國立交通大學

電機資訊學院 數位圖書資訊學程

碩士論文

敲鍵式音樂資料檢索系統-觀音

A Query by Tapping Music Retrieval System-KUAN YIN

研 究 生：蕭樹人

指導教授：柯皓仁　教授

中 華 民 國 九 十 四 年 七 月

敲鍵式音樂資料檢索系統-觀音
A Query by Tapping Music Retrieval System-KUAN YIN

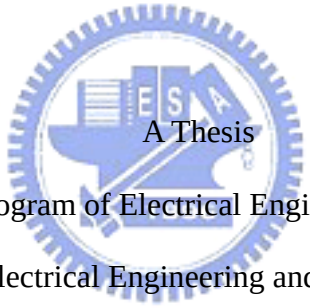
研 究 生：蕭樹人　　Student：Shu-Jen Hsiao

指導教授：柯皓仁　　Advisor：Hao-Ren Ke

國 立 交 通 大 學

電機資訊學院 數位圖書資訊學程

碩 士 論 文



A Thesis

Submitted to Degree Program of Electrical Engineering and Computer Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Digital Library

July 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年七月

敲鍵式音樂資料檢索系統-觀音

研究生：蕭樹人　　指導教授：柯皓仁博士

國立交通大學電機資訊學院 數位圖書資訊學程(研究所)碩士班

摘要

在查詢音樂形式資料時，大部分的檢索系統需透過與該音樂相關之文字詮釋資料進行關鍵字檢索，如歌曲名稱、作曲家、演奏者等，但是在使用者只知道音樂片段旋律的情況下，傳統檢索方式將無法直接檢索這種非文字型態的音樂資料。敲鍵式音樂檢索系統為一種內容式音樂檢索系統（Content-Based Music Retrieval, CBMR），它提供了一個全新的查詢方式，系統預先擷取每一首歌曲的關鍵旋律，並將其節奏中相鄰音符間隔之時間差存放在資料庫，藉由使用者在查詢時持續敲擊鍵盤產生的時間間隔長短不一之節奏，透過適當之近似比對（Similarity Matching）技術，能得到節奏相近的音樂原件。這篇論文將展示如何建構一個敲鍵式音樂檢索系統，系統以〝觀音〞為名，旨在希望將人們腦海中的音樂聲音，透過很直覺之方式將其轉換成節奏，以敲擊鍵盤輸入查詢，系統依相似程度排序回應音樂之曲名、作曲者、聲音等查詢結果，讓眼睛能直接觀看到這段心裡的聲音。

在查詢音樂資料時，敲鍵式系統紀錄每一次按鍵到下一次按鍵的時間差表示音樂旋律中每個音符長短不一之節奏，不同於一般以輸入音高變化為主的內容式音樂查詢系統，它更適合用於沒有經過嚴格音感訓練的愛樂者，即使欠缺準確辨別音高的能力，也能藉由每一段音樂都擁有的另一項音樂特徵即〝節奏〞，利用敲鍵的律動而檢索出對應的音樂資料。

關鍵字：內容式音樂檢索、敲鍵式音樂檢索、近似比對、觀音。

A Query by Tapping Music Retrieval System-KUAN YIN

Student：Shu-Jen Hsiao Advisor：Dr. Hao-Ren Ke

Degree Program of Electrical Engineering Computer Science

National Chiao Tung University

Abstract

When we have requirement for music data retrieval by an OPAC (Online public access catalog) system, in most systems the user needs to fill out some text-based metadata, like title, composer, and so on , in the query process. Sometimes the user can only remember fragments of music content. So the traditional text-based OPAC system can not satisfy this kind of content-based requirement. A query by tapping mechanism is designed for content-based music retrieval (CBMR). The system extracts key melodies and saves to the themes database with beat information (durations of note) in advance. Then the system takes the user tapping on the keyboard information as query string vector. Apply appropriate similarity matching algorithm to get the intended song in database. This paper presents a query by tapping CBMR system called KUAN YIN. The main purpose is to provide an intuitive way for the user to organize query string via tapping keyboard and get the result by similarity ranked list. All text-based metadata that associates with the intended song will be displayed on the monitor. So KUAN YIN can make visible for the melody linger in mind.

The system records time interval between two consecutive key pressed as the music feature rhythm. It is different from other popular CBMR systems which extract the music feature pitches. Since there is no pitch information in the retrieval process, the system is more suitable for an amateur user. Pitch profile is difficult for people without formal music training to perceive correctly. KUAN YIN takes music rhythm information for comparison with every key pressed tempo which user tapping.

Keywords: content-based music retrieval, query by tapping, similarity matching, KUAN YIN.

誌謝

前年七月，剛通過入學考試，滿心期盼開學後新的人事物；去年七月，歡欣準備迎接家庭中即將加入的新成員；今年七月，亦步亦趨如期完成了我的碩士論文。回顧這短暫且緊湊兩年求學生涯，一路走來身旁始終伴著扶持我的一群，千萬個感謝也只能感嘆言寡而情濃。

謝謝指導教授柯皓仁老師，不時的對我指導，整個研究過程有如進行曲般，以雄壯規律的節奏進行，步伐整齊，堅定信念朝目標邁進。謝謝黃明居老師及所有同學不吝對我的鼓勵，有了你們的加油支持，讓我能更盡情的發揮表演。

謝謝永遠在背後默默支持我的家人，女兒亮心宏亮哭聲，有如歌劇女高音詠嘆調般，總能激昂我疲乏的靈感，妻子芳如對家庭無微不致的付出，就像巴洛克音樂中常出現的頑固音型❶，靜逸的在低音聲部烘托著主旋律。

中華民國九十四年七月十一日

---

❶它為一種音樂創作手法，由固定音符所組成，常位於低音聲部，用來搭配變化豐富地主旋律，有時不易被察覺，不斷反覆出現，貫穿整首歌曲，雖然單調缺乏變化，但是卻讓歌曲多了一種沉穩安定的力量。

