

國立清華大學

碩士論文

以統計方法與音樂理論為基礎之和弦辨識系統
(Chord Identification Based On Statistical
Methods and Musical Theory)

系別 資訊工程學系 組別

學號姓名 894375 鄭雯妮 (Wen-Ni Cheng)

指導教授 張智星博士 (Jyh-Shing Roger Jang)

摘要

製作電子音樂是一件不容易的事情，首先他們要利用鍵盤輸入主旋律，並分析旋律和絃，最後再將伴奏加入使其悅耳動聽，這一連串的動作對沒有樂理知識的人來說是非常困難的工作。

本論文中，系統將對使用者利用麥克風所哼入的旋律進行和絃辨識。利用基本的樂理與統計方法作為理論基礎來執行和絃辨識，並將此兩種方法的分析結果利用 Dynamic programming 演算法找出最佳的和絃解答。如此一來，使用者不需利用鍵盤輸入，也不需有樂理基礎就可以利用辨識出的和絃結果為旋律配上伴奏。

Abstract

Most people can sing or hum a song that they are familiar with. However, it is rather difficult, if not impossible, for common people without formal music skills or training to compose a song. To be able to do this, they need to use a keyboard to produce a melody, analyze its chords, and then add accompaniment via physical musical instruments or computer software. This series of actions require years of practice and experience.

In this thesis, we have constructed a system that can convert a user's humming into music score that contains a melody track and appropriate accompaniments. First of all, a user can hum to the microphone directly and the system will do pitch tracking and note segmentation to identify music notes and measures. In the second step, the system will analyze each measure's music note and find the chord candidates as well as associated probabilities based on music theory and statistics from a set of sample music. Finally, the system will use dynamic programming to find the best chord sequence based on chord state probabilities and chord transition probabilities. Subjective tests on the resultant music show that the automatically generated chords are satisfactory for most common users.

致謝

首先，感謝張智星老師兩年來的熱心指導，讓我獲益良多。同時，也感謝實驗室裡一起努力的同學，學長及學弟妹，讓我度過了兩年愉快的實驗室生活，謝謝你們兩年來的照顧與幫助。

另外，感謝我的家人，在我疲倦的時候，總是給我熱情的支持，作為我精神的避風港，使我能在研究上盡情衝刺而無後顧之憂。

最後，要感謝我的男友，總是能包容我在壓力大時的壞情緒，並在我失落無力時給我最大的鼓勵，作為我精神的支柱。

目錄

摘要	錯誤！尚未定義書籤。
Abstract	錯誤！尚未定義書籤。
致謝	錯誤！尚未定義書籤。
目錄	v
圖表目錄	vii
第一章 緒論	1
研究動機	1
系統簡介	1
系統流程	2
系統之應用	2
章節概要	3
第二章 資料的前處理	4
基本樂理簡介	4
何謂 MIDI 檔	6
MIDI 檔的格式簡介	6
MIDI 操作實例	8
MIDI 轉換中介格式	9
中介格式之實例	11
第三章 音高追蹤	12
前言	12
系統流程簡介	12
第四章 和絃辨識系統	16
前言	16
何謂和絃	16
系統簡介	17
機率計算	18
機率計算 - 統計演算法	19
機率計算 - 樂理演算法	23
Dynamic Programming 演算法	29

第五章 實驗結果	31
前言	31
和絃為主之辨識實驗	32
利用問卷調查之實驗結果.....	35
第六章 結論與錯誤分析及未來方向	37
結論	377
錯誤分析	377
未來方向.....	38
參考文獻.....	39

圖表目錄

圖 2-1 音名與音階對照圖	4
圖 2-2 音名與音程對照圖	5
圖 3-1 基頻擷取流程圖	12
圖 3-2 音高追蹤範例圖	15
圖 4-1 和絃辨識系統流程圖	17
圖 4-2 統計演算法之機率計算流程圖	19
圖 4-3 音名與半音對照表	20
圖 4-4 和絃之組成音	24
圖 4-5 樂理演算法之機率計算流程圖	25
圖 4-6 和絃行進之範例圖	28
圖 5-1 和絃辨識之實驗結果	33
圖 5-2 問卷製作示意圖	35

第一章 緒論

研究動機

隨著科技的進步，人類對於多媒體娛樂的需求越來越大，但對於電子音樂，一般大眾似乎無法馬上入門，因為要製作電子音樂如 MIDI 檔，人們一定要有一定的樂理及工具(如鍵盤)，因此，是否能幫助人們用簡單的方法來製作電子音樂，進而吸引人們對音樂的興趣，即為我們所要研究的方向。

系統簡介

要製作電子音樂，最普遍的方法是利用鍵盤來將旋律輸入電腦中，不但要輸入主旋律，為了要讓音樂豐富，通常還需要加入伴奏，如此對樂理不甚熟悉的人來說是一件難事。

而和絃辨識系統則僅需要使用者利用麥克風，將主旋律利用哼唱的方式輸入至電腦中，系統不但會分析出主旋律音符，還會根據主旋律來配上和絃，最後還能整合成 MIDI 檔將結果輸出。

如此一來，便可大大減輕使用者的負擔，使用者僅需輸入主旋律即可，既不用熟悉樂理，也不需經由繁雜的程序來輸入主旋律，使得製作電子音樂變得輕鬆。

系統流程

大體來說，系統可分成下列幾個步驟：

(一) 哼唱系統：

1. 使用者利用麥克風輸入八小節主旋律。
2. 辨識哼唱的旋律。
3. 微調其旋律，使其最佳化。

(二) 和絃辨識系統：

1. 將哼唱系統的輸出轉換成中介格式。
2. 找出未知旋律各和絃的機率值為何。
3. 執行 Dynamic Programming 以找出最佳的和絃路徑。

系統之應用

1. 應用在手機中，製作屬於自己的和絃鈴聲。
2. 音樂創作，可自製 MIDI 檔，並可經由伴奏的設定製作不同曲風的電子音樂。
3. 應用於玩具上。

章節概要

本論文的章節安排如下：

第一章 「緒論」：介紹研究動機，系統簡介，系統流程，以及系統的應用為何。

第二章 「資料的前處理」：介紹本論文所使用的資料格式，基本樂理介紹與 MIDI 格式介紹。

第三章 「音高追蹤」：介紹如何將麥克風的音訊輸入轉換成方便辨識的 MIDI 格式。

第四章 「和絃辨識系統」：介紹和絃辨識系統的演算法及流程圖。

第五章 「實驗結果」：介紹各種實驗所得的辨識率。

第六章 「結論與錯誤分析及未來方向」：將實驗所得的結果進行分析歸納，並檢討所有的錯誤原因。

第二章 資料的前處理

由於我們資料庫中的音樂檔案為 MIDI 檔，要直接對 MIDI 檔做辨識或處理並不容易，因此我們必須先將 MIDI 檔轉換成較易處理的中介格式，以增加程式的可讀性，此外，對檔案做更改也較容易。

