網際網路上的 MP3 內涵式搜尋引擎之設計與實作

王俊淵, 陳信修, 姚邦佳, 葉億真, 陳孟森, 蔡伯俊, 劉志俊
中華大學資訊工程系
新竹市東香里東香 30 號
ccliu@chu.edu.tw

摘要

由於人類文明的快速進步及使用者需求的多元化，造就了多媒體應用與網際網路的快速蓬勃發展，因此每一個使用者皆可以透過網際網路去獲取更多的音訊、視訊及影像等多媒體的相關資料，使得多媒體資訊變的隨手可得。為了讓使用者能方便地在大量的多媒體資料中，擷取到自己所喜愛的資訊，必須提供使用者以一定的查詢條件來過濾出所想要的多媒體資料，並提供更有效的多媒體資料儲存管理功能。因此多媒體資料庫的相關研究探討成了學術界與產業界研究的焦點之一。

本文最主要的部分就是完成一個在網際網路上使用之 MP3 音樂資料庫和搜尋引擎的設計與實作。我們首先建立一 6000 首 MP3 歌曲的 MP3 音樂資料庫並提供傳統的的關鍵字查詢和分類查詢。此外，我們根據之前所提出的 MP3 內涵式查詢理論[1]，在視窗環境下，以 C++語言實作一 MP3 搜尋引擎。在文中，我們詳述了系統的架構，包含查詢樣本錄製、壓縮、特徵值擷取、MP3 索引建立、相似性比對程式等的設計與實作方式。

關鍵詞：MP3 音樂資料庫

一、簡介

電腦數位音樂大致上分為二類，一為將原本的類比信號經由取樣後忠實的將其聲音記錄下來的方式，例如：Wav 檔便是其中一種。另一為 MIDI(Musical Instrument Digital Interface)格式，它是將歌曲音符的音階記錄下來，再透過電腦內部的音源產生器將歌曲合成還原出來。其中 Wav 的儲存空間大小與我們所使用的取樣頻率和取樣解析度有密切的關係：取樣頻率越大，所需的儲存空間越大;取樣解析度越大;所需的空間也越大。由於其所需耗費的儲存空間太多，為了要解決這個問題，MP3 數位音樂檔案格式便因應而生。

在電腦中最常見的聲波格式資料就屬於 Wave 資料，其一般以*.wave 檔儲存，但是 wave 檔格式所需要的儲存空間以一分鐘的聲波資料就將近需要 10 幾 MB 的儲存空間大小，相當消耗儲存空間，為了解決此問題故有 MP3 的產生，因 MP3 是一種經過壓縮的 wave 檔，須要下載撥放器及 MP3 檔案才能夠撥放。

這種壓縮的技術其實是一種失真壓縮，但由於 586 的 CPU 與軟體相互支援配合下，使音色幾乎可以和 CD 的音色相媲美，收聽者最好有 586 級、16MB 記憶體以上的電腦，若您是 486 級電腦使用者，在聆聽時儘量不要打開其它們何的軟體，以免作業系統無法負荷而當機。

MP3 是利用 MPEG Audio Layer 3 的技術，將聲音以 1:10 甚至 1:12 的壓縮率，變成容量較小的檔案，不過在人耳聽起來，卻沒有什麼不同。MP3 勢將原有的聲波資料經過壓縮成 MP3 的檔案格式，經由此特殊的方法壓縮可以將一個將近 50 到 60MB 的 Wave 資料，轉變成只需要 4MB 多的資料而且音質幾乎和 CD 音樂音效一樣，將近相差十倍的資料儲存空間，所以一片 MP3 的音樂光碟可以將十幾片的 CD 音樂放到裡面，若以一片 CD 音樂專輯可以播放 60 分鐘來算，一片 MP3 的音樂 CD 就將近可以放入 10 幾小時的音樂資料，就像一個小型的音樂櫃一樣。當然這是一種失真壓縮(就像圖形檔 JPG 一樣)，而經過壓縮程序之後，所壓成的資料也就更多了。

MP3 音樂資料的出現，不僅只有縮短了下載 Audio 檔案的傳輸時間，也因此擴大了音訊應用方面的領域，如現在最流行的網路廣播(Internet Radio)、網路虛擬唱片公司及網路點歌(Audio on demand)等。又 MP3 的製作及撥放工具方面，已經有越來越多的免費軟體出現且有的都支援各種平台。 如 WinAmp 等都助長了 MP3 快速發展的原因之一。此外，也有許多廠商紛紛推出了 MP3 數位隨身聽，有此

本論文研究為國科會補助之研究成果。計畫編號 NSC-90-2213-E-216-010

可以得知 MP3 對於整個音訊領域的影響有多大，未來勢必有更多的 MP3 應用將會出現。

MP3 音訊資料最主要的取得管道就是透過網際網路的方式取得，但就目前而言，所有的 MP3 網站並沒有提供完整的 MP3 音樂資料的搜尋，且其搜尋採用傳統方式，將要找的資訊以輸入關鍵字的方式到資料庫中去搜尋，系統將使用者所想要找的資料加以分類，如男歌手、女歌手、團體類型等，之後在繼續將各類型往下依歌手姓氏或團體名稱作分類，最後才將該歌手或團體所出過的專輯資料放在該歌手或團體的目錄之下。

此次最主要的部分就是完成一個 MP3 音樂資料庫和搜尋引擎，除了將現有的搜尋引擎的關鍵字查詢和分類查詢外，另外還要研究如何撰寫一個 MP3 Query Tool，將使用者所哼的聲音樣本至資料庫中和 MP3 音樂作比對，比對方面得用到的技術就是擷取 MP3 音樂的特徵值去和使用者所哼的聲音之特徵值去做比對，這部分可能是我們得多花心思去研究的地方。

在本文中試著對 MP3 的內涵作分析，找出 MP3 的特徵值並對其做索引，並根據 MP3 特徵值建立 MP3 資料庫，最後可讓使用者輸入查詢範例，依據使用者查詢範例和 MP3 資料庫裡的資料作相似性的比對，找出使用者想要的 MP3 物件出來。另外，本論文報告所提出的方法也可以應用在隨選音樂(Music On-Demand)上：MP3 搜尋引擎，以及網路點歌等。

在本文中我們預期完成讓使用者只要隨口哼唱出一段絃律或歌詞不需記取歌名與原唱者，便能在 MP3 網際網路資料庫上輕鬆的找到自己所想要的歌曲。並且成立一個全台灣甚至全世界最大的中文 MP3 音樂資料庫。