目錄

表目錄

圖目錄

第一章 簡介

第一節　內容式音樂檢索系統

搜尋引擎是一般人在有資訊檢索需求時第一個想到的工具，藉由使用者提供關鍵字，搜尋引擎會檢索與這些關鍵字相關的文件。不可否認地，搜尋引擎在進行關鍵字檢索的表現上，已經有令人滿意的效率及結果；然而目前多媒體數位物件技術日趨成熟，愈來愈多非文字型態的多媒體資料成為數位圖書館的典藏對象，人們的資訊需求不再侷限於文字型態的資料，這些多媒體文件已逐漸成為搜尋對象，但到目前為止，多媒體文件的檢索技術仍未臻成熟。本論文主旨在研究如何設計一套音樂檢索系統，透過非文字型態的方式直接檢索音樂內容，讓使用者能夠很容易地找到其所想要的音樂資料。

目前常見的音樂檢索方式依然以傳統文字型態的詮釋資料為主，系統要求使用者在曲名、作曲者等任一欄位輸入關鍵字，利用文字比對方式，找出詮釋資料與關鍵字有關之音樂原件，這種檢索方式最大的缺點在於，使用者必須在檢索之前就要知道任一與歌曲相關的詮釋資料，方能輸入查詢條件，若當使用者在檢索時只能回憶部分音樂片段，或是突然聽到一段優美旋律而不知道曲名時，使用者則無法提供任何與音樂相關之文字詮釋資料，故文字型態的檢索方式將無法滿足這種只知悉音樂內容，而對於音樂相關詮釋資料毫無概念之檢索需求。

直接利用音樂內容進行檢索的音樂檢索系統又可稱為內容式音樂檢索系統（Content base music retrieval, 簡稱 CBMR），音樂內容本身包含了大

量的資訊，最常被挑出來描述音樂特徵的屬性有旋律（Melody）和節奏（Rhythm），擷取這些音樂特徵存放在資料庫中，使用者在查詢時輸入部分的音樂特徵屬性，系統將比對資料庫中所有特徵屬性值，並回應給使用者近似的音樂資料，整個查詢過程中使用者不需輸入文字型態的詮釋資料，取而代之的是輸入音樂內容的特徵屬性。

第二節　問題陳述

以資料內容為主的查詢方式，在每筆資料物件匯入資料集（Corpus）時，均要對該資料物件進行索引詞（Index Term）之建立。如同文字型態資料物件，當建立了適當的索引詞，全文檢索的執行效率及準確度將大幅提升；同樣地，為了提昇內容式音樂檢索系統的效率與準確度，也必須建立合適之關鍵索引片段（Key Index Piece）。在擷取音樂資料中的關鍵索引片段或稱主題（Theme）片段時，是不是可能利用程式自動地抓出合適之索引片段？

當一段旋律在我們腦海中回憶起時，如何透過電腦的輸入設備將音樂片段輸入系統內，也是一個需要克服的難題。舉例來說最直覺的輸入方式為透過麥克風以「哼」或「唱」的方式來查詢音樂資料庫，也可稱為哼唱式檢索系統（Query By Humming, 簡稱 QBH）〔1〕〔2〕〔3〕，但是要準確地哼出心中之旋律，對平常沒有歌唱習慣的使用者，哼出的調性旋律不一定容易被系統辨識；此外，也有一些系統〔3〕〔4〕〔5〕提供輸入簡譜的方式查詢音樂，這對沒有經過音感訓練的門外漢來說更是難如登天。

文字型態資料的比對，一般要達到查詢字串（Pattern）與被比對文件

中的子字串（Sub String）完全相同，才可將該文件視為符合檢索條件的文件，這種比對又稱完全比對（Exact Matching）。但是音樂資料的比對不能用完全相同來表示符合檢索條件的音樂資料，人們記憶的音樂片段可能無法如音樂資料中的音樂片段般地完整，很有可能多幾個音符或少幾個音符，甚至原始音樂資料中許多裝飾音及合聲部份均不易被辨別或記憶，所以近似比對（Similarity Matching）技術較適合用於音樂資料的比對。

第三節　研究目的

本論文主要的研究目的即是探討如何去針對音樂資料的內容，設計一個可行的自動索引策略，並能讓使用者用很直覺的方式產生查詢條件，系統透過合適之近似比對演算法，依相似度排序回應近似的音樂資料。

Witten 曾指出〔6〕，針對不同性質之資料必須各別設計特殊的索引策略，所以不管資料型態為文字、圖片、聲音或影片，勢必都必須針對這些不同型態的資料各自別發展一套合適的索引策略。本論文將研究的範圍限定在聲音資料中的音樂資料，在眾多的的音樂資料格式中選擇 MIDI（Musical Instruments Digital Interface）〔7〕格式為自動索引研究之對象，研究的目的包括：

1. 音樂的關鍵索引片段，為該片段在整首音樂中易被聽者記憶的部分，所以該部分也容易被使用者拿來當作檢索條件，是否有規則去擷取這些存在音樂資料裡的關鍵索引片段呢？
2. 構成音樂的三大要素為節奏（Rhythm）、旋律（Melody）、和聲（Harmony），在判斷關鍵索引片段時，該以哪個音樂屬性為主。

另外還需廣泛研究現行發展之內容式音樂檢索系統，了解其運作原理並比較各系統之優劣。為了讓使用者簡單地表達出腦海中記憶不全的音樂片段，在設計系統時也必須考慮到如何讓沒受過音樂訓練的使用者，能利用很直覺的方式輸入欲檢索的音樂片段。論文中也討論在不同系統中使用的近似比對演算法，比較之間的差異，分析是否需發展更合適之演算法。

