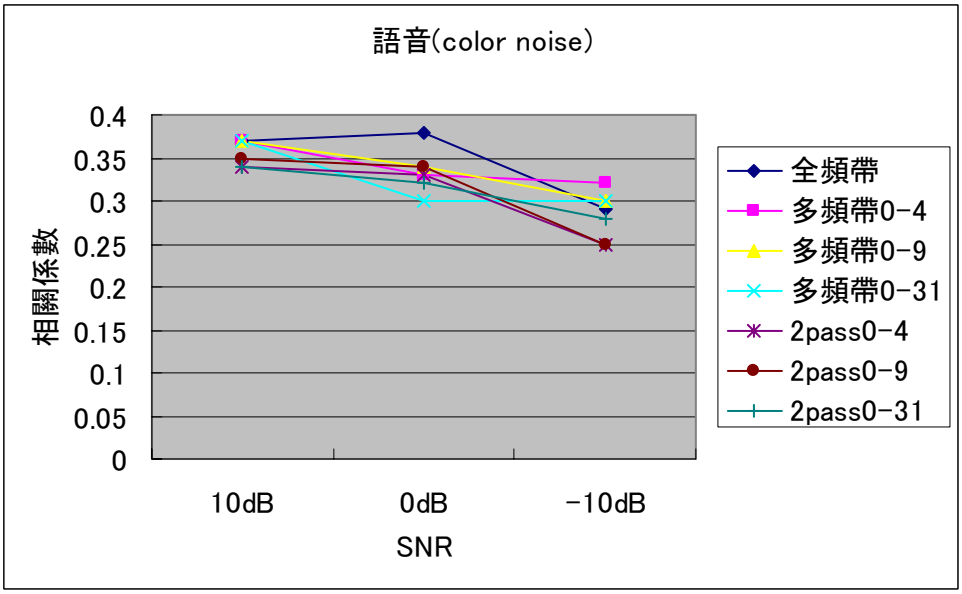
(a)



(b)


音樂(white noise)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
相關係數
10dB
0dB
-10dB
SNR
全頻帶
多頻帶0-4
多頻帶0-9
多頻帶0-31
2pass0-4
2pass0-9
2pass0-31

(c)


語音(color noise)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
相關係數
10dB
0dB
-10dB
SNR
全頻帶
多頻帶0-4
多頻帶0-9
多頻帶0-31
2pass0-4
2pass0-9
2pass0-31

(d)

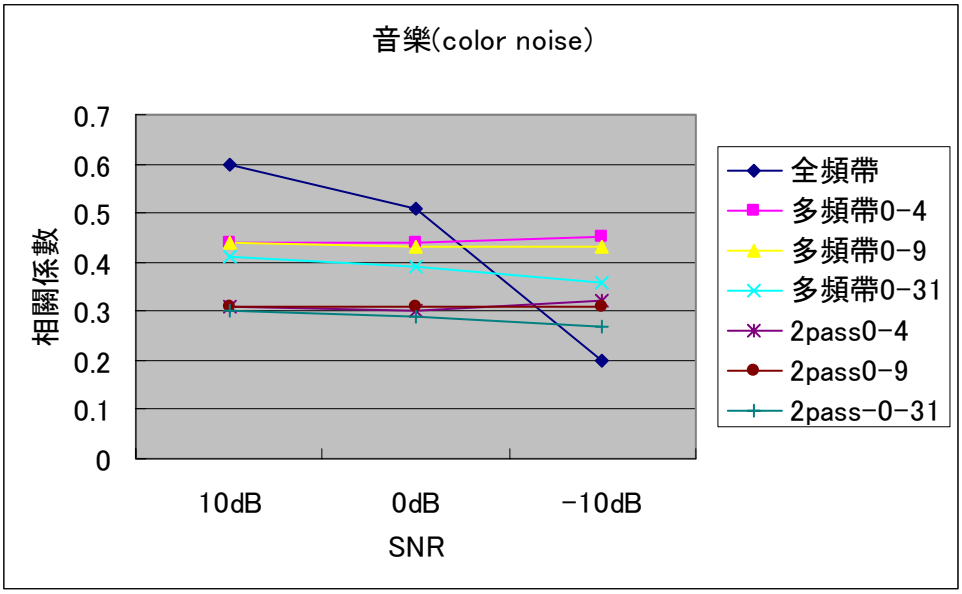


圖 43 修補效果曲線

頻帶	0-4頻帶noiseless	0-4頻帶10dB	0-4頻帶0dB	0-4頻帶-10dB	0-9頻帶noiseless	0-9頻帶10dB	0-9頻帶0dB	0-9頻帶-10dB	0-31頻帶noiseless	0-31頻帶10dB	0-31頻帶0dB	0-31頻帶-10dB
0	56	51	45	31	53	49	42	24	51	46	32	3
1	9	13	12	9	5	8	10	2	4	7	7	1
2	5	6	5	12	4	5	4	6	4	2	1	3
3	4	4	8	16	3	1	6	8	3	1	1	3
4	3	3	7	9	0	3	4	2	0	1	4	0
5	0	0	0	0	1	4	4	11	0	0	1	3
6	0	0	0	0	3	1	2	4	3	0	2	2
7	0	0	0	0	1	1	2	13	1	0	0	1