二、相關研究

2.1 MPEG 1 音訊編碼原理

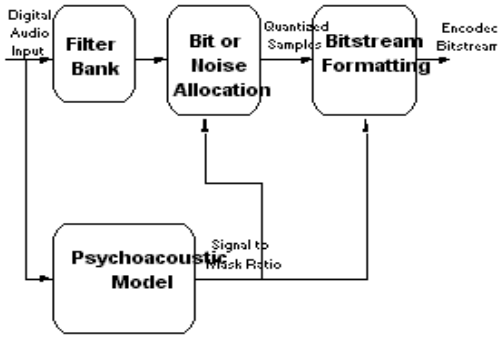


圖 1 MPEG 1 音訊編碼流程圖

圖 1 顯示 MPEG 1 音訊編碼的流程共有 4 個模組(Block)： 濾波器組(Filter Bank)可以將輸入的數位信號分割成許多不同子頻帶(Subband)的信號，此時原本輸入的數位信號也同時輸入至心裡聽覺模型(Psychoacoustic Model)這個模組，心理聽覺模型會根據人類聽覺的一些特性決定哪一些資料是可以不要的，之後便將這樣的資訊傳給位元/雜訊配置(Bit/Noise Allocation)模組，位元/雜訊配置模組會根據心裡聽覺模型所傳來的資訊及濾波器組所分割出不同頻帶的音訊資料，做適當的資料配置(將人類聽不到的音訊資料去掉)、量化的動作(也就是決定要用多少個位元去表示)，最後將經過量化的樣本，經過位元流封裝(Bit Stream Formatting)模組將其包裝成一個一個的 MP3 框架(Frame)格式，最後輸出整個編碼後的音訊格式.

2.2 MPEG 1 音訊解碼原理

在解碼方面，圖 2 為音訊解碼流程圖，其包含了 3 個區塊：框架拆解(Frame Unpacking)這個區塊負責將原本的 MP3 檔案以每個框架為單位拆解成量化後的樣本及一些輔助的資料，接著還原(Reconstruction) 區塊會將量化後的樣本還原成原本的頻率樣本(Frequency Samples)，最後經由反轉映射(Inverse Mapping)將頻率樣本轉換成時間樣本(Time Samples)，輸出解碼後的音訊資料。

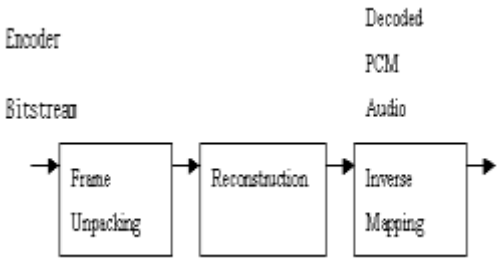


圖 2 MPEG1 音訊解碼流程圖

濾波器組與修正式餘弦轉換

我們知道過濾器的用意再於將一個訊號裡的某些頻率的訊號過濾出來，所謂的濾波器組便是由一堆的濾波器所組成的，濾波器組的功用為做時間領域-頻率領域的轉換(Time-Frequency Mapping)，Layer 1,2 用的濾波器組為稱為多相位濾波器組(Polyphase Filter Bank)，他是由 32 個頻帶濾波器所組成的，所以多相位濾波器組可以將輸入的訊號分割成 32 個相同寬度的頻帶(Frequency Bands)

信號。

Layer 3 除了用到多相位濾波器組，他還用到了修正式餘弦轉換(**M**odified **D**iscrete **C**osine **T**ransform)。MDCT 的用意在於將多相位濾波器組的輸出，再分成更細的頻帶信號。經由多相位濾波器組所分割出頻帶訊號經過 MDCT，可以在細分成 18 個頻帶信號，這樣的用意在於可以提供較好的頻譜解析度(Frequency Resolution)，進而找出因為只用多相位濾波器組，所造成的訊號重疊誤差。圖 3 為其示意圖：

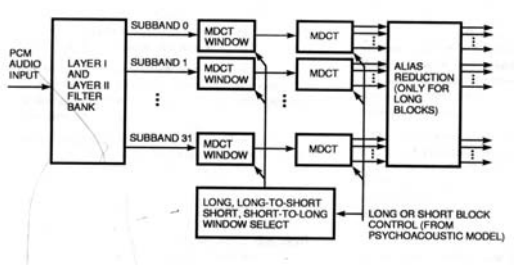


圖 3 MDCT 架構圖

綜合以上所述，在此，我們所使用的 MP3 特徵值擷取來源有二：一是取至多相位濾波器的輸出，既聲音在 32 個頻帶的係數。另一是取自修正式餘弦轉換的輸出，既 MDCT 係數。

2.3 MP3 的壓縮原理

在 MPEG-1 的標準中，將聲音訊號的壓縮標準分為三個層級，分別是 MPEG Layer 1、MPEG Layer 2 與 MPEG Layer 3。影音光碟就有採用 Layer 2 的標準，至於 MP3 就是 MPEG Layer 3 的產物了。更進一步來說，MP3 就是把現在的 CD 音樂檔，以壓縮的方式儲存，透過 CPU 強大的運算能力，以軟體解壓縮的方式，就可以在電腦上聽到好聽的音樂了。

至於這其中的壓縮效果有多好? 我們可以如此計算，普通的 CD 音樂大約是 44.1KHz 的頻率、十六位元取樣，平均每分鐘音樂就要花掉 44100*16*2 stereo*60 的容量，大約是 10MB 的儲存空間。以目前每片光碟 650MB 的容量來說，一片 CD 的儲存量約在 65 到 75 分鐘之間。

MP3 就是將這些樂曲透過壓縮的方式，增加更多的儲存量。由於 MP3 的壓縮大約在 10 到 12 倍之間，一分鐘的樂曲透過 MP3 壓縮，只要 1MB 左右的儲存空間，換言之每片光碟可以儲存 650 到 750 分鐘的音樂，更重要的是，即使壓縮比如此驚人，音樂的品質依然直追 CD，因為利用人類聽覺遮蔽的緣故，以現在一般個人電腦 CPU 的速度解壓 MP3 時，人類聽覺沒辦法分辨壓縮後的有所不同，讓使用者不須為了追求高容量，而犧牲了聽的品質。