第四節　論文架構

本論文架構如下：第二章敘述內容式音樂檢索系統的相關研究工作；第三章闡述觀音系統的架構及採用的演算法；第四章說明為評估觀音系統所做的實驗，以及實驗結果的討論；第五章總結本論文並提出未來研究方向。

第二章 相關研究工作

內容式音樂檢索系統在國內外均有相關研究在進行，在這些研究工作中，大致可分為三個問題領域進行探討：

1. 音樂屬性的擷取：一個音樂媒體資料是由多種不同的音樂屬性所組成，目前常見用在各種研究中的音樂屬性包含：
   (1) 旋律（Melody）：即擷取音樂之音高屬性，爲了避免調性的差異，也有些研究是採音程（Interval）表示音樂的旋律〔4〕〔5〕〔8〕〔9〕〔10〕。
   (2) 旋律輪廓（Melody Contour）：將歌曲之旋律依音高升、降、持續的變化情形分別以 U、D、R 三個字母表示〔2〕〔3〕。
   (3) 主旋律（Perceived Melody）：若歌曲由多個聲部所構成，聽者感受到的旋律片段往往會在不同聲部之間切換，組合這些旋律片段即為歌曲之主旋律，在擷取旋律屬性時，要先分析如何在多個聲部中取出單一之主旋律〔11〕。
   (4) 節奏（Rhythm）：歌曲中相鄰音符的時間差〔1〕〔12〕。
2. 使用者查詢介面的設計：研究方向旨在提供使用者利用直覺的方式來輸入查詢音樂資料，常見的查詢介面包含：
   (1) 彈奏（Query by playing）：使用者直接透過 MIDI 樂器彈奏一段查詢旋律〔13〕。
   (2) 輸入（Query by typing）：要求使用者直接在查詢介面上輸入音高簡譜字串或旋律輪廓〔3〕〔4〕〔5〕。
   (3) 哼唱（Query by humming）：使用者直接透過麥克風哼唱出查詢片段〔2〕〔3〕〔14〕。

(4) 敲擊（Query by tapping）：使用者在 MIDI 電子鼓或麥克風等輸入設備上敲擊出歌曲之節奏〔1〕〔12〕。

3. 近似比對演算法：受限於人耳聽力敏感度高低、回憶音樂片段記憶能力、及查詢音樂時的表達能力，再加上音樂原件也有可能因為不同詮釋方式而呈現出不同的音樂特徵屬性，所以針對這種資料的特性，近似比對技術被視為基本檢索需求，常見的近似比對演算法包含：編輯距離（Edit Distance）〔1〕〔2〕〔9〕、EMD（Earth Mover's Distance）距離〔15〕、直接量測（Direct measure）〔12〕、n-note〔4〕〔5〕〔8〕等。

在以下各節將針對幾個代表性的研究，討論其核心技術，並作一比對敘述。

第一節　哼唱式檢索系統 QBH

Ghiast 等人〔2〕早在 1995 就開發了一套名為 QBH（Query By Humming）的系統，此系統提供使用者透過麥克風以哼唱的方式查詢音樂資料，系統將由麥克風輸入的聲音訊號，轉換成包含了 U（這個音比前一個音高）、D（這個音比前一個音低）、R（這個音和前一個音相同）三個符號所組成的查詢字串（即旋律輪廓），同時也擷取旋律輪廓作為音樂資料的特徵屬性，旋律輪廓描述音高變化方向，同樣也以 U、D、R 表示，比對資料時利用動態程式規劃（Dynamic Programming）之近似字串比對（Approximate String Matching method）計算查詢字串向量與資料字串向量之編輯距離。所謂編輯距離就是將兩個字串向量利用 Insert、Delete、Replace 三個步驟，能轉換成相同字串之最少步驟個數，也可以說是使用最少的字元操作，使得一字串轉變成另一字串的操作次數，又可稱為 Levenshtein distance〔16〕。編輯距離的計算方式可依下面 4 個步驟進行：

1. 假設查詢字串向量 P 長度為 m，資料字串向量 T 長度為 n。

2. 設定編輯距離之初始值：

   (1) d[0，i]=i　　0 <= i <= m

   (2) d[j，0]=j　　0 <= j <= n

3. 將兩個字串從頭比到尾，則比對到 P 的第 i 個字與 T 的第 j 個字之編輯距離可表示成：

   (1)先計算 cost 值，若兩個字母相同則 cost=1，否則 cost=0：

```
if   ( P[i] ==T[j] ) then
     cost=0
else
     cost=1
end
```

   (2)d[i，j]可以用下面的遞迴式表示，分別比較 d[i，j]上面 cell 之數字加 1，d[i，j]左邊 cell 之數字加 1，d[i，j]對角 cell 之數字加 cost，這三個值中取最小的數字表示為 d[i，j]：

```
d[i，j] = min ( d[i-1，j] + 1，
                d[i，j-1] + 1，
                d[i-1，j-1] + cost )
```

4. 最後 d[m，n]的值即為 P 與 T 之編輯距離。

以表 1 為例，假設查詢字串 P=UDDRU，資料字串 T=DDRDDU，計算其編輯距離 d[5，6]=3。

表 1：T 與 P 之編輯向量矩陣

		U	D	D	R	U
	0	1	2	3	4	5
D	1	1	1	2	3	4
D	2	2	1	2	3	4
R	3	3	2	2	2	3
D	4	4	3	2	3	3
D	5	5	4	3	3	4
U	6	6	5	4	4	3

這種設計方式允許使用者以任何音調哼唱歌曲，即使音高稍微不準只要音高在改變時的方向掌握正確即可，缺點是一般音樂在音高上均有複雜的變化，若單純用旋律輪廓來描述，則系統在查準率（Precision Rate）的表現會較差。此外，在計算編輯距離時需要的時間複雜度達 O（m*n），查詢費時及近似字串比對只允許由字串頭開始等都是待改善的缺點。