8	0	0	0	0	2	1	1	1	0	1	0	0
9	0	0	0	0	5	4	2	6	4	1	0	1
10	0	0	0	0	0	0	0	0	1	2	0	0
11	0	0	0	0	0	0	0	0	2	1	2	1
12	0	0	0	0	0	0	0	0	0	1	7	22
13	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	1	1	4
15	0	0	0	0	0	0	0	0	1	1	0	1
16	0	0	0	0	0	0	0	0	0	0	2	3
17	0	0	0	0	0	0	0	0	1	1	1	2
18	0	0	0	0	0	0	0	0	0	0	0	1
19	0	0	0	0	0	0	0	0	0	1	0	1
20	0	0	0	0	0	0	0	0	0	1	0	1
21	0	0	0	0	0	0	0	0	1	1	2	2
22	0	0	0	0	0	0	0	0	0	1	0	1
23	0	0	0	0	0	0	0	0	1	0	0	4
24	0	0	0	0	0	0	0	0	0	0	1	2
25	0	0	0	0	0	0	0	0	0	6	8	7
26	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	2	1
29	0	0	0	0	0	0	0	0	0	0	1	2
30	0	0	0	0	0	0	0	0	0	0	0	2
31	0	0	0	0	0	0	0	0	0	1	1	3

表 3 語音(white noise)多頻帶使用頻帶次數

頻帶	0-4 頻帶 noiseless	0-4 頻帶 10dB	0-4 頻帶 0dB	0-4 頻帶 -10 dB	0-9 頻帶 noiseless	0-9 頻帶 10dB	0-9 頻帶 0dB	0-9 頻帶 -10 dB	0-31 頻帶 noiseless	0-31 頻帶 10dB	0-31 頻帶 0dB	0-31 頻帶 -10 dB
0	261	257	272	249	198	228	215	90	167	172	131	13
1	103	127	131	103	78	103	97	42	65	67	51	7
2	53	50	49	61	39	38	33	10	32	24	9	2
3	51	54	31	18	36	39	20	4	32	17	8	0
4	55	35	40	92	33	24	16	12	25	14	2	1

5	0	0	0	0	28	22	28	11	19	7	4	0
6	0	0	0	0	24	21	39	137	16	10	10	6
7	0	0	0	0	27	24	71	217	17	7	10	1
8	0	0	0	0	39	23	4	0	16	10	0	0
9	0	0	0	0	21	1	0	0	14	0	0	0
10	0	0	0	0	0	0	0	0	14	12	2	1
11	0	0	0	0	0	0	0	0	12	1	0	0
12	0	0	0	0	0	0	0	0	5	20	85	337
13	0	0	0	0	0	0	0	0	14	0	0	0
14	0	0	0	0	0	0	0	0	7	6	1	0
15	0	0	0	0	0	0	0	0	5	2	0	0
16	0	0	0	0	0	0	0	0	8	6	1	0
17	0	0	0	0	0	0	0	0	14	1	0	0
18	0	0	0	0	0	0	0	0	6	2	0	0
19	0	0	0	0	0	0	0	0	17	0	1	1
20	0	0	0	0	0	0	0	0	1	10	5	2
21	0	0	0	0	0	0	0	0	9	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	2	0	0	0
24	0	0	0	0	0	0	0	0	1	1	0	0
25	0	0	0	0	0	0	0	0	0	127	203	152
26	0	0	0	0	0	0	0	0	1	0	0	0
27	0	0	0	0	0	0	0	0	1	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	1	0	0
30	0	0	0	0	0	0	0	0	1	2	0	0
31	0	0	0	0	0	0	0	0	1	4	0	0

表 4 音樂(white noise)多頻帶比對使用頻帶次數

頻帶	0-4 頻帶 noiseless	0-4 頻帶 10dB	0-4 頻帶 0dB	0-4 頻帶 -10dB	0-9 頻帶 noiseless	0-9 頻帶 10dB	0-9 頻帶 0dB	0-9 頻帶 -10dB	0-31 頻帶 noiseless	0-31 頻帶 10dB	0-31 頻帶 0dB	0-31 頻帶 -10dB
0	56	53	51	40	53	48	46	36	51	47	43	27
1	9	7	8	10	5	3	4	5	4	1	4	2

2	5	7	3	8	4	4	1	6	4	4	1	1
3	4	5	10	7	3	4	4	2	3	2	0	2
4	3	5	5	12	0	3	4	9	0	1	2	4
5	0	0	0	0	1	2	3	3	0	1	1	0
6	0	0	0	0	3	3	4	4	3	2	1	0
7	0	0	0	0	1	2	3	6	1	1	1	1
8	0	0	0	0	2	4	3	2	0	3	0	0
9	0	0	0	0	5	4	5	4	4	2	1	1
10	0	0	0	0	0	0	0	0	1	1	2	1
11	0	0	0	0	0	0	0	0	2	3	1	2
12	0	0	0	0	0	0	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	1	2	1
14	0	0	0	0	0	0	0	0	0	0	0	4
15	0	0	0	0	0	0	0	0	1	0	2	1
16	0	0	0	0	0	0	0	0	0	0	2	2
17	0	0	0	0	0	0	0	0	1	0	0	1
18	0	0	0	0	0	0	0	0	0	0	2	1
19	0	0	0	0	0	0	0	0	0	2	1	5
20	0	0	0	0	0	0	0	0	0	1	0	2
21	0	0	0	0	0	0	0	0	1	0	1	1
22	0	0	0	0	0	0	0	0	0	1	1	3
23	0	0	0	0	0	0	0	0	1	1	0	2
24	0	0	0	0	0	0	0	0	0	1	1	1
25	0	0	0	0	0	0	0	0	0	0	1	3
26	0	0	0	0	0	0	0	0	0	0	2	2
27	0	0	0	0	0	0	0	0	0	0	0	1
28	0	0	0	0	0	0	0	0	0	0	2	0
29	0	0	0	0	0	0	0	0	0	0	0	3
30	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	1	3	3