MPEG/audio 的壓縮，其 Sampling rate 可分為 32KHz、44.1KHz、48KHz，支援的聲道有 monophonic，dual-monophonic，stereo mode，joint-stereo mode，Error detection 為 CRC error detection code 還有 Ancillary data。其主要是利用人類聽覺系統在某些情況下，會產生聽覺的 mask 而無法分辨出量化的雜訊，根據實驗亦發現人類聽覺有一個極限，也就是人類所能聽到的聲音頻率約為 20Hz 到 20KHz 之間，critical band 並不能完整呈現出人類聽覺系統的聽覺特性，因為人類聽覺系統依據頻率來分辨聲音能量，所以任何頻率的雜訊遮罩只和其限定頻寬內附近的信號能量有關。MPEG/audio 將聲音信號分配成接近 critical band 的 subband，然後依據每一個 subband 的聽覺量化雜訊程度來量化。最有效的壓縮，即是將不需要的聽覺量化雜訊移除。也就是我們可以將一大部份人類聽覺系統所無法察覺的資料移除，將人耳聽不到或不易辨認部份省去，只針對我們可辨認的音頻作壓縮，因此可以減少壓縮的量，使壓縮後的檔案變得很小。

2.2 MP3 資料庫網頁的相關研究

時下的網際網路為何如此盛行呢？相信大家都會一口同聲的說它"酷、流行"，網路吸引人們的原因是藉由多媒體製造出酷炫的網頁，讓大家投入網頁的編寫行列，再加上網路無遠弗屆，自然成為眾所矚目的焦點。現今國內外的網站可說不計其數，光拿 MP3 網站來說，就以讓大家看的眼花撩亂，所以我們對於 MP3 網站的設計，必須符合使用者的需求，達到不等待、要什麼資料就有什麼資料、下載迅速便利....等功能，才能吸引更多使用者的青睞。

綜合現今的 MP3 網站功能，不外乎要達到一些基本功能，如：歌手資料、歌詞、下載少數 MP3 歌曲等等，有些 MP3 網站要求大家在上站時註冊，一方面想知道大家的需求，另一方面可以寄一些廣告信或電子報，讓使用者隨時注意時下流行的音樂或新消息。在電子商務時代的來臨，相信網路訂單、下載將是未來的趨勢。

對於此次的研究計畫，將利用最新的 ASP 功能，配合 Flash 炫麗的動畫效果，將 MP3 網站架設起來，除了讓網站能與使用著互動外，還希望做到功能最齊全，以滿足使用者對 MP3 網站的真實需求。

三、MP3 音樂之內涵式查詢的含意

所謂的內涵式查詢便是針對你所要尋找的 MP3 對其內涵作分析、擷取媒體特徵值以建立索引結構，以及查詢比對搜尋的技術。

我們要如何找出我們想要的 MP3 資料呢？或許可以利用傳統的方式，將 MP3 資料用文字的方式做索引，提供使用者用關鍵字的方式去做查詢。但是這樣的方式有者一個很大的缺點，就是：有些多媒體資料很難用文字去表達。譬如我今天想要找一首歌，但我忘記他的歌名，我只記得他大概是怎麼唱的，此時關鍵字查詢便毫無用武之地。此時我們便需要所謂的內涵式查詢技術，對其 MP3 內容作分析，找出有用的資訊，幫助使用者解決類似上述的問題，達到用聲音去找音訊資料的目的。而這樣的技術我們稱為 MP3 音樂資料的內涵式查詢。

在這裡我們使用的查詢單位為一段 MP3 樂句，這是因為如果用一首歌去做查詢的話，其所需的比對時間一定會比較長；另外由於人們大多是只哼個一兩句做查詢，並不會唱完整首歌，所以也比較不符合人類的直覺，所以我們使用一段 MP3 樂句當作查詢單位。

在前文中我們說明了 MP3 編碼及解碼的原理。接下來將說明根據 MP3 編碼的原理來定義 MP3 的特徵值。我們將定義 MP3 特徵描述子(MFD, MP3 Feature Descriptor)並且詳細的描述 MFD 的意義。

3.1 MP3 索引單位的定義

如前面所述，基於 MP3 索引與查詢的單位必須符合人類的直覺，我們以一個樂句而非整首 MP3 歌曲做為索引與查詢的基本單位。如圖 4 所示，對每一首 MP3 歌曲(MP3 Object)，我們透過切割(Segmentation)的方式將其切成一組的 MP3 樂句(MP3 Phase)。對每一個 MP3 樂句又進一步分為樂音、框架以及取樣樣本等三層結構。1 個框架分為 1152 個取樣樣本。其中 1 個框架等於 1152 個取樣樣本是根據 MPEG 標準所規範。

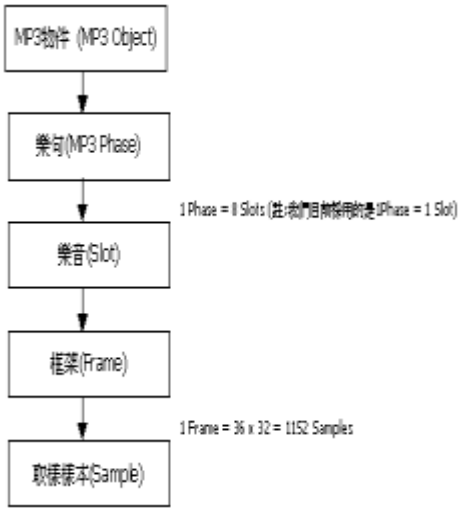


圖 4　MP3 物件的組成結構

3.2 MP3 音段的特徵值

多相位濾波器係數向量(PCV：Polyphase filter Coefficient Vector)根據前面所介紹的 MP3 編碼\解碼理論得知一個 wav 波形會經過 32 個濾波器組將其分為 32 個頻帶所組成的信號，而每個頻帶的值代表其信號在此頻帶的強度大小。所以我們取便是這 32 個頻帶的輸出信號強度，在一個音段內的平均值，合成一個 32 維的特徵值向量，稱之為多相位濾波器係數向量(PCV, Polyphase filter Coefficient Vector)，代表在一個音段內，聲音信號在 32 個頻帶強度分佈情形。其直觀意義為在某一時間點上，歌手及伴奏發出聲音的音高及音量在 32 個頻帶的能量分佈。

3.3 描述子(Discriminator)

在一 MP3 樂句裡大概有 8 個音符變化，這也是我們將一個 MP3 樂句分成 8 個樂音的原因。(註:在這部份我們只作一整個樂句。)在這裡描述子的意思是我們對每個樂音算出他們的 PCV，而樂音的 PCV 我們稱為樂音的描述子(Discriminator)。由於 PCV 為 32 維，所以描述子也是一個 32 維的向量。