基本樂理簡介

由於此研究橫跨資訊與音樂的領域，因此我們針對所需的樂理做一些簡單的介紹，以方便讀者理解。

1. 音階(Scale)：

意即音的梯子之意，把製造旋律時必須用的音，以高低順序排列而成的八度音列，而本研究是以大音階為主，就是在一個八度音中，白鍵所構成的音階部分。

2. 音名(Tone name)：

為了便於區別許多的樂音，人們將所有的音加上其名稱以方便區分。由於音階具有重複性，因此真正屬於不同音階的僅七種(Do、Re、Mi、 Fa、 Sol、 La、 Si)，且通常由 Do 音開始：

音名	C	D	E	F	G	A	B
音階	Do	Re	Me	Fa	Sol	La	Si

圖 2-1 音名與音階對照圖

3. 音程(Interval)：

兩個音的高度關係(例如 Do 到 Re)，而其計算的最小單位為半音(Semitone)，有些音之間的距離僅差半音(例如 E 到 F)，此時我們稱之為“半音程”，而有些則差兩個半音(例如 C 到 D)，我們則稱之為“全音程”。以下是音階中各個音之間的音程表：

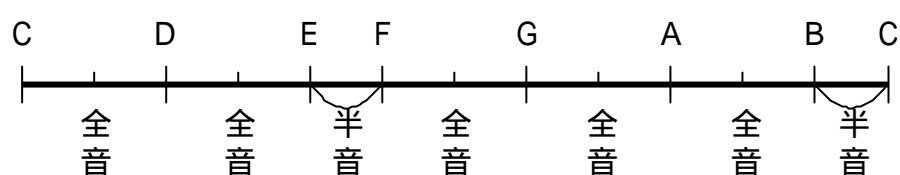


圖 1-2 音名與音程對照圖

4. 拍子(Time)與小節(Measure)：

在樂音的進行中，強弱以一定的、很有規則的型態反覆的，就叫做拍子。而在樂譜上，為了區分拍子的重複所畫的縱線，則稱做小節。若拍子的記號表示為 4/4，則代表此旋律是以 1/4 音符為一拍，而每小節有 4 拍。

5. 調號(Key Signature)：

意指音階具有一定的高度時，第一音(主音)的位置，固定在某一音名上。例如，C 大調就是以 C 音當作主音的大音階調子。

何謂 MIDI 檔

MIDI 全名為(Musical Instrument Digital Interface)，其原理是將各個樂器的音色加以編號，但這些音色並不儲存於 MIDI 的檔案中，相對的，為了減少檔案大小，MIDI 檔案裡面只記錄了音樂的代號及一些高低音的設定，而那樂器的聲音則記錄在音效卡上面。

MIDI 檔的格式簡介

要介紹 MIDI 的規格並不容易，因此我們僅對我們需要用到的部分做介紹。

1. 音軌(Track)：

MIDI 檔是由多條音軌(Track)所構成，而音軌可允許 MIDI 檔同時發出多種聲音，而 MIDI 檔內建有 16 條音軌供使用者使用，其範圍為 0 ~ 15。

2. 事件(Event)：

MIDI 檔的運作，簡言之就是利用發出事件(Event)來執行命令，其中最主要的就是兩種 Event Note on 事件及 Note off 事件。

(a)Note on 此事件是讓 MIDI 發出聲音。若我們要讓 MIDI 檔發出聲音，需要設定的是音軌、發出的音高及音量大小。當我們要發出一個聲音如中音 C 時，系統會發出 3 個位元組給 MIDI Out 。在十六進位中，其格式是：

9n kk vv

其中 n 代表所在的音軌，kk 代表所要發出的音高(0 ~ 127)，其單位為半音，而 vv 是其音量大小(0 ~ 127)。舉例來說，若我們要在第 1 條音軌上發出中音 C 音量為 127 的指令，我們就要對 MIDI 發出以下指令：

90 3C 7F

(b) Note off 此事件是將聲音關閉。其原理與 Note on 相似。格式為

8n kk vv

若我們要將在第 1 音軌其音量為 127 的中音 C 關閉的話，其設定如下：

80 3C 7F

但我們也可以利用 Note on 其音量設為 0 來取代 Note off：

90 3C 00

3. 時序(Time)：

MIDI 檔其時間間隔的方法，是以 Tick 為單位，利用其與 1/4 音符(Quarter Note)的關係來推算出該事件所佔據的時間為何，其公式如下：

$$\text{SamplingRatePerTick} = \frac{\text{SamplingRate}}{\text{BeatsPerSecond} \times \text{TimeFormat}}$$

(a) SamplingRate 取樣頻率。

(b) BeatPerSecond 亦可稱為 QuarterPerSecond，計算出一個 1/4 音符為幾秒。

(c) TimeFormat 1 個 1/4 音符有幾個 Ticks。

MIDI 操作實例

要發出連續的聲響，僅需在事件與事件中間加上其間隔時間即可。

例如小蜜蜂的前兩小節，其簡譜為：

| 5 3 3 | 4 2 2 |

在此不贅述 MIDI 檔的檔頭及一些固定的規格，僅說明其重要部分。

假設我們的設定是第一音軌，且其音量為 100，則其設定為：

<註> 以下的位元皆用十六進位

90 43 64 78	(中音 S0 , Note On , 音量 100 , 時間 120 Ticks)
90 43 00 00	(中音 S0 , Note Off , 音量 0 , 時間 0 Ticks)
90 40 64 78	(中音 ME , Note On , 音量 100 , 時間 120 Ticks)
90 40 00 00	(中音 ME , Note Off , 音量 0 , 時間 0 Ticks)
90 40 64 81 70	(中音 ME , Note On , 音量 100 , 時間 240 Ticks)
90 40 00 00	(中音 ME , Note Off , 音量 0 , 時間 0 Ticks)
90 41 64 78	(中音 FA , Note On , 音量 100 , 時間 120 Ticks)
90 41 00 00	(中音 FA , Note Off , 音量 0 , 時間 0 Ticks)
90 3E 64 78	(中音 RE , Note On , 音量 100 , 時間 120 Ticks)
90 3E 00 00	(中音 RE , Note Off , 音量 0 , 時間 0 Ticks)
90 3E 64 81 70	(中音 RE , Note On , 音量 100 , 時間 240 Ticks)
90 3E 00 00	(中音 RE , Note Off , 音量 0 , 時間 0 Ticks)