第二節　MELDEX

紐西蘭 Waikato 大學的 McNab 等人在 1997 年發表了一個名為 MELDEX 的系統〔3〕，近年來該系統並整合入著名的數位圖書館軟體 Greenstone〔17〕，使得 Greenstone 也擁有了檢索音樂資料的能力。如圖 1 所示，MELDEX 提供了更彈性的檢索方式，在〝Raw Audio Input〞頁面中，查詢方法為哼唱式查詢，使用者可對著麥克風哼唱，錄製一段查詢音樂片段；在〝Symbolic Input〞頁面為彈奏式查詢介面，使用者在顯示的鋼琴鍵盤上，彈奏出查詢片段。此外在比對音樂屬性時可以選擇旋律輪廓、音程、節奏中任一種為判斷標準，當然也可以組合兩種以上的判斷條件，甚至允許選擇要完全比對或近似比對。

當然此系統還是存在一些缺點，在哼唱時因為 MELDEX 尚無法正確地將音符切割開來，所以要自行在音符與音符之間加入「da」或「ta」聲，不符合一般歌唱的習慣。在彈奏式查詢的設計也是必須要有受過相當訓練才能彈出內心裡的旋律，每按下一個鍵還需搭配輸入每個音符的拍號長度，遇到休止符時也要再按下〝Add a rest〞按鈕，距離直覺的彈奏方式還有一段距離。另外系統採 Java Applet 架構設計，在使用前需要安裝合適的 Java Plug in，也會造成使用上的不方便。

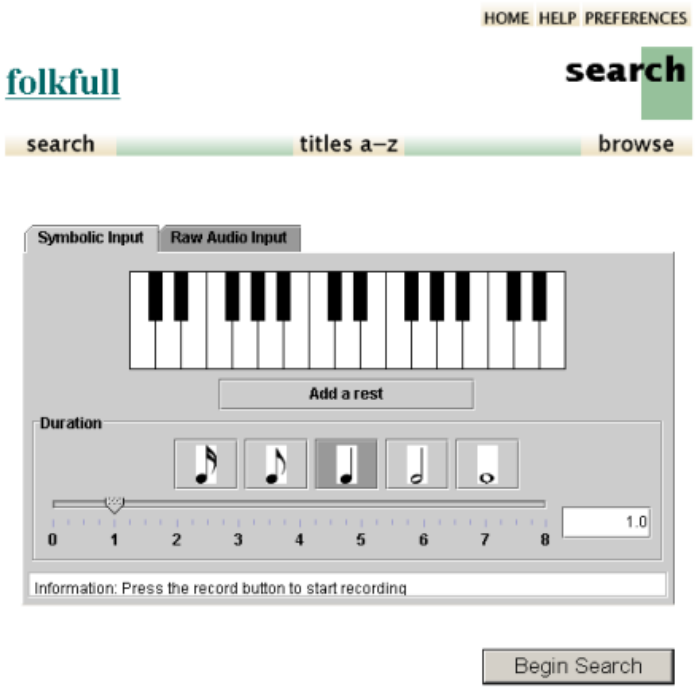


圖 1：MELDEX 操作介面
（資料來源：紐西蘭國家數位圖書館計畫
http://www.sadl.uleth.ca/nz/cgi-bin/music/musiclibrary）

第三節　超級點歌王

國內清華大學的張智星等人也開發了一套內容式音樂檢索系統〝超級點歌王〞〔1〕，此系統是以哼唱為主的音樂檢索系統，如圖 2 所示為使用一段系統提供的範例哼唱音樂片段 Let It Be.wav 進行查詢，所得到之查詢結果。

目前此系統也加入了敲擊式查詢的功能〔1〕，使用者利用敲擊麥克風的方式輸入節奏，擷取音樂資料中音符的間隔時間差（即節奏）為判斷條件進行比對，爲了克服一般輸入的節奏速度（Tempo）很難和資料庫內音樂資料之節奏速度一樣，所以在比對這兩個時間向量時，先將各項絕對時間值正規化成 1~1000 的整數（又稱 Linear Scaling 技術），還有可能因為輸入的音符多一個或少一個，所以將音樂資料庫裡欲比對的向量維度個數相對於輸入之個數加減 2 個維度，則輸入查詢之 t 向量與資料庫中參考的節奏片段 r 向量經過轉換後的 t’與 r’可表示為：

t’=round(1000*t/sum(t))

r’=round(1000*r(1~q)/sum(r(1~q))

// sum(t)=t 向量中所有維度的加總值

// q 為一個變動值，假設 t 的長度為 p，則 q 介於 p-2 與 p+2 之間

在計算 t’和 r’向量的相似度時採 DTW（Dynamic Time Warping）演算法〔14〕，每一個查詢向量 t’要和 r’做 5 次比對，分別比較 r 向量中第 1 個至第 p-2、p-1、p、p+1、p+2 等維度，計算其編輯距離表示向量之相似度，由實驗結果可以看出音樂之節奏屬性也為辨識音樂資料的一個重要屬性。

不過系統在判斷向量相似度時仍採用動態程式規劃法，這個方法較適

用於文字符號的資料比對，若用來判斷整數型態的資料，很多重要資訊將被忽略，如"100"與"101"是兩個完全不同的符號，但若視為整數 100 與 101 將是很相似的數字，這個問題將在本論文提出的研究方法中獲得改善。