每個樂音的 PCV 算法和 MP3 樂句的 PCV 算法相同：將每一個 slot 所包含的 PCV 平方加總便得到一個 slot 的 PCV，在對此 PCV 做正規化，最後此 PCV 便代表了這個樂音的特徵值。

	Subband1	Subband2	...	Subband32
PCV1	1.104086	0.044518	...	0
PCV2	1.03792	0.022601	...	0
PCV3	1.082287	0.051332	...	0
PCV4	1.17893	0.03522	...	0
PCV5	1.024902	0.037145	...	0
PCV6	1.078007	0.05127	...	0
PCV7	1.070043	0.021432	...	0
PCV8	0.279093	0.655118	...	0

圖 5 "City is so empty"樂句的八個樂音的特徵值向量

圖 5 顯示"city is so empty"樂句的八個樂音的特徵值向量，分別以 PCV1，PCV2，...，PCV8 表示之。

3.4 MFD: MP3 Feature Discriminator

綜合前面所述，我們定義了 MP3 的特徵值：MP3 特徵描述子(MFD)：

<MFD>=<PCV><Discriminator1><Discriminator2>...<Discriminator8>。MFD 為兩種向量的組成，所以一個 MP3 樂句的特徵值 MFD 的意義是此樂句的 32 個頻帶頻率強度分佈向量(PCV)和其 8 個描述子順序所組合而成的。由於 PCV 和描述子的維度都是 32 維，所以 MFD 的最多為一個 32*9 = 288 維的向量。

<Discriminator1><Discriminator2>...<Discriminator8>並不是代表第一段描述子、第二段描述子...第八段描述子的意思，而是根據彼此之間的歐基理得距離總和大小決定。如果其中一個描述子的其他七個描述子的歐基理得距離總和最大，此描述子便是<Discriminator1>，和其他七個描述子距離總和第二大的描述子為<Discriminator2>，以此類推...。

至於為何要根據描述子彼此的歐基理得距離來決定描述子的順序的原因是我們想要找出在這八個描述子中找出最特別的描述子，也就是最有可能是一段 MP3 樂句裡轉音的地方。設想一個描述子的和其他七個描述子的歐基理得距離總和為最大是不是有可能代表這個描述子有可能是這段 MP3 樂句裡發音最不一樣的部分(可能是轉音、突然聲音變大變高或是聲音變低變小)。所以描述子的用意在於找到 MP3 樂句裡 8 個小段並且根據其特別度加以排列組合，以配合 MP3 樂句的 PCV 做更精確的比對。

	PCV1	PCV2	PCV3	PCV4	PCV5	PCV6	PCV7	PCV8	Sum	Order
PCV1	0	0.07184	0.12591	0.19512	0.13942	0.18151	0.22539	0.43909	1.37827	2
PCV2	0.07184	0	0.00833	0.03046	0.01361	0.02572	0.04305	0.62499	0.81799	8
PCV3	0.12591	0.00833	0	0.00804	0.00292	0.00525	0.01544	0.68162	0.84751	7
PCV4	0.19512	0.03046	0.00804	0	0.00796	0.00056	0.00131	0.80689	1.05034	4
PCV5	0.13942	0.01361	0.00292	0.00796	0	0.00562	0.01504	0.70505	0.88962	6
PCV6	0.18151	0.02572	0.00525	0.00056	0.00562	0	0.00332	0.76554	0.98753	5
PCV7	0.22539	0.04305	0.01544	0.00131	0.01504	0.00332	0	0.86424	1.16779	3
PCV8	0.43909	0.62499	0.68162	0.80689	0.70505	0.76554	0.86424	0	4.88741	1

圖 6 計算描述子順序的例子

$$Similarity = dist(PCV_{D_i}^{P1}, PCV_{D_i}^{P2}) + \sum_{i=0}^{7} w_i \cdot dist(PCV_{D_i}^{P1}, PCV_{D_i}^{P2})$$ 其中 $dist(PCV_{D_i}^{P1}, PCV_{D_i}^{P2})$

為 $P1$ 的 PCV 和 $P2$ 的 PCV 的歐基理得距離

$dist(PCV_{Di}^{P1}, PCV_{D_i}^{P2})$ 為 $P1$ 第 Di 個 $Slot$

和 $P2$ 第 Di 個 $Slot$ 的歐基理得距離

Wi 為每個 $Slot$ 所代表的權重

圖 6 中 PCV1~PCV8 代表描述子 1~8，其相對應的數值為描述子之間彼此的歐基理得距離；Sum 是代表某一個描述子對其他描述子的歐基理的距離總和；Num 則表示描述子優先選擇順序。以第八個描述子(D8)來說，它和其它七個描述子的歐基理得距離總和為 4.88741，為裡面最大值，所以它便是優先被選取的描述子，但在這部份我們專題只做一整個樂句部份。

3.5 MP3 內涵式查詢:

根據前面所述我們定義了所謂的 MFD 為我們的特徵值，其物理意義為代表了一段 MP3 頻率強度的分佈情形及其 8 小段頻率強度的組合。這節便要說明如何用這樣的特徵值來做搜尋比對的動作。整個比對的流程如圖 7 所示：

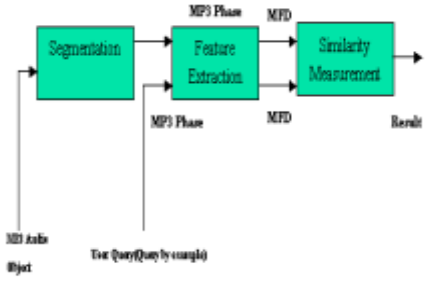


圖 7 MP3 內涵式比對流程圖

據使用者查詢的MFD和所有MP3句子的MFD，我們便可以做相似性的計算。在此我們使用了PCV歐基理德距離(Euclidean Distance)和樂音彼此的歐基理德距離及權重乘績來作為相似性的判斷依據，若距離小於一個基準值，便認定其為相似，若高於基準值則訂為不相似。

其中歐基理德距離的物理意義為:

假設有一個向量空間，有兩個向量空間上的點 α ， β 分別為 (a1,b1,c1,……) 和 (a2,b2,c2, ……)(其中維度可以無限延伸) ，其兩點間歐基理德距離為

$$dist(\alpha,\beta) = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2 + (c_1 - c_2)^2 + \ldots}$$