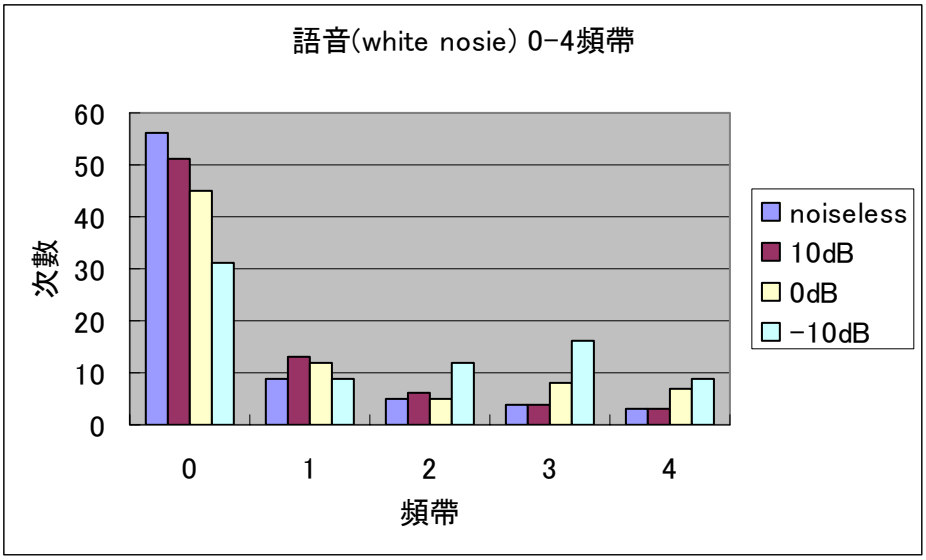
表 5 語音(color noise)多頻帶比對使用頻帶次數

	0-4 頻帶	0-4 頻帶	0-4 頻帶	0-4 頻帶	0-9 頻帶	0-9 頻帶	0-9 頻帶	0-9 頻帶	0-31 頻帶	0-31 頻帶	0-31 頻帶	0-31 頻帶
頻	nois	10dB	0dB	-10	nois	10dB	0dB	-10	nois	10dB	0dB	-10

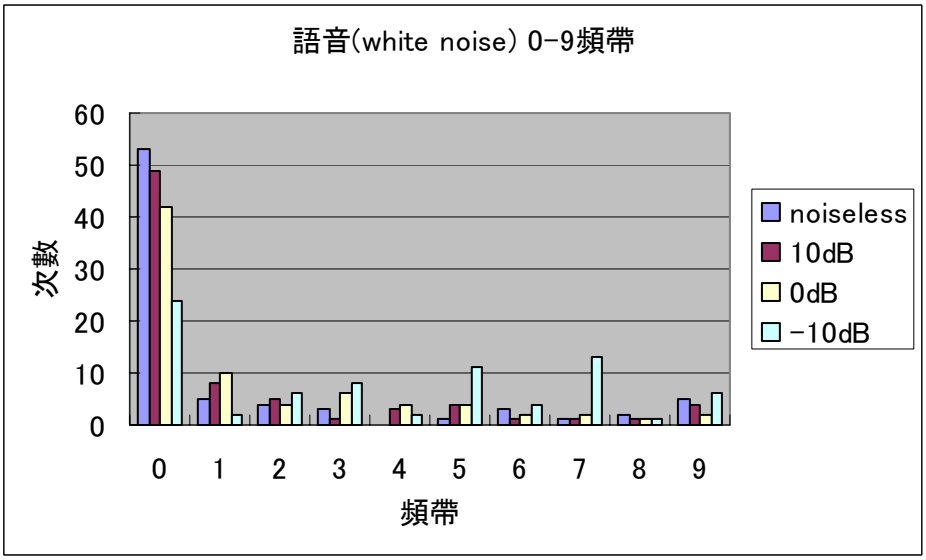
帶	eles s			dB	eles s			dB	eles s			dB
0	261	257	259	272	198	196	205	242	167	161	172	200
1	103	112	120	117	78	79	96	106	65	63	71	83
2	53	50	45	60	39	40	28	39	32	27	18	23
3	51	49	49	42	36	35	27	26	32	25	21	16
4	55	55	50	32	33	34	24	19	25	21	14	10
5	0	0	0	0	28	26	29	20	19	23	18	10
6	0	0	0	0	24	30	38	28	16	22	26	14
7	0	0	0	0	27	23	23	19	17	15	15	4
8	0	0	0	0	39	43	36	20	16	28	23	11
9	0	0	0	0	21	17	17	4	14	10	11	1
10	0	0	0	0	0	0	0	0	14	15	18	5
11	0	0	0	0	0	0	0	0	12	13	6	1
12	0	0	0	0	0	0	0	0	5	14	5	4
13	0	0	0	0	0	0	0	0	14	5	0	0
14	0	0	0	0	0	0	0	0	7	10	4	0
15	0	0	0	0	0	0	0	0	5	0	0	0
16	0	0	0	0	0	0	0	0	8	12	3	9
17	0	0	0	0	0	0	0	0	14	0	0	0
18	0	0	0	0	0	0	0	0	6	12	71	102
19	0	0	0	0	0	0	0	0	17	0	0	0
20	0	0	0	0	0	0	0	0	1	8	5	2
21	0	0	0	0	0	0	0	0	9	5	1	1
22	0	0	0	0	0	0	0	0	0	4	0	0
23	0	0	0	0	0	0	0	0	2	1	0	0
24	0	0	0	0	0	0	0	0	1	10	13	15
25	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	1	5	8	12
27	0	0	0	0	0	0	0	0	1	2	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	1	0	0	0
30	0	0	0	0	0	0	0	0	1	4	0	0
31	0	0	0	0	0	0	0	0	1	8	0	0