MIDI 轉換中介格式

由於 MIDI 檔的格式太過繁雜，因此我們僅擷取其重要的部分，以方便做資料的分析即可。此外，資料庫中的每筆資料皆為 8 小節的 MIDI 檔，其內容為中英文的流行歌曲，因此我們將每一筆 MIDI 檔處理成中介格式以利研究。

以資料庫中每一個 MIDI 檔案為單位，每筆資料都轉換成一筆 Struct 檔案，且將所需的資料儲存其中。以下就是每個 Struct 裡頭的參數：

1. Track : 此為紀錄 MIDI 檔案中音高音長的陣列。經由 MIDI 檔我們可以擷取出所有的半音(Semitone)及其音長(Duration), 因此我們可以利用一個一維陣列來儲存之。其格式為:

[Semitone1 Duration1 Semitone2 Duration2 Semitone3 Duration3]

其中 Semitone 的範圍為 0~127, Duration 的單位是毫秒。例如上述小蜜蜂的例子。我們可以設定其 Semitone 陣列為:

[67 600 64 600 64 1200 65 600 62 600 62 1200]

2. Quarter : 用來記錄四分之一小節的時間。 由於 MIDI 檔中時間的單位是以 Tick 為單位, 而一個 Tick 所佔的時間又相關於 1/4 音符的時間, 因此 1/4 音符的時間是非常重要的, 而 1/4 音符的時間是以毫秒為單位。
3. Name : 紀錄所擷取的 MIDI 檔檔名。
4. Key : 此 MIDI 檔的調號。

5. Measure：將每一個 MIDI 檔案再切分以小節為單位所成的 Struct 檔案。由於每一首 MIDI 檔案都有 8 小節，因此 Measure 裡頭將有八個 Struct 檔案。每個 Struct 裡的參數如下：

(a) track：該小節的音高音長陣列，其格式同(1)。

(b) chord：此小節的和絃為何。

中介格式之實例

midi(1)的內容為：

```
track: [1x72 double]
quarter: '600000'
name: 'main\rightHereWaiting_RichardMarx_1.mid'
key: 'C'
measure: [1x8 struct]
```

而 measure 的參數內容為：

midi(1).measure(1)

```
track: [0 1200 67 300 72 300 74 300 74 300]
chord: 1
```

第三章 音高追蹤

前言

如何將使用者所哼唱的音正確地辨識出來，其實並不是件容易的事，因為所出現的誤差有可能出自於系統本身，也有可能是因為哼唱者哼入錯誤的音高所造成。要如何在此兩項誤差中找到最佳的辨識結果，就是我們需要研究的地方。

首先我們先利用語音訊號處理的觀念將聲音的訊號擷取出來，然後執行音高追蹤(Pitch Tracking)[1]以求得我們所需的資訊，之後在經由微調找出最佳的辨識結果：

系統流程簡介

一．基頻擷取(Pitch Tracking)

其流程圖如下：

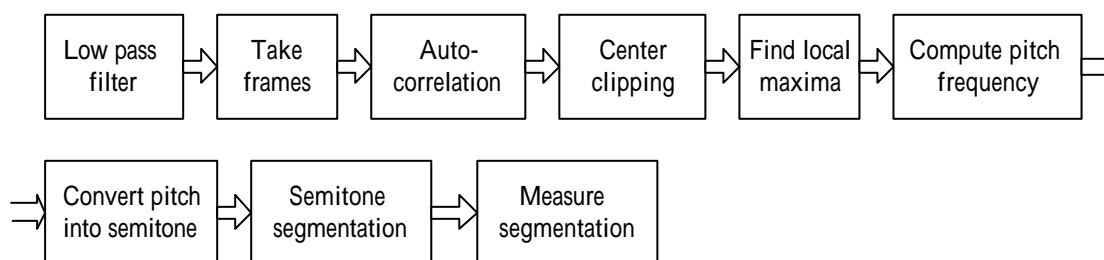


圖 2-1 基頻擷取流程圖

Pitch Tracking 的技術有很多種，例如 Wavelet，Autocorrelation 和 AMDF(Average Magnitude Difference Function)等。而我們的辨識系統是利用 Autocorrelation 的方法。以下簡述各步驟的一些概念：

1. Low pass filter：

聲音訊號進來後，為了泯除一些高頻的雜訊或暴音，故以 Low pass filter 來處理，期望一些較高頻率被濾除，只留下較低的頻率訊號。

2. Frame blocking：

將訊號作音框化處理，大約以 512 為一個音框的大小，而音框重疊部分的長度大小為 314。

3. Auto-correlation：

對每一個音框作 Auto-correlation，如此可以找出相似波形重複出現的週期，其公式為：

$$r(t) = \frac{1}{N} \sum_{n=1}^N S(n)S(n+t)$$

4. Center clipping：

由於一些比較弱的訊號會造成我們在算週期時的錯誤，因此在算 Local maximum 之前，我們需用 Center clipping 來過濾這些訊號。

5. Local maximum or Local minimum：

經過 Auto-correlation 及 Center clipping 後，我們便可以利用相鄰峰點在時間軸上的距離找到聲音的週期，最後取其倒數其為其基頻。

二. 基頻量化處理

由於資料庫中的檔案為 MIDI 檔而非一般的語音訊號，因此我們需要將頻率轉換成 MIDI 格式的聲音訊號(Semitone)。在語音訊號中，A440(Middle La)[2] 是常被用來做頻率轉換的標準，而由於半音間相差 $\sqrt[12]{2}$ ，全音間相差 $(\sqrt[12]{2})^2$ ，因此我們可以得到下面的轉換公式：

$$freq = 440 * 2^{\frac{offset}{12}} \quad (offset = \text{Semitone} - 69)$$

而由於我們音樂資料庫裡，音樂的格式為 MIDI 格式，因此我們在將公式做一些轉換後可得：

$$semitone = 12 * \log_2\left(\frac{freq}{440}\right) + 69$$

三. 音符擷取

我們將基頻量化的結果(Semitone)作音符擷取的工作，將旋律切割成一段一段的音符。找出各段音符的起始點後，再利用中位數(Median)將 Semitone 平滑化。