所以歐基理得距離代表了在向量空間上兩個點之間的距離。距離越小，表示兩點越接近，也代表他們越相似;距離越大，表示兩點越不接近，也代表他們越不相似。

四、實作平台與技術

本文的實作平台是採用 Microsoft Windows NT Server 4.0作為作業系統且將資料予以更新至 Service Pack 6，另外在資料庫部份是採用Microsoft SQL Server 7.0作為建構本專題資料庫系統的軟體，系統架構如圖 8 所示。MP3 特徵值查詢，採用 Microsoft Visual C++ 6.0作為整個程式的開發環境，最後是連結到 SQL Server 內去做查詢比對，程式流程如圖 9 所示。

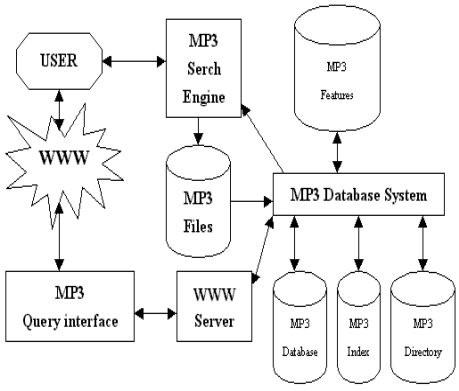


圖 8. 系統架構圖

MP3 特徵值查詢部份可以分為音樂查詢樣本錄製及播放、查詢樣本壓縮、MP3特徵值擷取與 MP3 播放、音樂查詢處理四個主要部份。

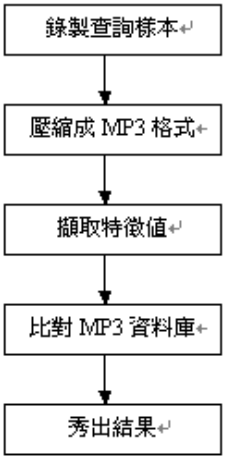


圖 9. 特徵值比對程式流程圖

4.1 音樂查詢樣本錄製及播放：

錄音時主要是利用到 Microsoft Visual C++ 6.0 所提供的 CPlayMMSound 、CrecordSound 和 CwriteSoundFile 這三個類別;其中 CPlayMMSound 是負責用來播放 WAV 資源的類別，也可以指定播放特定的 WAV 檔案，而 CrecordSound 是用來錄音的類別，若要將錄音的內容存成 WAV 檔案就需要用 CwriteSoundFile 這個類別。在實作上，一些主要的類別及函數的宣告，例如:CrecordSound、CplaySound 的宣告，請參考附錄 A。主要的流程如下:查詢樣本透過錄音程式，利用 CrecordSound 所提供的函式將聲音錄起來，然後透過 CwriteSoundFile 建立一個 WAV 檔案，最後可以利用 CPlayMMSound 的函式，撥放我們所錄製而成的 WAV 檔案。

4.2 查詢樣本壓縮：

這部份是將音樂查詢樣本錄製功能所錄製的 WAV 檔案予以壓縮成 MP3 檔案格式，我們所採用的技術是在 WinLame 3.70 Beta 版這開放原始碼中去擷取我們所需用到的類別予以修改整合到我們程式中。最主要的編碼函式如下:(其他相關函數及其說明請參考附錄 B)

```
int lame_encode_buffer(lame_global_flags
*,short int leftpcm[], short int rightpcm[],int
num_samples, char *mp3buffer,int
mp3buffer_size);
```

其中 leftpcm[]: 左聲道的 16bit 的 pcm data 矩陣；rightpcm[]: 右聲道的 16bit 的 pcm data 矩陣；num_samples: 左聲道和右聲道的取樣個數；mp3buffer: MP3 輸出 buffer 的指標；mp3buffer_size: MP3 buffer 的大小，回傳值如果為-1, 代表 MP3 buffer 太小。

4.3 MP3 特徵值擷取與 MP3 播放：

這部份我們所採用的技術 amp11 這開放原始碼，去擷取我們所需用到的類別，予以修改整合到我們程式中。但是 amp11 只提供播放 MP3 部份，而特徵值擷取部份則是抓 amp11 播放最後輸出的 hybridout 的值來做正規化，再求出它的特徵值，特徵值擷取的畫面如圖 10 所示。(使用 amp11 的說明請參考附錄 C)

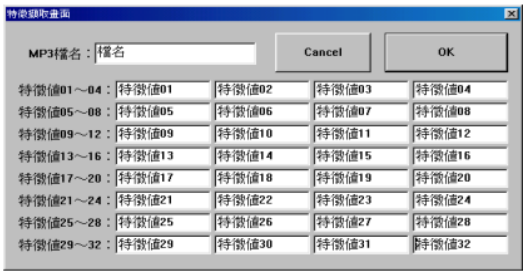


圖 10. 特徵擷取畫面

4.4 MP3 特徵資料庫與內涵式搜尋引擎

完成特徵值擷取的動作後，就是要將所得到的特徵值輸入至資料庫中;這次使用的特徵資料庫是採用 Microsoft SQL Server 7.0，做為資料庫的 SEVER 端;在資料欄位上，首先定義歌名、32 個特徵值與編號等欄位;利用 SQL 的語法，可以與資料庫作溝通，讓我們可以作一些資料庫的基本功能，例如:新增與刪除;在 CLINET 端則是合併前面所提的各個相關部分整合成 MP3 內涵式搜尋引擎(如圖 11 所示)，進一步與資料庫作連結。

圖 11 MP3 內涵式搜尋引擎

4.5 音樂查詢處理：

查詢處理可大約分為三大步驟，連結資料庫、計算歐基里德距離、排序，其中歐基里德距離的計算方法在之前有提到。

首先建立一個 mp3namefeature 類別宣告

public: CString mp3name ; double feature;

其中 CString mp3name 為 MP3 的檔名。

double feature 為 MP3 比較後的特徵值。

MP3 比較函式，計算歐基里德距離：

Double distancearray(double *aarray,double *barray)其中 double *aarra 和 double *barray 為要比較的兩個特徵值矩陣。採用 Quick Sort，傳入值為要排列的矩陣和左邊界和右邊界:

void QuickSort(mp3namefeature *list,int left,int right)

其中 mp3namefeature *list 為想要排列的矩陣。int left 為左邊界，通常為 0。

int right 為右邊界，內容為矩陣的大小。

4.6 測試與結果：

在測試時找來了幾位同學，實際以哼唱的方式，作 MP3 特徵值比對，測試的過程中，按照錄音、MP3 壓縮、MP3 特徵值擷取與資料庫比對等步驟，來完成測試;輸出的結果依照比對所計算的成績，列出最為接近的前 10 名，如下圖 12 所示。