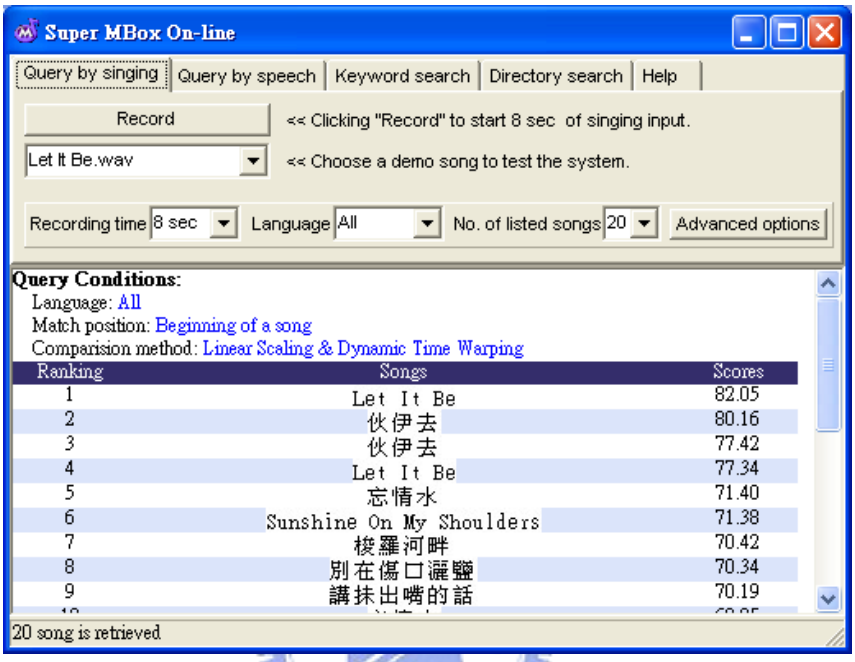


圖 2：超級點歌王操作介面

（資料來源：清蔚科技 http://www.supermbox.com.tw/）

第四節　Beat Bank

Eisenberg 等人在 2004 年發表一個敲擊式檢索系統 Beat Bank〔12〕，該系統使用符合 MPEG-7 標準（ISO/IEC 15938）〔18〕描述音樂資料之節奏，並要求使用者於 MIDI 設備上將節奏輸入，當然這些輸入的資訊一樣要轉換成 MPEG-7 相容的表示方式，同樣的這類敲擊式檢索系統一般都不考慮音樂中音高的資訊。雖然它使用了 MPEG-7 XML 文件的格式來描述音樂的節奏資訊，但在進行比對時一樣要轉換成兩個節奏向量，再進行相似度計算。向量中每個元素所代表的值不是音符與音符之時間間隔（超級點

唱王的表示方式)，而是用音符間隔時間之累積值表示，故節奏向量中每個元素值將大於或等於其前一個元素的值。Eisenberg 等人採用的近似演算法為直接量測法(Direct Measure)，依下面步驟可計算兩個節奏向量之近似值：

1. 假設查詢節奏向量 P 長度為 m，資料節奏向量 T 長度為 n。
2. 由開始處依序比較 T 和 P 兩個向量的每個元素，直到達到任一向量的尾端，Compare 變數表示比較的次數，Match 變數表示比較吻合的次數。

```
While not reach EOF T or P {
        Compare++
        if ( P[i] == T[j] )
            Match++ ; i++ ; j++
        else {
            if ( P[i] > T[j] )
                j++
            else
                i++
        }
}
```

3. P 與 T 之相似度 = 吻合的次數/比對的次數。

Similarity = Match/Compare

假設 m 和 n 分別代表兩個向量的長度，利用動態程式規劃法〔4〕〔5〕最多比對次數將達 m*n 次，若採 Direct Measure 則最多比對次數僅 m+n，故 Direct Measure 在比較向量相似度時，相較於動態程式規劃法有更佳的效率表現。但 Direct Measure 也僅適用於文字符號的比對，當兩個極為接近的數字拿出來比對時，雖然在拍子的感覺上非常類似，但在 Direct Measure

的比對下也視為一次失敗（不吻合）的比對，所以在設計敲擊式音樂檢索系統時，不僅僅要考慮兩個向量的整體相似度，另外元素與元素的相似度也是在判斷相似之重要因子。

第五節　擷取多聲部音樂之主旋律

在許多古典音樂中會利用和聲（Harmony）、對位（Counterpoint）等創作或演奏方式來增加聲音的豐富性，也就是說會有多個音符在同一時間內響起，調和後的音樂旋律將更能表現出聲音重疊的立體感。大部分的MIDI 檔中包含了多個聲部（Part），每一個聲部都代表一個樂器演奏的樂譜，有些樂器表演包含音高變化的聲部（如管絃樂器），有些樂器只單純負責敲擊節拍的部分（如打擊樂器），當多個聲部同時演奏時能產生樂團般演奏的效果，雖然有多個樂器在一起演奏，但是作曲家還是會將整首音樂要強調的主旋律藉由和絃或對位旋律烘托出來。根據一些音樂心理學的研究〔19〕發現，聽眾感覺到的主旋律不會一直集中在固定聲部，而是會在不同聲部間切換，而聽覺越是敏銳的聽眾其在聲部之間切換的頻率越是頻繁。若是依作曲家組織的方式去獨立分析每個聲部，則使用者若利用自己感覺到的旋律去檢索，將很難把目標音樂檢索出來，所以擷取重複片段時其旋律必須模擬人們是如何感覺到的才有意義，因為即使在和絃旋律上有大量重複片段，但聽眾在欣賞音樂時可能忽略的部分不容易拿來當做檢索條件，所以在分析重複片段前，要先決定音樂之主旋律，要決定音樂之主旋律，則須先探討聽眾注意力在不同聲部間切換是否有固定之規則。Uitdenbogerd 等人〔11〕爲了解決這個問題，設計了四種演算法：

1　在多聲部音樂裡，將每一個獨立聲部上所有音符集中於單一聲部中表

示，演算法擷取在同一時間要一起發聲之音符當中（在樂譜上以合聲音符表示），音高最高的音符（Top Note）當作構成主旋律的音符，產生之主旋律內每個音符為同一時間各聲部中最高音之音符，所以主旋律有可能會在不同聲部中切換。