表 6 音樂(color noise)多頻帶比對使用頻帶次數

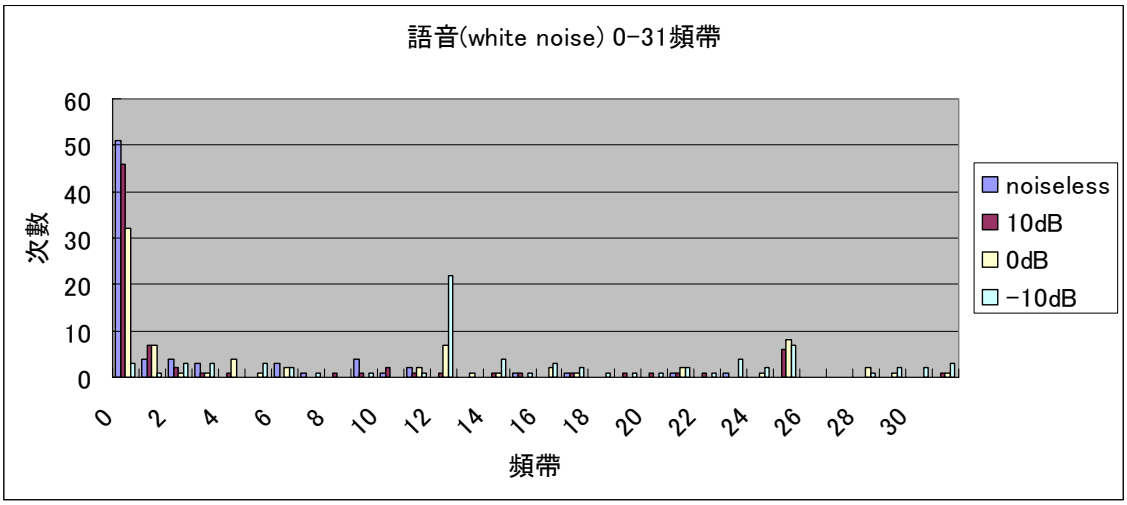
(a)


語音(white nosie) 0-4頻帯
60
50
40
30
20
10
0
次數
0
1
2
3
4
頻帯
noiseless
10dB
0dB
-10dB

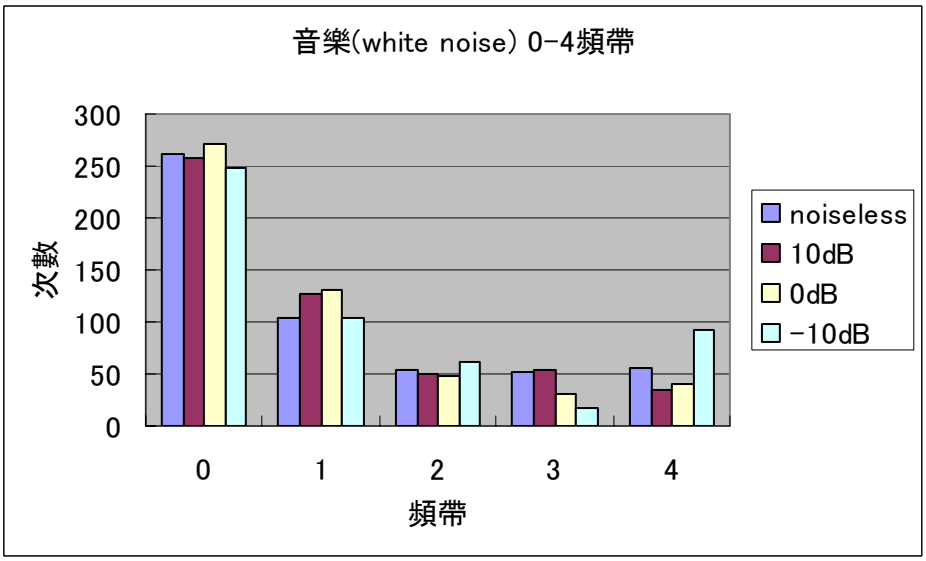
(b)