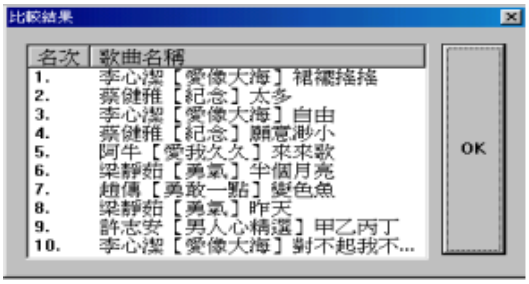


圖 12.一位女同學以哼唱裙襬搖搖做測試的結果

五、MP3 網頁設計與實作

5.1 Highlight MP3 Music Station 網頁設計之內容

『Highlight』一詞即是**豐富**、**精采**的意思，網站的名稱之好壞會直接影響各網友們前來參觀之意願，所以使用『Highlight』即是想表現出響亮、大方、簡單的目的，故取名為

Highlight MP3 Music Station。Highlight MP3 Music Station 其主要的進站畫面是希望利用流行音樂的強烈色彩來說明 MP3 音樂帶動網路潮流之意義，而其中各種樂器的繪製便是要強調 Highlight MP3 Music Station 之內容是以流行音樂為主，並進而吸引愛好 MP3 音樂的網友們前來參觀指教，如圖 13。

圖 13. Highlight MP3 Music Station 網頁

使用工具：此一進站畫面是利用 Flash 4.0 的軟體工具來達到製作動畫的效果，文字的縮放漸變、滑鼠經過產生泡沫的應用、文字顏色的變換與閃爍、樂器繪製的動畫、文字遮罩的使用以及可愛按鈕的製作。對於之前找過這麼多的網站，多多少少都了解和清楚這次網頁要如何設計，要有哪些項目和功能，而首頁頁面的設計主要是參考之前在找網站時，有些網站將它所有的功能都一併擺在首頁上，讓使用者能知道如何去使用這個網站的各項功能，所以設計首頁的版面，也是希望能讓使用者一看就能明瞭，每一個功能的作用，然後更能一看首頁就知道要如何操作和使用這個網站的全部功能，不需要使用者點來點去，花費時間去找尋他所需要的一些東西，我想這是首頁設計的主要重點。故將首頁歸劃成四個部份(分成上中左右共四部份)：

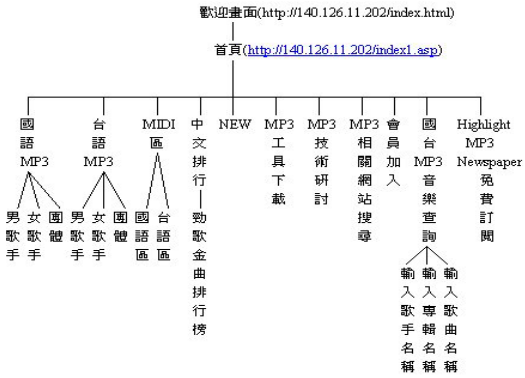
第一個部份是針對上面的部份：有 NEW、MP3 工具下載、MP3 技術研討、MP3 相關網站搜尋、會員加入五個功能，這些功能主要是針對之前所找的網站功能所做的統計，大部份的網站都有這些功能，故也要具備有相同的功能。在 N 方面，將做最新專輯最完整的介紹；在 MP3 工具下載方面，將有下載 MP3 所需的工具和聽 MP3 所需的軟體等等；在於 MP3 技術研討方面，是希望能做個大家都能來討論製作 MP3 一些技術的問題，或者是反應網站負責人的一些問題；在 MP3 相關網站搜尋方面，除了列出數個不錯的網站連結之外，還有一些像是蕃薯藤、奇摩之類的網站搜尋器；而在會員加入方面，只是希望能獲知大部份是哪些人在本站做些下載，或者是本站有哪些最新的活動都會優先讓會員參與。

第二部份是針對右邊的部份：有國語區(分男歌手、女歌手、團體)、台語區(分男歌手、女歌手、團體)、MIDI 區(分國語區和台語區)、中文排行榜(勁歌金曲排行榜)，對於 MP3 的區分，本有多種排法和意見，到最後還是選擇用國語區和台語區來分類，其中又分成男歌手、女歌手、團體；而在 MIDI 區方面，因為收集到的 MIDI 很多，又很雜亂，有些 MIDI 已經是很舊的了，甚至可能不知道歌手的名字，最後還是以國語區和台語區來分類；在中文排行榜方面，是參考別的網站的排行榜所作的整理，希望能讓使用者得到最新的流行歌曲的動向。

第三部份是針對中間的部份：中間的部份是大家一進站都會首先先看的部份，所以我們將本站最新消息和最新大碟推薦擺在中間，讓使用者一進站就能明瞭，本網站在近期的動態，還有哪些新進的歌曲，如果使用者只是要下載最新的 MP3 歌曲，就只要點選進入就能直接找到所需要的 MP3 歌曲。

第四部份是針對左邊的部份：查詢國台語 MP3 歌曲，而查詢又可依專輯名稱查詢、歌手名稱查詢、歌曲名稱查詢，只要在所指定的框框裡依名稱做輸入，按查詢鍵，就能讓使用者能在最快速的時間找到你所需要的 MP3 歌曲，這也是精心為使用者們做最貼心的設計；還有 Highlight MP3 音樂報，只要輸入 e-mail 就能不定期的收到本站的最新動態和消息。

5.2 Highlight MP3 Music Station 首頁設計之樹狀圖：



六、評估與展望

6.1 預期與實際成效之差距

經過實作後我們可以發現雖然輸出結果會因為一些人為因素，如樂句長度不一、查詢

音高不準、查詢 key 不準、查詢節奏不準影響而不如預期每次查詢都可以都在第一首歌就是自己想要查詢的歌，但都還在可接受的範圍內；在理想上，依照使用者的查詢，然後輸出所需要的結果給使用者，這樣就是一個最佳的查詢工具，也是我們所預期所希望的結果，但是世界上並無十全十美的東西，在實際上利用 MP3 特徵值去做查詢，只能做到相當近似的結果，並不能保證能夠完全正確無誤，所以我們在輸出結果時將最接近的 10 首歌列出，讓使用者知道，以便縮小查詢的誤差。