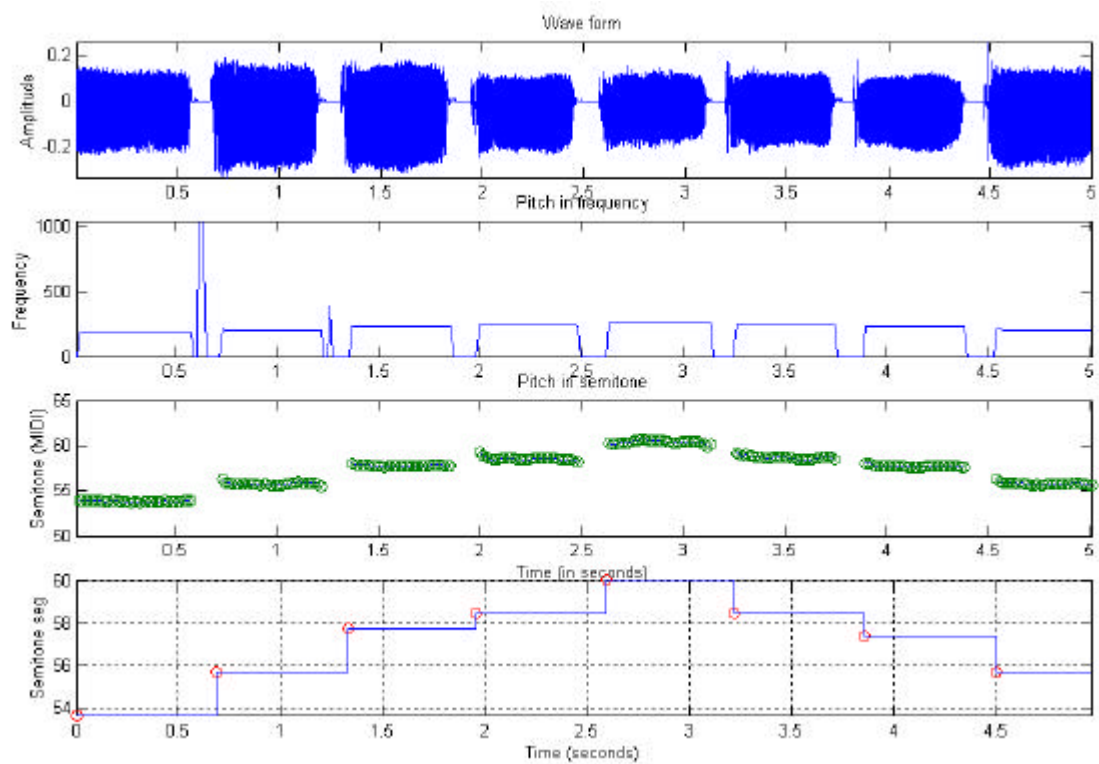


圖 3-2 音高追蹤範例圖

圖 3-2 由上而下依序為：聲音訊號圖，基頻頻率圖，基頻量化圖，音符擷取圖。

第四章 和絃辨識系統

前言

要能產生優美的旋律，在樂理上，和絃扮演著很重要的角色，換言之，若我們能掌握旋律的和絃，大體的伴奏即可呼之欲出。然而，和絃的辨識卻是初學者最困難的一環。學生為了要取得較客觀的和絃樂理，曾請教過一些音樂老師，結果發現，幾乎是經驗凌駕著理論，和絃辨識並沒有一個強而有力的法則，因此，要如何利用電腦來進行和絃辨識，就變成了一個頗具有挑戰性的研究。

何謂和絃

由數個(至少兩個)高度不同的音，同時鳴響而生共鳴，則稱之為和絃或合聲，簡言之，就是能配合主旋律而發出最自然和絃的聲響。因此，若我們能找到某小節的和絃，就可以利用此和絃來幫此小節找到最佳的配樂合聲。所以在樂理上，若我們要幫一段未知的旋律配樂，首要條件除了要找到此旋律的調性之外，還要知道此旋律的和絃。

系統簡介

系統流程圖如下：

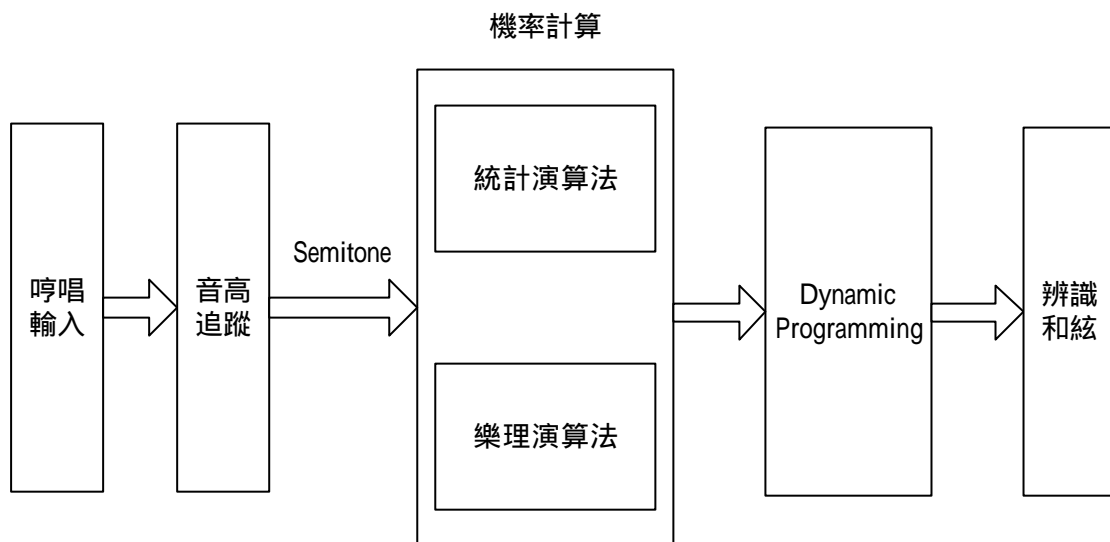


圖 4-1 和絃辨識系統流程圖

圖 4-1 中主要的辨識系統有兩種演算法 - 統計演算法和樂理演算法，經由這兩種演算法，我們可以將經驗及樂理作完整的結合，讓辨識結果更為理想完善。此兩種演算法將會分別找出測試資料的和絃機率，之後再經由 Dynamic Programming 演算法找出最佳的和絃答案組合。