語音(white noise) 0-9頻帯
60
50
40
30
20
10
0
次數
0
1
2
3
4
5
6
7
8
9
頻帯
noiseless
10dB
0dB
-10dB

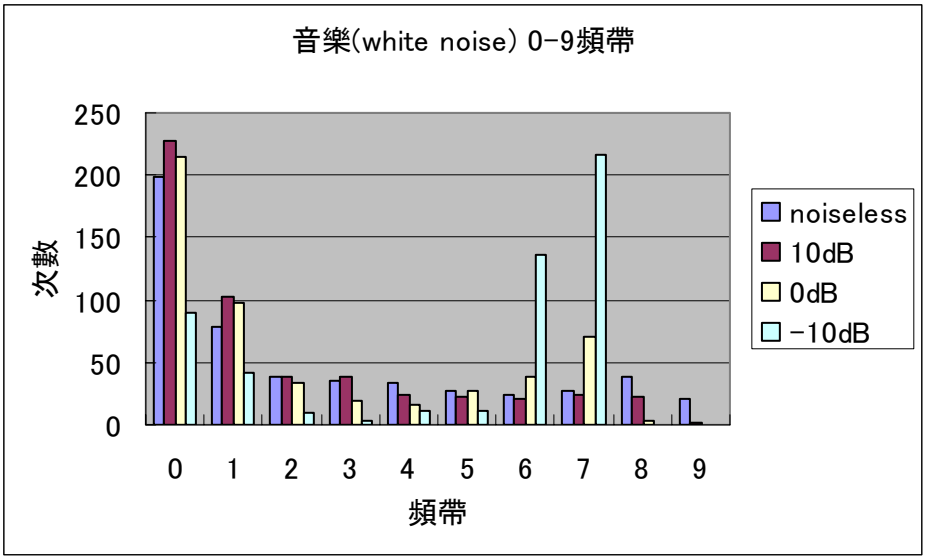
(c)


語音(white noise) 0-31頻帶
60
50
40
30
20
10
0
次數
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
頻帶
noiseless
10dB
0dB
-10dB

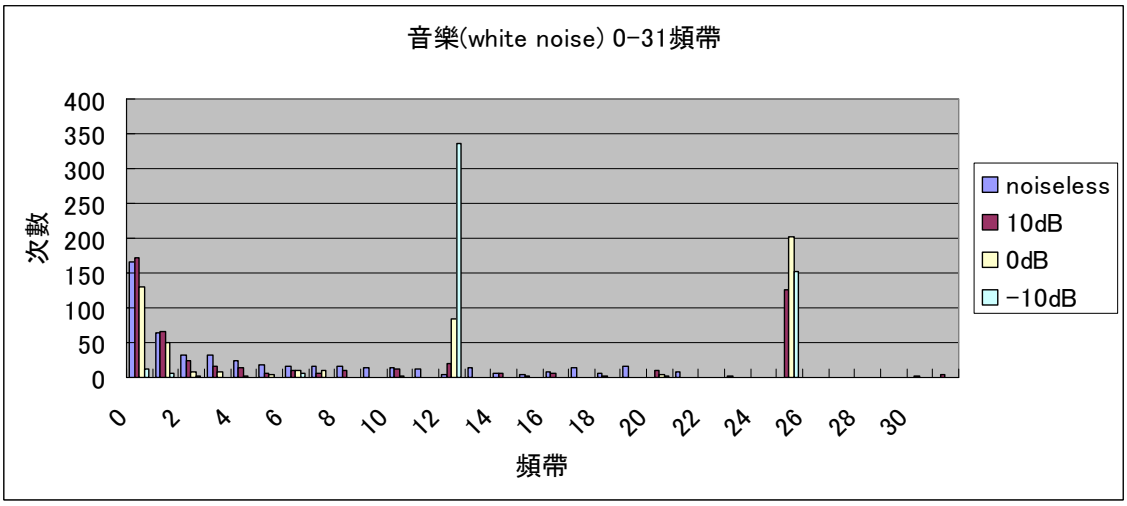
(d)


音樂(white noise) 0-4頻帶
300
250
200
150
100
50
0
次數
0
1
2
3
4
頻帶
noiseless
10dB
0dB
-10dB