6.2 未來可能之擴展方向

這個專題的完成相信能將 MP3 音樂推向另一個高峰，讓使用者只用哼唱不需記取歌名與原唱者，便能在 MP3 網際網路資料庫上輕鬆的找到自己所想要的歌曲；未來若能與網際網路搜尋器做更密切的結合，未來任何人甚至利用大哥大手機上網，在任何時間、地點只要隨口哼出自己喜好的音樂便能自動比對、搜尋由自己的喇叭中唱出自己喜歡的歌曲，讓科技更深入生活，讓音樂更貼近人心，創造無限的商機。

七、結論

在本篇論文中,根據MP3編碼與解碼,而設計出來的 MP3 內涵式搜尋引擎,可以說是在網路上收尋 MP3 音樂的一大創舉,不僅是更方便且容易使用,在未來的應用也可以運用於隨選音樂(Music On-Demand)上以及網路點歌等。另一方面,我們要收集大量的 MP3 音訊資料，朝全台灣甚至全世界最大的中文 MP3 資料庫的目標來繼續努力。

八、參考文獻

[1] 蔡伯俊, "MP3 音樂物件之內涵式查詢", 中華大學電機工程學系碩士論文,1999.

[2] ISO 11172-3 Layer Ⅰ ~ Ⅲ Decoder Diagram.

[3] MPEG/Audio Coding/Decodeing Software Chad Fogg 1991.

[4] 資料庫系統 郭斯(Henry F. Korth)著 1992.

[5] 多媒體通訊與區域網路技術 張詩言編著 1993.

[6] HTML 設計寶典 洪錦魁、蔡昌均編著 1997/5.

[7] Active Server Pages & WEB 資料庫 王國榮著 1997/10.

[8] 多媒體於 WWW 之應用策略 王向葵編著 1997.

[9] C++ Program Design J. P. Cohoon and J. W. Davidson 1999/2.

[10] Visual C++ Ivor Horton's 著 蔡明志譯 1999/05.

[11] 輕鬆享用 MP3 郭長祐、王正裕 著 1999/9.

[12] SQL Server 7.0 設計實務 施威銘研究室著 2000/1.

[13] Windows 98 程式設計聖典 黃昕暐等譯.

[14] 精通 SQL Server7.0 資料庫系統 方盈編著 1999/3.

[15] Arman, F., A. Hsu, and M. Y. Chiu, "Image Processing on Compressed Data for Large Video Databases," *In Proc. of ACM Intl. Conf. on Multimedia*, 1993.

附錄 A 查詢樣本錄製及播放函數的宣告

CRecordSound 可用以下程式碼宣告：

```
m_pRecordSound = new CRecordSound();
m_pRecordSound->CreateThread();
```

CPlaySound 可用以下程式碼宣告：

```
m_pPlaySound = new CPlaySound();
m_pPlaySound->CreateThread();
```

開始錄音的程式碼如下：

```
m_RecordThread->PostThreadMessage
(WM_RECORDSOUND_STARTRECORDING,
0,0L);
```

停止錄音的程式碼如下：

```
m_RecordThread->PostThreadMessage
(WM_RECORDSOUND_STOPRECORDING,0,
0);
```

開始放音的程式碼如下：

```
m_PlayThread->PostThreadMessage
(WM_PLAYSOUND_STARTPLAYING,0,0);
```

停止放音的程式碼如下：

```
m_PlayThread->PostThreadMessage
(WM_PLAYSOUND_STOPPLAYING,0,0);
```

利用 CWriteSoundFile 來寫入 wav 檔，使用方法如下：

```
m_WriteSoundThread->PostThreadMessage
(WM_WRITESOUNDFILE_FILENAME,0,(LP
ARAM)(PWRITESOUNDFILE)&structWri
teSoundFile);
m_WriteSoundThread->PostThreadMessage
(WM_WRITESOUNDFILE_WRITEBLOCK,0,
```

(LPARAM)(WAVEHDR)pWaveHdr);
m_WriteSoundThread->PostThreadMessage
(WM_WRITESOUNDFILE_CLOSEFILE,0,0);

要將所錄的聲音記錄在 wav 檔中，需要有一個名稱為 WRITESOUNDFILE 的 structure，而 WRITESOUNDFILE 的描述如下：

```
typedef struct writesoundfile_tag {
  char lpszFileName[MAX_PATH];
  WAVEFORMATEX waveFormatEx;
  TCHAR buffer[100];
} WRITESOUNDFILE, *PWRITESOUNDFILE;
```

使用時必需要給一個檔名，然後 WAVEFORMATEX structure 就會定義要寫入的 wav 資訊。CPlayMMSound 可以讀入一個 wav 檔案並且利用聲音裝置來播放它，只要利用一個指標傳給 CPlaySound thread 即可，使用訊息如下：

```
m_pPlayMMSound->PostThreadMessage(WM_PLAYMMSOUND_PLAYFILE,0,(LPARAM)"sound.wav");
m_pPlayMMSound->PostThreadMessage(WM_PLAYMMSOUND_PLAYSOUNDPTR,0,(LPARAM)(CPlaySound*)m_pPlaySound);
m_pPlayMMSound->PostThreadMessage(WM_PLAYMMSOUND_CLOSEFILE,0,0);
```

其中 WM_PLAYMMSOUND_PLAYFILE 代表要開啟一個 wav 檔來準備播放,準備好要執行的工作後，用 WM_PLAYMMSOUND_PLAYSOUNDPTR 來執行工作，如果想在播放中停止播放，就用 WM_PLAYMMSOUND_CLOSEFILE。