以下我們將對機率計算的演算法做介紹。

機率計算

一. 前言：主要可分為兩部份：

1. 小節內之和絃分析(State Probability)：

根據和絃原理，和絃的功能主要是配合主旋律來發出最自然和音的聲響，由此可知，小節裡的音符與其和絃之間的關係是密不可分的。因此我們可以假設，若小節中出現了某些音符，那麼它可能是哪種和絃的機率較高。

2. 相鄰小節間的和絃分析(Transition Probability)：

學生發現，小節與小節之間也有其關聯性。例如，當某小節出現 A 和絃時，則下一小節出現 B 和絃的機率則會大為增加，因此，我們也可以統計一下相鄰小節間其和絃的關係。

由於統計與樂理方法是根據兩種截然不同的基礎來執行機率計算，因此我們將利用它們來分別找出各自的 State Probability 與 Transition Probability，然後再執行 Dynamic Programming 找出最佳解。以下我們將分別對這兩種演算法作詳細的說明介紹。

機率計算 - 統計演算法

系統流程圖如下：

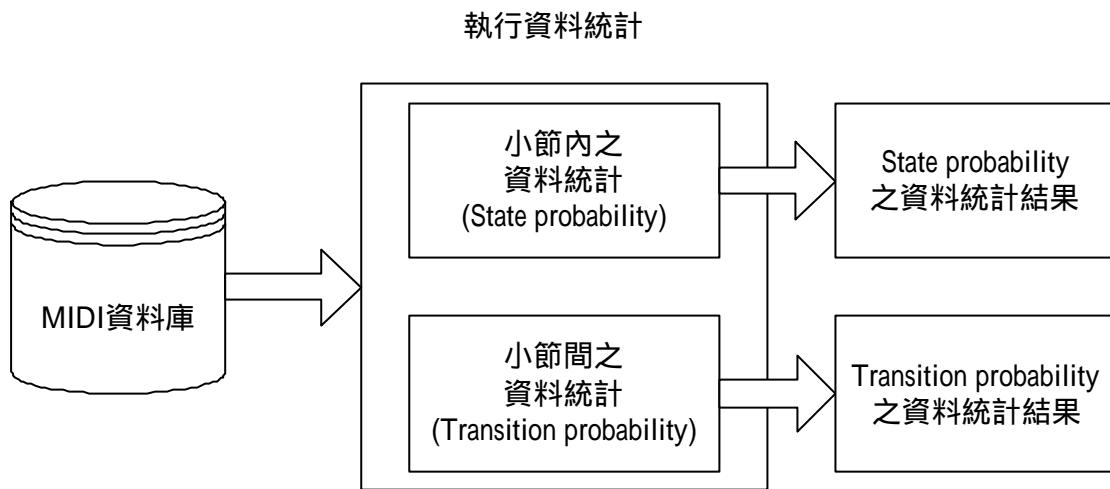


圖 4-2 統計演算法之機率計算流程圖

以下我們分別介紹小節內(State Probability)與小節間(Transition Probability)的特徵擷取方法。

一. $State(i, j)$ - 小節內之分析部分：

首先我們要統計資料庫中的檔案。第一步即統計小節中各種音符的組合。假設不同小節，若具有相同的音符組合，則加總其和絃並記錄起來。接著，將欲測試與統計完的資料作比對，若有相同的音符組合，我們就可利用其和絃機率(State Probability)來預測其可能的和絃為何。其格式如下：

1. Note : 小節中的音符組合。由於音符具有重複性，因此我們將以半音 (Semitone) 為單位，以八度音 (Do 到高音 Do) 為一循環，共有 12 半音來記錄其音符的組合。其對照表如下：

C	#C	D	#D	E	F	#F	G	#G	A	#A	B
1	2	3	4	5	6	7	8	9	10	11	12

圖 4-3 音名與半音對照表

2. Prob : 紀錄該種音符組合可能的和絃機率。

以下利用一個例子來說明：

Mx 代表不同的小節，Cx 代表其和絃，Nx 代表其音符組合，注意我們僅紀錄小節中出現的音符，其出現的順序、次數的多寡並不考慮。由此我們發現音符的組合都相同：

M1 = | 1 1 5 5 6 6 5 5 | , C1 = “C” , N1 = [1 5 6]
M2 = | 1 5 6 6 5 6 6 6 | , C2 = “A” , N2 = [1 5 6]
M3 = | 5 5 6 6 1 1 5 5 | , C3 = “D” , N3 = [1 5 6]
M4 = | 1 5 6 6 6 5 1 1 | , C4 = “C” , N4 = [1 5 6]

則我們可以將其整理成：

```
trainData(1).Note = [ 1 5 6 ]
trainData(1).Prob = [ C 50% D 25% A 25% ]
```

二. $\text{Transition}(i, j)$ - 相鄰小節間之分析部分：


在這裡我們也是利用資料庫的資料來作相鄰小節間的和絃統計，計算一下若前一小節的和絃為 X 和絃的話，那下一節和絃為 Y 和絃的機率將為何。由於我們的測試資料皆為 8 小節，且所有和絃共有 12 個，因此我們可以利用 7 個 12 * 12 的二維陣列來儲存小節間的和絃機率(Transition Probability)。其演算法如下：

```
% ===== 對 7 個 12*12 的二維陣列初始化 =====  
for i=1:7,  
    measure(i).transProb=zeros(12);  
end  
  
% ===== 進行統計工作 =====  
for i=1:length(midi),  
    for j=1:7  
        from=midi(i).measure(j).chord;  
        to =midi(i).measure(j+1).chord;  
        measure(j).transProb(from,to)=  
            measure(j).transProb(from,to)+1;  
    end  
end
```

```
% ===== 執行正規化 =====
for i=1:7,
    measure(i).transProb=measure(i).transProb;
    rowSum=sum(measure(i).transProb, 2);
    measure(i).transProb=diag(1./rowSum)*measure(i).transProb;
end
```

因此，若我們想知道從第二小節 C 和絃到第三小節 D 和絃的
Transition Probability，我們僅需查詢：

```
prob = measure(2).transProb(C,D)
```



2~3 小節 C 至 D 小節

機率計算 - 樂理演算法

一 .前言：

上一種的統計演算法是根據資料庫中的資料來找出和絃機率，而此演算法是利用基本的樂理觀念，並沒有用到任何資料庫中的檔案資料。如此，我們的系統既有經驗法則(Experience-base)，也有樂理的理論(Rule-base)來加強我們的預測結果，使得實驗結果更加客觀。