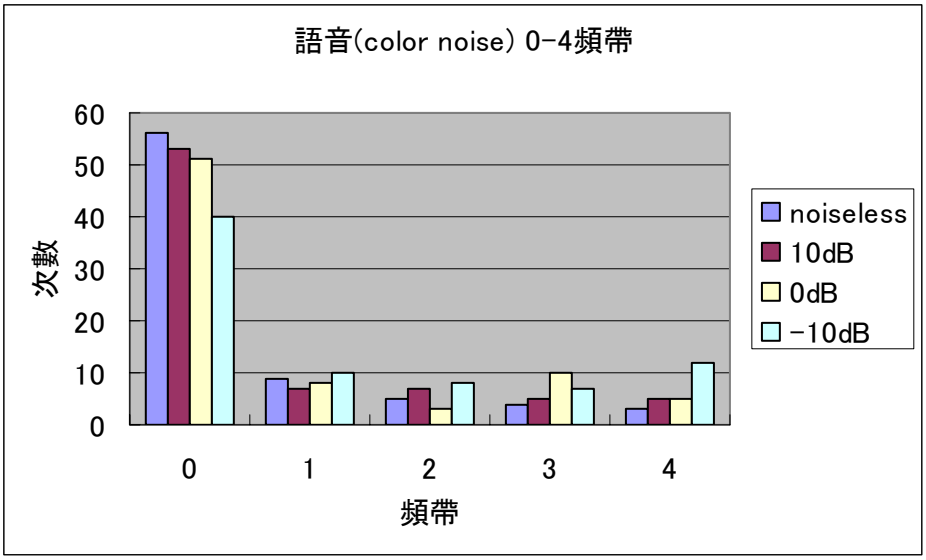
(e)


音樂(white noise) 0-9頻帶
250
200
150
100
50
0
次數
0 1 2 3 4 5 6 7 8 9
頻帶
noiseless
10dB
0dB
-10dB

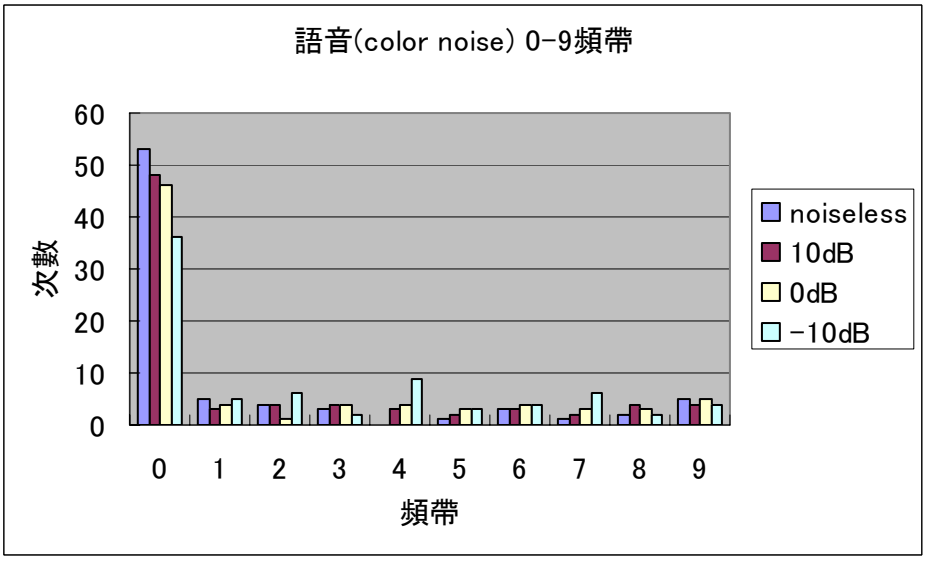
(f)


音樂(white noise) 0-31頻帶
400
350
300
250
200
150
100
50
0
次數
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
頻帶
noiseless
10dB
0dB
-10dB

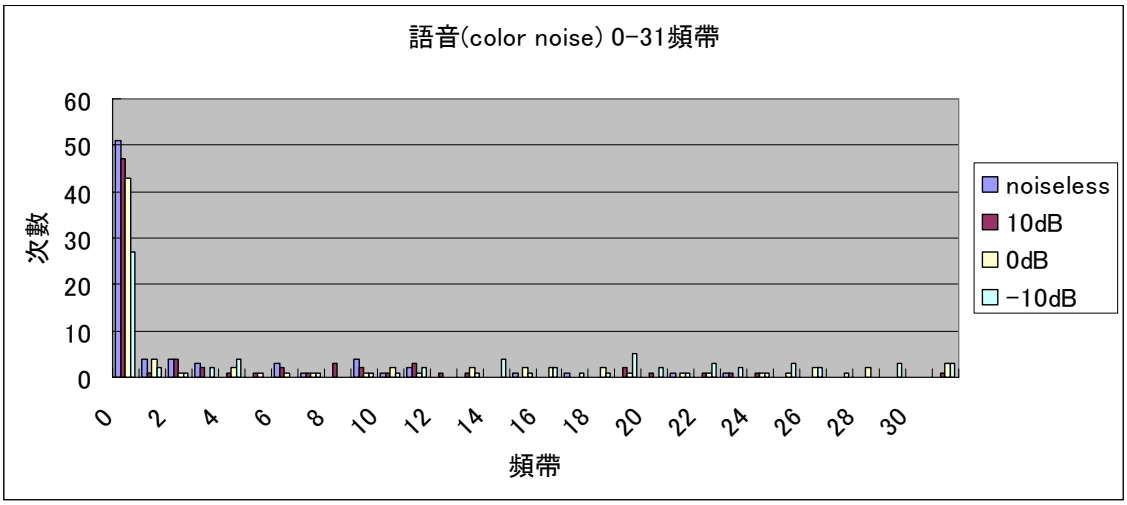
(g)


語音(color noise) 0-4頻帶
60
50
40
30
20
10
0
次數
0
1
2
3
4
頻帶
noiseless
10dB
0dB
-10dB