附錄 B 查詢樣本壓縮

首先先講解 lame_global_flags 結構，它儲存著 MP3 內部的所設定：

```
typedef struct  {
/*輸入檔案描述 */
unsigned long num_samples;
/*取樣的個數, 內定為 2^32-1*/
int num_channels; /*聲道的個數,內定為 2*/
int in_samplerate;
 /* 輸入的取樣速率, 內定為 44.1kHz */
int out_samplerate;/* 輸出的取樣速率 */
/* 一般的控制旗標 */
int gtkflag; /* 是否執行 frame analyzer?*/
int bWriteVbrTag;
 /*是否增加 Xing VBR tag? */
int quality; /*音質設定 0=best, 9=worst */
int silent; /* disable 一些狀態輸出 */
int mode;
/* 0,1,2,3 stereo,jstereo,dual channel,mono */
int mode_fixed;/* 使用自定模式*/
int force_ms;
/*如果要使用 force M/S mode,此值為 1*/
int brate; /* bitrate */
/* frame 旗標 */
int copyright;/* 是否有版權, 內定值為 0*/
int original;/* 是否為原創, 內定值為 1*/
int error_protection; /* 是否每個 frame 用 2 bytes 來做 CRC 偵錯, 內定值為 0 */
int padding_type;/* 0=no padding, 1=always pad, 2=adjust padding */
int extension;/* MP3 的額外沒意義的 bit*/
 /* quantization/noise shaping */
int disable_reservoir;/* 是否使用 bit 儲存*/
int experimentalX;
int experimentalY;
int experimentalZ;
  /* VBR 控制 */
int VBR;
int VBR_q;
int VBR_min_bitrate_kbps;
int VBR_max_bitrate_kbps;
  /* 濾波器 */
int lowpassfreq; /* freq in Hz. 0=lame choses. -1=no filter */
int highpassfreq; /* freq in Hz. 0=lame choses. -1=no filter */
int lowpasswidth;/* freq width of filter, in Hz (default=15%)*/
int highpasswidth;/* freq width of filter, in Hz (default=15%)*/
 /* 讀取輸入檔案 – 如果用 i/o calling 程式就不使用 */
```

```
sound_file_format input_format;
int swapbytes;
/* force byte swapping, 內定值為 0 */
char *inPath;/* 輸入檔檔名*/
char *outPath;/* 輸出檔檔名*/
/* 聲學分係 */
int ATHonly;/* only use ATH */
int noATH; /* disable ATH */
float cwlimit; /* predictability limit*/
int allow_diff_short; /* allow blocktypes to d
iffer between channels ? */
int no_short_blocks; /*disable short blocks*/
int emphasis; /* obsolete */
    /*MP3 內部設定 */
long int frameNum; /* frame 計數器*/
long totalframes;
/* frames: 0..totalframes-1 (estimate) */
int encoder_delay;
int framesize;
int version;/* 0=MPEG2  1=MPEG1*/
int padding;
/*padding for the current frame?*/
int mode_gr; /* granules per frame*/
int stereo; /* 聲道數 */
int VBR_min_bitrate; /*min bitrate index */
int VBR_max_bitrate; /*max bitrate index*/
float resample_ratio;
/* 輸入取樣數率 除 輸出取樣數率除*/
int bitrate_index;
int samplerate_index;
int mode_ext;
/* Low pass 和 High pass 濾波器設定 */
float lowpass1,lowpass2;/* normalized frequen
cy bounds of passband */
float highpass1,highpass2;/* normalized freque
ncy bounds of passband */
/* polyphase filter (filter_type=0)  */
int lowpass_band;/*zero bands >=
lowpassband in the polyphase filterbank */
int highpass_band;/* zero bands <= highpass
_band */
int filter_type;/* 0=polyphase filter, 1= FIR
filter 2=MDCT filter(bad)*/
int quantization;/* 0 = ISO formual, 1=best
amplitude */
int noise_shaping;    /* 0=none 1=ISO AAC
model 2=allow scalefac_select=1 */
int noise_shaping_stop;
int psymodel; /* 0 = none 1=gpsycho */
int use_best_huffman;  /* 0 = no. 1=outsidel
oop  2=inside loop(slow) */
} lame_global_flags;
```

使用方法

void lame_init(lame_global_flags *);初始化

void lame_init_infile(lame_global_flags *);
/*開啟輸入的檔案*/

void lame_init_params(lame_global_flags *);
/*設定更多的內部設定*/

int lame_readframe(lame_global_flags *,short
int Buffer[2][1152]); /*讀取一個由 lame_init_i
nfile 所開啟檔案的一個 frame*/

int lame_encode_buffer(lame_global_flags *,sh
ort int leftpcm[], short int rightpcm[],int num
_samples, char *mp3buffer,int mp3buffer_siz
e);其中 leftpcm[]: 左聲道的 16bit 的 pcm data
矩陣;rightpcm[]: 右聲道的 16bit 的 pcm data
矩陣;num_samples: 左聲道和右聲道的取樣個
數; mp3buffer: MP3 輸出 buffer 的指標
mp3buffer_size: MP3 buffer 的大小
回傳值如果為-1, 代表 MP3 buffer 太小

void lame_close_infile(lame_global_flags *);
如果 lame_init_infile()已經開啟了, 就把它關
閉。

附錄 C amp11 所使用部分的說明

decoder 是一個類別, 利用 ampegdecoder dec; 宣告：

利用下列程式碼來打開 decoder：

dec.open(instream, freq, stereo, fmt, dwn, chn);其中 binfile &instream: 宣告為 binfile 的 encoded 資料。

int &freq:/* decoder 資料的頻率*/。

int &stereo: /*當 decoder 資料為 stereo 時為 true。*/

int fmt:/* 代表 format, 0: float, 1: little endian 16 bit。*/

int dwn: /*代表 downsampling, 0: none, 1: by 2, 2: by 4。*/

int chn: /*代表 channels, -2: 2 but downmixing, 0: as in mpeg, 1: 1, 2: 2。*/

結束 decoder 類別時用下列程式碼：

dec.close();

要讀 decoder 內的一些資料, 用下列程式碼：

n=dec.read(buf, len);

其中 int &n: 要讀幾個 bytes。

void *buf: 存回傳所需資料。

int len: 要讀幾個 bytes。

要 seek 到 decoder 資料內的的某一點, 用下列程式碼：

p=dec.seek(pos);

其中 int &p: 新的那一點指標。

int pos: 要 seek 到那一點。

ps: 通常會有誤差。

利用下列程式碼設定 equalizer 為 32bands：

dec.ioctl(dec.ioctlsetequal32, equal32, 0);

利用下列程式碼設定 equalizer 為 576 bands：

dec.ioctl(dec.ioctlsetequal576, equal576, 0);

其中 float equal32[32]:/* per band (freq/64) amplification,1:normal。float equal576[576]:

per band (freq/1152) amplification, 1: normal。*/

利用下列程式碼設定 amplification：

dec.ioctl(dec.ioctlsetvol, &vol, 0);其中 float &vol: volume, 1:normal。利用下列程式碼修改 stereo 輸出：dec.ioctl(dec.ioctlsetstereo, vols, 0);其中 float vols[3][3]: volume matrix [dst] [src]: [0]:left, [1]:right, [2]:mono. 。