2 每個聲部分別獨立處理，利用演算法一將各聲部中在同一時間要一起發聲之音符，擷取音高最高的音符當作構成該聲部之主旋律的音符，每個聲部的主旋律都決定後再計算各主旋律的平均音高（Pitch Average），將平均音高最高的聲部之主旋律視為整首樂曲之主旋律。演算法二產生之主旋律只在同一聲部內移行，不會在不同聲部中切換。

3 類似演算法二，每個聲部分別獨立處理，利用演算法一將各聲部中在同一時間要一起發聲之音符，擷取音高最高的音符當作構成該聲部之主旋律的音符，每個聲部的主旋律都決定後再計算各主旋律的熵資訊含量（First-Order Predictive Entropy），若構成該主旋律之音符組合為S_1，S_2，S_3，••，Sm，而Si/S表示該音符在整段主旋律中出現之機率，則該主旋律的熵資訊含量可以表示為：

$$I(s_1,s_2,\ldots,s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

將熵資訊含量最高的聲部之主旋律視為整首樂曲之主旋律，資訊含量高表示該聲部構成主旋律的音符有較大的亂度，相對的音程變化將較豐富，而較有規律的合旋伴奏聲部熵的含量較低。相同的演算法三產生之主旋律只在同一聲部內移行，不會在不同聲部中切換。

4 將每個聲部對應之位置切割成若干區段，區段之起點位於各聲部在同一時間內有多個要一起發聲之音符處，區段之終點則位於下一個區段的起始處，分別計算各聲部中每一段區段的熵值，主旋律在樂曲中移行時選擇各聲部中最大熵含量的區段切換。

Uitdenbogerd 等人的研究實際利用了上面四種演算法，應用在 10 首不同樂風的 MIDI 歌曲上，分別擷取每一首的主旋律，並由 8 位使用者感覺哪一種演算法所擷取出之主旋律較符合他們心中的樂曲主旋律。實驗結果發現，雖然沒有一種方法可以在每一種不同類型音樂中，均能正確地決定出主旋律，但是演算法一在不同樂風的歌曲上的平均表現最佳，所以直接把各聲部同時響起時最高音之音符當作構成主旋律的音符，是最簡單且最有效率的方式。

第三章　敲鍵式音樂檢索系統-觀音

在第三章中將探討透過本論文發展的敲鍵式音樂檢索系統〝觀音〞，第一節描繪系統架構；第二節說明觀音如何擷取音樂屬性並建立索引片段，參考過去相關研究〔20〕〔21〕〔22〕，觀音所擷取的音樂屬性為旋律輪廓，並嚐試利用詞尾樹（Suffix Tree）〔23〕，由旋律輪廓〔2〕〔3〕中分析重複片段，將該片段視為歌曲之索引值；接著第三節展示敲擊式查詢介面，在輸入查詢介面的設計上，由使用者敲擊鍵盤產生查詢節奏。觀察研究趨勢可以發現，在近期發展之內容式音樂檢索系統中，有慢慢地朝向利用節奏查詢音樂資料〔1〕〔12〕，取代早期利用音高查詢音樂資料的趨勢發展〔2〕〔3〕〔4〕〔5〕〔8〕〔9〕〔10〕，畢竟要使用者去感受音樂的節奏，要比能正確地分析出音高變化來的容易，而觀音系統也是一個典型之敲擊式檢索系統，並利用節奏屬性來進行音樂檢索比對；第四節說明系統採用的近似比對技術，採 n-note 分析兩段節奏向量之相似度〔4〕〔5〕〔8〕，先利用〝節奏商標準差〞（Rhythm quotient standard different）演算法計算兩段音樂向量中每個顆粒（Gram）之相似度，再利用 DICE 函數〔24〕計算整段向量之相似度。

第一節　系統架構

圖 3 是參考 Birmingham〔25〕所述的一個內容式音樂檢索系統架構，幾乎所有現行之系統都是按照這種架構在發展，觀音系統也可以用圖 3 所包含的模組一一說明，這些模組主要可分三類：

1. 儲存模組類（圓柱狀）：

(1) MIDI files of Full pieces：尚未處理過的原始音樂元件，本論文中所處理的音樂資料限於 MIDI 之音樂檔案格式。

(2) Meta data（Themes）：每一個 MIDI 的音樂資料都須經過適當的索引程序，擷取出該音樂的關鍵旋律並視為索引片段，在本論文中這些用旋律表示的關鍵旋律片段還需要再轉換成節奏型態，將原來用相對音高描述的旋律，轉換成用音符開始時間差描述的節奏，而這些轉換後的資料節奏即為一首音樂資料的索引片段。將這些索引片段存入 Themes 資料庫中，以加快資料檢索之速度。

2. 程序模組類（四角形）：

(1) Theme extraction：關鍵旋律擷取之程序可分手動與自動兩種，手動方式即是先將歌曲聆聽數次，再靠聽覺感受去定位關鍵旋律片段之位置；而自動模式適用於結構形式較單純之歌曲，觀音系統利用 Suffix tree 分析音樂旋律輪廓屬性，以擷取重複片段。

(2) Transcription：將使用者輸入之查詢字串轉換成和 Themes 資料庫相同的節奏時間單位值，系統使用之時間計量單位為 1/1000 秒。

(3) Retrieval Engine：利用合適之近似比對演算法，比較輸入之查詢節奏與音樂庫中資料節奏的差異程度。

3. 輸入輸出模組類（四圓角形）：

(1) Query：使用者只要在輸入方塊中，依印象中的節奏速度，有節奏地在鍵盤上敲出每一個音符，不用費力地彈奏出每個音符之音高，系統會自動紀錄相鄰二次按鍵被按下時的時間差，並將這些時間差視為查詢節奏。