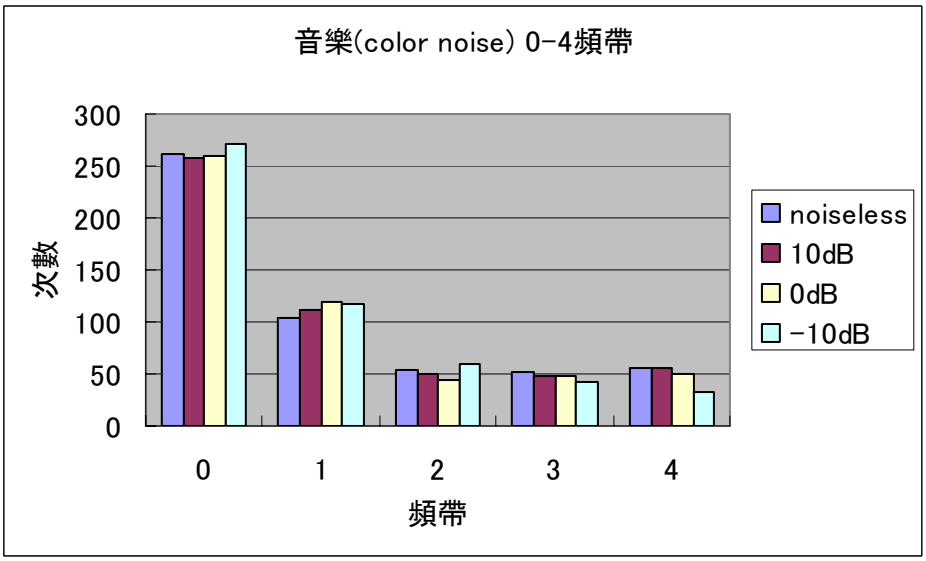
(h)


語音(color noise) 0-9頻帶
60
50
40
30
20
10
0
次數
0
1
2
3
4
5
6
7
8
9
頻帶
noiseless
10dB
0dB
-10dB

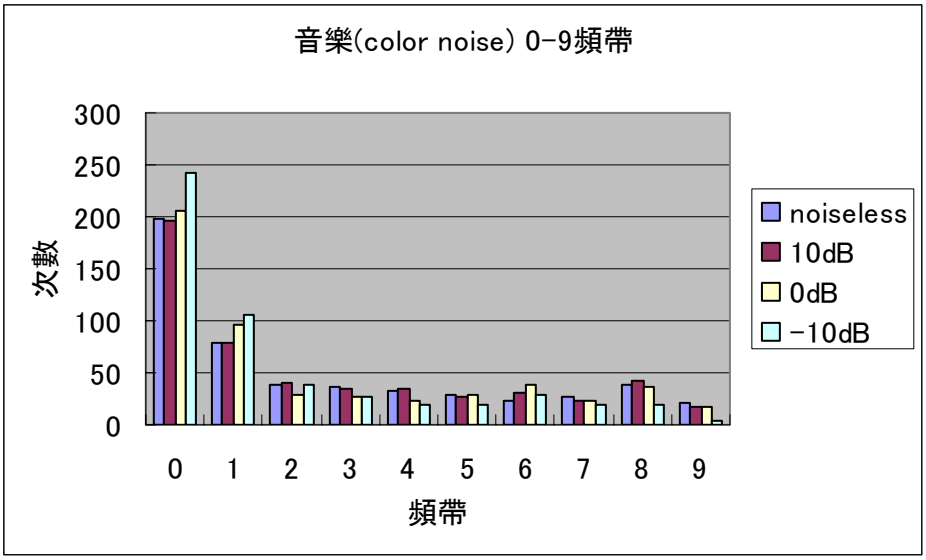
(i)


語音(color noise) 0-31頻帶
60
50
40
30
20
10
0
次數
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
頻帶
noiseless
10dB
0dB
-10dB

(j)


音樂(color noise) 0-4頻帶
300
250
200
150
100
50
0
次數
0
1
2
3
4
頻帶
noiseless
10dB
0dB
-10dB

(k)



(1)