我們將利用一些基本的樂理理論作為基礎來找出每小節最有可能的和絃候選人。

二 .樂理簡介 - 和絃組成音：

和絃的原理，就是協和音的組合，本實驗主要的和絃是以三和絃為主(Triad)，也就是以某音做根基，往上各以三度重疊兩個音而形成(以 C 和絃為例，其組合音就為 C、E、G)，而這三個音，分別稱為根音(Root)、第三音(Third)和第五音(Fifth)。下圖 4-4 為 12 個和絃的組成音對照表(為了方便閱讀，避免混淆，以下組成音皆由數字代替)：

和絃	根音(Root)	第三音(Third)	第五音(Fifth)
C	1	3	5
#C	#1	4	#5
D	2	#4	6
#D	#2	5	#6
E	3	#5	7
F	4	6	1
#F	#4	#6	#1
G	5	7	2
#G	#5	1	#2
A	6	#1	3
#A	#6	2	4
B	7	#2	#4

圖 4-4 和絃之組成音

由於和絃共有 12 個，而每個和絃又有 3 個音，因此每個音都會有重複出現的情形(例如，若出現 1 的音，他有可能 C(1,3,5)和絃、F(4,6,1)和絃、#G(#5,1,#2)和絃)。所以我們利用這種方法，當某個音跟和絃組成音有很強的關聯性時，我們可以利用這種方法來預測此小節的和絃為何。

三. State(i, j) - 小節內的分析部分：

總共有三個特徵參數

(1)Duration：小節中前三長的音符視為重要的音符

Ex - | 1 1 3 2 6 6 5 - | 較重要的音符為 => (1 , 5 , 6)

(2)組成音：和絃組成音能包含最多音符的視為重要的和絃

Ex - | 1 1 5 5 6 6 5 - | 可能的和絃為 =>

C (1, 3, 5) 或 A (6, 1, 3)

(3)強弱拍：強拍的音符視為重要的音符。若以八分之一拍為單位時，則我們視第 1,3,5,7 拍為強拍，其餘為弱拍。

Ex - | 1 1 5 5 6 6 5 - | 較重要的音符為 => (1 , 5 , 6)

接著我們將特徵參數的音符與和絃的組成音對照，能包含特徵參數之音符的和絃越多，就表示此和絃為正確答案的可能性越高。我們將上述三種方法所擷取出的音符分別與組成音對照，接著再將他們整合在一起成為最後的 State Probability。

其流程圖如下：

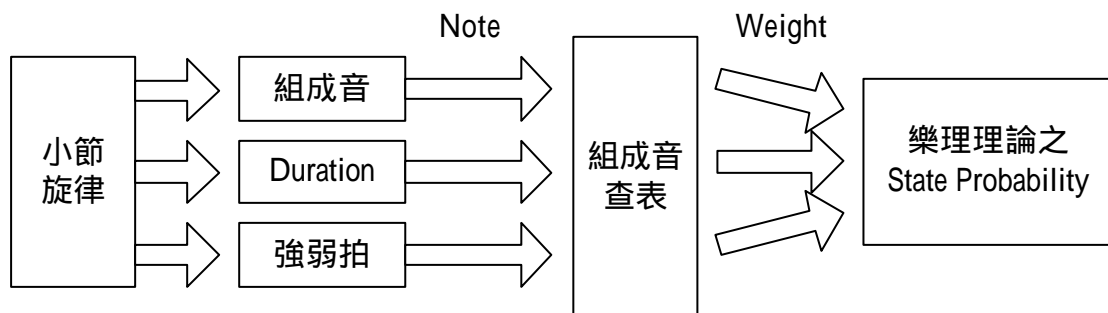


圖 4-5 樂理演算法之機率計算流程圖

我們用以下的例子說明之。一個未知小節其音符如下：

Measure = | 1 1 5 5 4 4 3 2 |

Step 1： Feature extraction：

(a)Duration：較重要的音為 (1, 4, 5)

(b)組成音：可能的和絃為 C (1, 3, 5)

(c)強弱拍：較重要的音為 (1, 3, 4, 5)

Step 2：組成音查表：

將 Duration 及強弱拍所擷取出的音符與和絃組成音表作對照。由於和絃的定義就是找出與主旋律最和諧的聲響，因此我們可以假設若小節中的音符與和絃組成音越相似，其所發出的合音一定較為理想。所以我們就將小節中的音符與和絃候選人的組成音作比對，若組成音中能包含其小節的音符越多，那此和絃就是我們所預測的最佳解：

Feature	音符	可能和絃
Duration	(1, 4, 5)	C(1, 3, 5) 或 F (4, 6, 1)
組成音		C(1, 3, 5)
強弱拍	(1, 3, 4, 5)	C(1, 3, 5)

我們將會根據經驗設計一組基本的權重，然後再將這三組特徵做權重的總和：

	C	#C	D	#D	E	F	#F	G	#G	A	#A	B
Duration	20%	3%	10%	3%	10%	20%	3%	10%	3%	10%	3%	5%
組成音	45%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
強弱拍	45%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
Total	36%	4%	6%	4%	6%	15%	4%	6%	4%	6%	4%	5%

最後的 Total 即為根據音樂理論所得到的 State Probability。

四. $\text{Transition}(i, j)$ - 相鄰小節間之分析部分：

我們可以根據樂理的“和絃行進法則”來制定各個和絃在樂理方面的 Transition Probability。和絃與和絃間的進行，為了使演唱者能夠很快的找到演唱的音高，所以在和絃行進上有必要將和絃的共同音(相同音高)保留下來，如此也可避免演奏者在彈奏時其和絃間的距離不會太遠。也就是說和絃的行進，轉換，盡量以最少的動作來完成。