(2) Ranked list of pieces：將符合最小相似度要求之歌曲全部輸出，查詢結果依相似度高低順序排列。由於使用者輸入的查詢節奏有可能較資料庫的資料節奏片段更長或更短，故在比較前要以兩段節奏中較短的節奏為主，比較也只限於由第一個音開始。

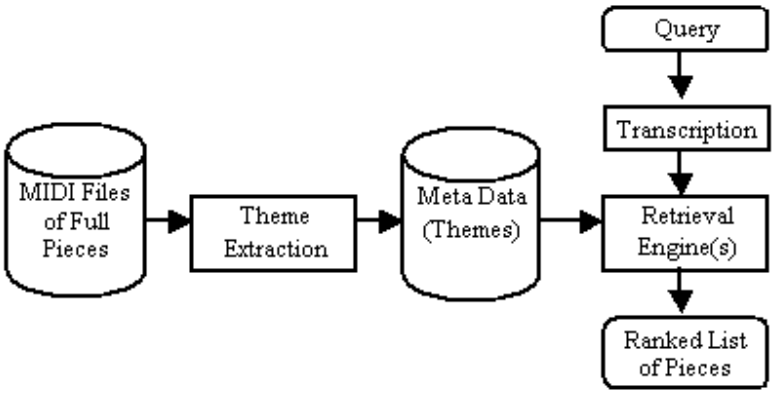


圖 3：系統架構

第二節　擷取屬性及建立索引片段

當歌曲的結構較單純時，如只有單聲部演奏，節奏旋律變化明顯且不加入即興片段特徵之歌曲，在觀音系統中特別針對此類音樂資料開發了一個 Theme extraction 模組〝MIDI Miner〞，目的在能自動地從這些歌曲結構單純的 MIDI 音樂中，擷取出關鍵音樂片段，並將該片段視為歌曲之索引值。自動索引的程序可以分為以下四個步驟進行：

1. 收集符合 General MIDI 規範〔7〕之 MIDI 檔，並將其內容用每一個 MIDI 事件（Event）發生之時間順序表示。
2. 針對每個 On Note 事件之音高值，分析其旋律輪廓。
3. 利用詞尾樹演算法，分析歌曲旋律輪廓之重複片段。
4. 將旋律輪廓重複片段轉換成節奏型態。

以下四小段分別說明這些步驟的詳細過程，首先將 MIDI 檔透過〝MIDI file DisAssembler〞〔26〕程式，將 MIDI 格式之音樂資料，轉換為程式較易剖析之純文字格式，在該文字檔中會依時間發生順序紀錄每一個 MIDI 事

件，如圖 4 為童謠〝小星星〞轉換所得之文字檔案，注意當在每個 Note On 事件發生時（即音符開始發聲之時間），需要同時指定該音符位於之聲部（Chan），該音符之音高（Pitch），與音量值（Vol）。

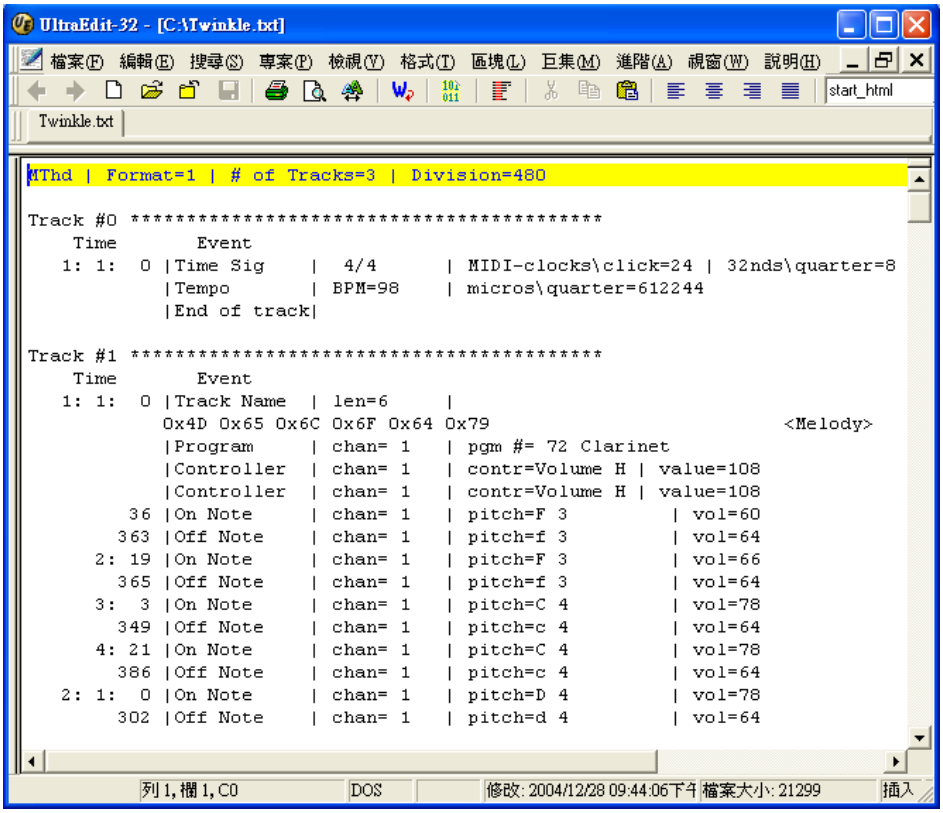


圖 4：小星星之 MIDI 文字檔

接下來在第二個步驟中要擷取 MIDI 中的旋律輪廓音樂屬性，MIDI Miner 首先會先抓出歌曲中依時間次序出現的音符音高資料，再將後一個音符的音高減去前一個音符之音高，以得到整首歌曲的音程變化值，程式在分析關鍵旋律前將音高轉為音程，可以避免同一首歌曲在不同調性間轉換，造成每個音符絕對音高不同的困擾，不同調性的轉換，旋律感覺幾乎相同，因為音符和音符間相對的音高距離（音程）沒有改變，對聽者而言只是有較高或較低音的差別〔11〕。MIDI Miner 在剖析 MIDI 文字檔時只會擷取每個 Note On 時音高值變化的情形，將音高升、降、持續的變化情形