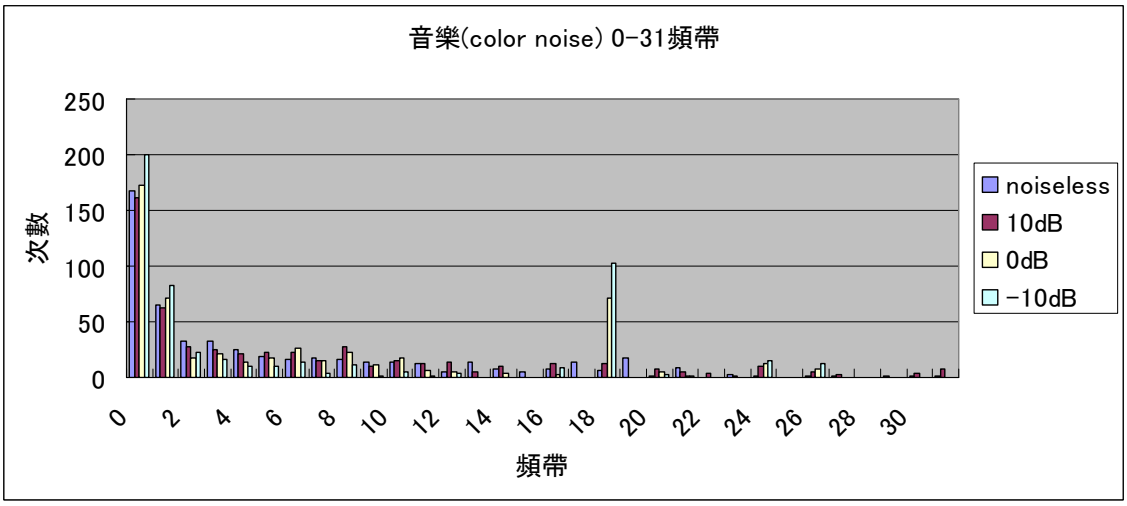


圖 44 多頻帶使用頻帶分布

(a)

0-4 頻帶，和無雜訊的結果相比自動選擇頻帶的變化率												
類型	語音						音樂					
雜訊類型	White			Color			White			Color		
SNR	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db
變	0.14	0.36	0.62	0.12	0.33	0.50	0.25	0.41	0.60	0.07	0.17	0.37

化比率												

(b)

0-9 頻帶，和無雜訊相比自動選擇頻帶的變化率												
類型	語音						音樂					
雜訊類型	White			Color			White			Color		
SNR	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db
變化比率	0.21	0.43	0.69	0.18	0.34	0.56	0.42	0.60	0.84	0.20	0.34	0.51

(c)

0-31 頻帶，和無雜訊相比自動選擇頻帶的變化率												
類型	語音						音樂					
雜訊類型	White			Color			White			Color		
SNR	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db	10db	0db	-10db
變化比率	0.32	0.58	0.97	0.29	0.42	0.68	0.62	0.77	0.97	0.40	0.53	0.64

表 7 和無雜訊相比，選擇的頻帶的變化比率

6.5 討論

從數據中可以看出，white noise 時全頻帶比對總是比多頻帶比對來的好，不過在 color noise 時，遇到音樂的話，全頻帶比對的品質下降快速，多頻帶比對可能會比全頻帶比對來的出色。

在多頻帶的頻帶選擇上，可以看出當雜訊越來越大時，選擇的頻帶會越來越

往頻率高的地方走，這應該是雜訊主導了比對的結果，高頻的地方信號比較弱，所以雜訊相對之下很強，就主導了比對。所以在只比對低頻的頻帶時常常得到比較好的結果。

一階段比對的時候，因為我們是降低取樣率的比對，回朔取樣數都是 32 的倍數，可能因此造成了問題，所以我們做了兩階段的比對，不過在兩階段比對方面，並沒有比較出色的表現。

7. 結論

在這邊我們成功的建立起了一個能夠即時串流傳輸聲音的系統，這個系統符合 RTP 規範，而且能夠對聲音封包的遺失進行修補。

在修補方面，從結果可以看出，全頻帶的樣本比對跟多頻帶的樣本比對夠有所長，該要視使用者需要來選擇。雜訊小的環境下全頻帶比對有著良好的結果，不過當面對 color noise 時，多頻帶比對則比較能夠對抗雜訊。使用者的環境如果是在多雜訊，而且雜訊經常會集中在某些頻率附近時，就可以選擇多頻帶比對。

未來可以嘗試著做多頻段的全取樣率的比對，或是在多頻段比對的時候，針對每個頻段給予不同的權重來決定是否選擇此頻段。

Reference

[1] H. Schulzrinne, S. Casner, 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.

[2] ISO/IEC JTC1/SC29/WG11 MPEG, IS11172-3 “Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5Mbit/s, Part 3: Audio” 1992.

[3] ISO/IEC JTC/SC29/WG11 MPEG, International Standard ISO/IEC 14496-3

“Advanced Audio Coding”, 1999
[4] H. Sanneck: A. Stenger, K. Ben Younes, B. Girod, "A New Technique for Audio Packet Loss Concealment," IEEE Global Internet 1996, Dec. 1996 pp. 48-52. cited by other .
[5] Osman EROGUL and Irfan KARAGOZ, “Time-scale modification of speech signals for language-learning impaired children.” IEEE 2nd International Biomedical Engineering Days, IBED-98, 1998; 33.
[6] C. J. Weinstein and J. W. Forgie, “Experience with speech communication in packet networks,” IEEE J. Selected Area Commun., vol.SAC-1, pp. 963-980, Dec. 1983.
[7] D.J. Goodman, G.B. Lockhart, O.J. Wasem, and W.C. Wong, “Waveform substitution techniques for recovering missing speech segments in packet voice communications,” IEEE Trans. Acoust. Speech Signal Process., vol.ASSP-34, no.6, pp.1440－1448, Dec. 1986.
[8] Yumi TAKIZAWA, Shinichi SATO, and Atsushi PUKAZAWA, “A new speaker-independent voice recognition scheme for voice dialing,” IEEE 1987 p. 547-551.
[9] O. J. Wasem, D. J. Goodman, C. A. Dvorak & H. G. Page, The Effect of Waveform Substitution on the Quality of PCM Packet Communications, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 26, No. 3, March 1988.