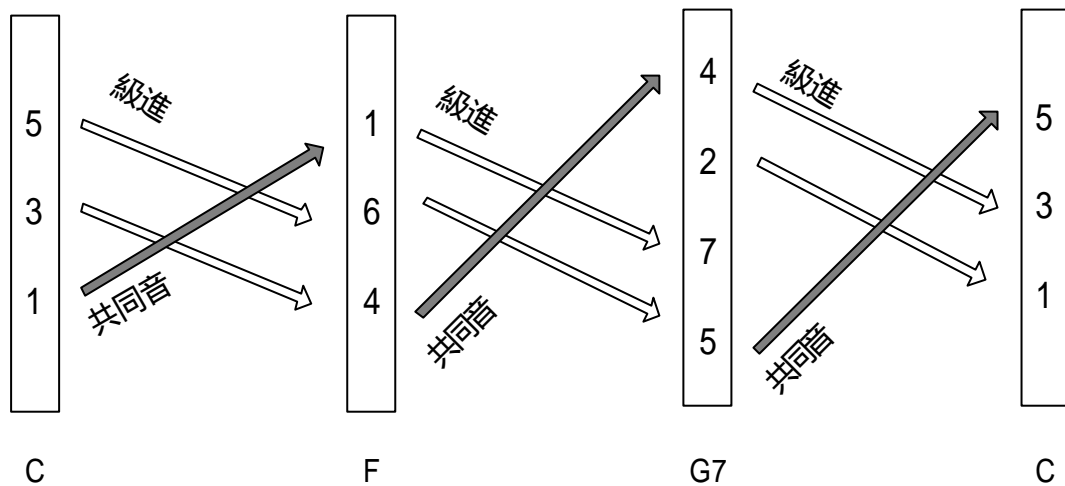


圖 4-6 和絃行進之範例圖

由此我們可得知，若和絃間有相同音或是相鄰音的話，他們之間的 Transition Probability 將會比其他沒有相同或相鄰音的和絃來得高。

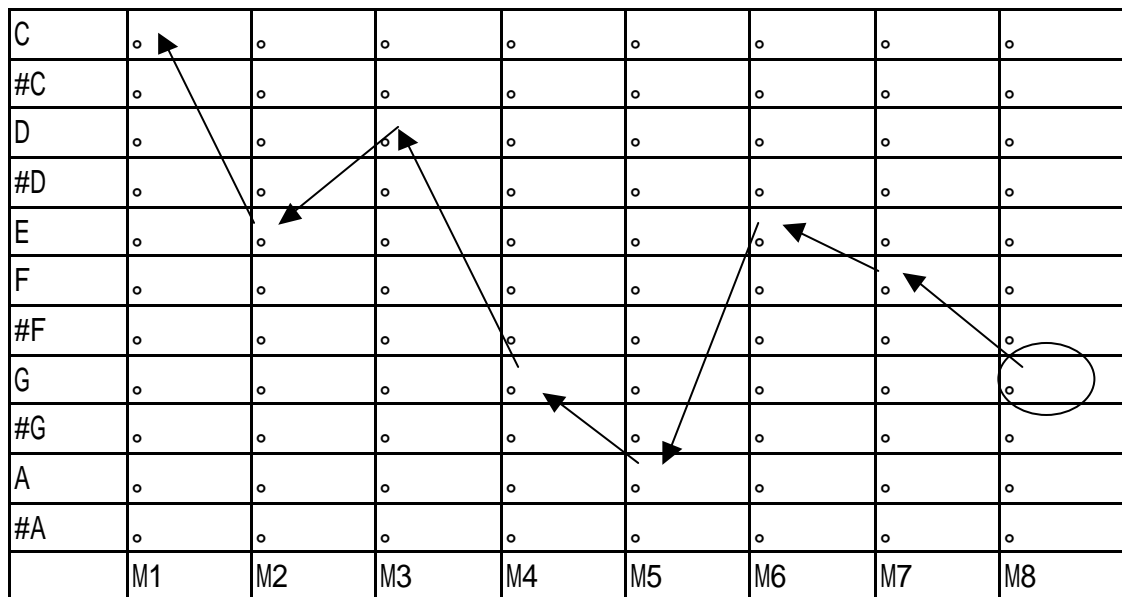
Dynamic programming 演算法

本實驗利用 Dynamic Programming 方法來找出每小節的最佳和絃候選人。由於系統在尋找最佳解時，需要參考到小節內與小節之間的關係，因此利用 Dynamic Programming 尋找最佳路徑是最好的方法。在本論文中，我們以 8 小節的和絃的機率總和 D 為最大時，為找到的最佳路徑，其數學式如下：

$$D(i, j) = state(i, j) + \max_{k=1 \sim 12} \{ D(i-1, k) + trans_i(k, j) \} \quad 1 \leq i \leq 8 \quad 1 \leq j \leq 12$$

其中 $D(i-1, k)$ 是前一步之最大和絃機率總和值，而 $D(i, j)$ 則為小節內與前一步最大機率之機率總和。當執行到最後一小節時，我們將找出最大的機率總和，然後再 Back tracking 至第一小節，以找出和絃的最佳路徑。

示意圖如下：



Mx : 小節數 $1 \leq x \leq 8$
。 : State Probability
→ : Back Tracking
○ : 最大機率總和

經由上圖的 Dynamic Programming 程序我們就可找出一條最佳的和絃路徑。

第五章 實驗結果

前言

由於每個人對音樂的定義喜好各有不同，相同的音樂，不一定每個人都覺得悅耳動聽，因此本論文的實驗分為兩方面：

1. 以和絃為主的辨識實驗：

由於本論文是以和絃為主要的辨識重點，因此我們會將預測和絃與標準答案作比對，以求得辨識率。

2. 利用問卷調查：

若只以和絃是否正確來決定音樂是否悅耳動聽，似乎不太合理，因為本論文主要目的是要做出一般大眾都能接受的和絃伴奏，因此我們將所有辨識結果與標準答案皆製作成 MIDI 檔，並製作問卷請人填寫，以求得較客觀的實驗結果。

和絃為主之辨識實驗

實驗之測試資料為 150 首八小節之 midi 檔案，皆是參考坊間的歌本資訊。因此我們總共有 1200 個小節 ($150 * 8 = 1200$)，每個小節皆有他們的和絃資訊。

根據辨識系統，我們共有兩種演算法(統計及樂理演算法)，每種演算法都有各自的 State probability 及 Transition Probability，因此我們可將這四種和絃機率權重作排列整合，並執行 Dynamic Programming 找出最佳和絃路徑。本實驗我們將分別列出不同排列組合所得的辨識結果。

我們將有四種不同的和絃機率權重，分別如下：

1. Stat State : 統計方法之 State Probability
2. Stat Tran : 統計方法之 Transition Probability
3. Rule State : 樂理方法之 State Probability
4. Rule Tran : 樂理方法之 Transition Probability

根據樂理，有些小節中的某些和絃彼此之間是可合理變換的，因此下表除了顯示各種排列整合的辨識結果外，最後一欄既為加入此樂理觀念後所得的辨識率：

	State Probability		Transition Probability		辨識率	加入變換和絃的觀念
	Stat	Rule	Stat	Rule		
1				V	11.75%	23%
2			V		33.33%	44.25%
3		V			53.08%	58.58%
4	V				46.33%	51.83%
5			V	V	33.33%	44.25%
6		V	V		40.5%	50.25%
7	V	V			54.83%	60.67%
8		V		V	24%	30.83%
9	V			V	46.25%	51.58%
10	V		V		53.16%	59.08%
11		V	V	V	43.42%	52.91%
12	V	V	V		57.33%	63%
13	V		V	V	50.50%	56.5%
14	V	V		V	51.83%	56.91%
15	V	V	V	V	55.08%	61.08%

圖 5-1 和絃辨識之實驗結果

若我們將上表中辨識率最高的組合(第 12 列)作細部的權重微調時，其辨識率可達 61%，再加入變換和絃觀念後其辨識率可達 65%。

上表我們可得到下面幾點結論：

1. 最高的辨識率出現在第 12 列，將統計與樂理的 State Probability 整合後，利用統計的 State Probability 執行 Dynamic Programming 所得的結果。
2. 樂理的 Transition Probability 對於辨識率似乎沒有較大的幫助。
3. 加上樂理觀念的辨識率平均可增加 6%的辨識率。
4. 對於小節與和絃的關係，似乎小節內(State)的資訊會比小節間(Transition)的來得重要。

利用問卷調查之實驗結果

因為每個人對於音樂的喜好皆有不同，而本實驗的主要目的為如何找出大眾皆能接受的和絃伴奏，因此問卷的製作調查是絕對不可缺少的。

我們將資料庫中 150 首 Midi 檔之主旋律擷取出來，然後將標準答案與預測答案皆作成 MIDI 檔作為問卷調查的內容，讓人們聆聽後寫下他們認為較好聽的版本。其示意圖如下：

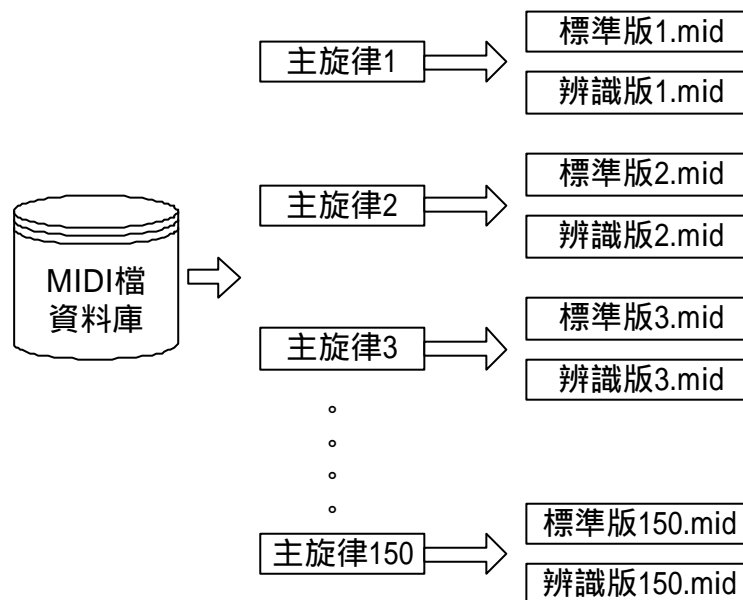


圖 5-2 問卷製作示意圖

而此時的辨識標準則修改如下：

1. 若受訪者認為標準版的 MIDI 檔較好聽時，則視為辨識失敗。
2. 若受訪者認為辨識版或是兩者皆好聽時，則視為辨識成功。

在和絃辨識率僅 55%的情況下將標準答案與辨識和絃製成 MIDI 檔時：

受訪人數：

共有 18 人，其中有 3 人較有樂理觀念，每個人皆聽 30 首 MIDI 檔。

結果如下：

1. 150 首 MIDI 檔的整體辨識率(不考慮是否有樂理概念)

=> 68.21%

2. 某 30 首 MIDI 檔分別給一般人及樂理知識的人聆聽時

一般人所得的辨識率為(人數為 2 人)

=> 60%

具有樂理知識的辨識率為(人數為 3 人)

=> 36%

由上述結果可知，在辨識版本所作出的 MIDI 檔中，約有七成的 MIDI 檔是一般大眾都覺得好聽的。但若把具樂理觀念的人分開時，辨識版本的 MIDI 檔就較不被具樂理觀念的人所接受了。

第六章 結論與錯誤分析及未來方向

結論

根據上述實驗，我們可得下列幾項結論：

1. 雖然單就和絃的辨識率最高僅達 57%，但若用人們實際聆聽 MIDI 檔，卻可達 68% 的辨識率，代表標準答案的辨識並不能直接判斷音樂是否動聽悅耳。
2. 顯然經由電腦辨識出的和絃伴奏，與專業樂理人的水準似乎還有一段差距，但就一般大眾而言，利用電腦來製作的和絃伴奏結果似乎還頗能接受。

錯誤分析

本實驗的辨識率並不高，其主要的我們歸納如下：

1. 由於資料的標準答案取自坊間的音樂歌本，但歌本的品質卻良莠不齊，使得辨識效果不彰。
2. 歌曲的曲風多變，增加辨識難度。
3. 辨識率不容易訂定，如何才能適切地定義辨識率是一個難題。
4. 音樂過於主觀，辨識結果差的人們不一定覺得難聽。

未來方向

1. 尋找更有力的特徵，以增加辨識率。
2. 慎選作為標準答案之歌本，以提高標準答案的品質。
3. 先將歌曲進行分類，以解決由於曲風多變所產生的問題。
4. 尋求更客觀的辨識標準，使得實驗結果更具意義。

參考文獻

- [1] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. “*Query by Humming Musical Information Retrieval in an Audio Database*” Proc. ACM Multimedia, San Francisco, 1995.
- [2] 曾明賢、王駿發, ”不特定歌者卡拉 OK 歌曲辨識系統”, MS Thesis, National Cheng Kung University, Taiwan, R.O.C.
- [3] Onishi, G. , Niizeki, M., Kimura, I., Yamada, H. ”A *Kansei model for musical chords based on the structure of the human auditory system* “IEEE International Conference on Neural Networks, 2001.
- [4] Borching Su and Shyh-Kang Jeng , “*Multi-Timbre Chord Classification Using Wavelet Transform And Self-Organized Map Neural Networks*” , IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001