分別以 U、D、R 三個字母表示，以小星星的前兩個 Note On 為例，第一個 pitch=F3；第二個 pitch=F3，則這兩個音之旋律輪廓為 R，整首歌曲計 42 個音，故可表示成下面 41 個字元之旋律輪廓：

RURURDDRDRDRDURDRDRDURDRDRDDRURURDDRDRDRD

在第三個步驟中即要分析這個旋律輪廓字串中的關鍵旋律片段，根據音樂心理學的研究認為作曲家普遍大量運用重複片段（Frequent Pattern）〔27〕，而這些重複片段一般均為整首曲子之精華所在，也是聽眾最容易記得的部分，因此使用者在檢索歌曲時最常用的查詢旋律，應該就是這些大量出現的重複片段，所以在設計檢索系統時若能將音樂資料的重複片段擷取出來，視為該音樂的索引片段，對檢索的效能將有很大的改善。在本論文中利用詞尾樹（Suffix Tree）演算法〔22〕〔23〕，取得字串中的重複片段，即出現次數大於 1 次，且必須包含 2 個字元以上之子字串，若該子字串也出現於包含它且較長之重複字串，則該子字串不算重複片段，假設 S 代表旋律輪廓字串，則詞尾樹 T 定義如下：

> Def：A suffix tree T for an m-character string S is a rooted directed tree with exactly m leaves numbered 1 to m. Each internal node, other than the root, has at lease 2 children and each edge is labeled with a nonempty substring of S. The key feature of the suffix tree is that for any leaf i , the concatenation of the edge-labels on the path from the root to leaf i exactly spells out the suffix of S that starts at position i . 〔23, p90〕

以 S=UDDRUDDRDDR$為例（$符號用來表示字串的結束）可依下面程序將 S 表示成詞尾樹 T：

1. 擷取 S 的所有詞尾：

Suffix[1]= UDDRUDDRDDR$

Suffix[2]= DDRUDDRDDR$

Suffix[3]= DRUDDRDDR$

Suffix[4]= RUDDRDDR$

Suffix[5]= UDDRDDR$

Suffix[6]= DDRDDR$

Suffix[7]= DRDDR$

Suffix[8]= RDDR$

Suffix[9]= DDR$

Suffix[10]= DR$

Suffix[11]= R$

Suffix[12]= $

2. 依字母順序排序所有詞尾：

Suffix[12]= $

Suffix[9]= DDR$

Suffix[6]= DDRDDR$

Suffix[2]= DDRUDDRDDR$

Suffix[10]= DR$

Suffix[7]= DRDDR$

Suffix[3]= DRUDDRDDR$

Suffix[11]= R$

Suffix[8]= RDDR$

Suffix[4]= RUDDRDDR$

Suffix[5]= UDDRDDR$

Suffix[1]= UDDRUDDRDDR$

3. 將排序過的詞尾依順序表示成詞尾樹 T，如圖 5：

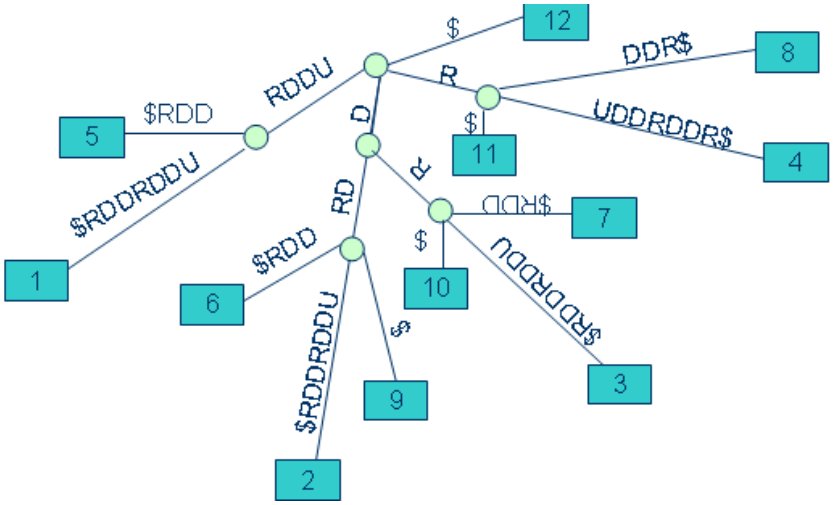


圖 5：詞尾樹 T

由詞尾樹 T 中可觀察到，Suffix[5]與 Suffix[1]中有重複片段 UDDR；Suffix[6]、Suffix[2]、Suffix[9]中也有重複片段 DDR，若回到前面小星星的例子中，字串 S 的重複片段為：

S=
RURURDDRDRDRDURDRDRDURDRDRDDRURURDDRDRDRD$

FP[1]= RURURDDRDRDRD（一閃一閃亮晶晶，滿天都是小星星）

FP[2]= RDRDRD（掛在天空放光明），（好像許多小眼睛）

MIDI Miner 在決定出以旋律輪廓表示的關鍵旋律片段後，還需要參考該片段位於 MIDI 樂譜內的小節（Measure）位置，依該節拍速度（Tempo）值，將音程符號轉換成以 1/1000 秒為單位的節奏數字資料，並以每個片段為一個單位將節奏值存入 Themes 資料庫中，上面兩段重複片段轉換成節奏資料後得到：

FP[1]= 591,592,635,585,588,638,1225,644,599,629,614,608,624

FP[2]= 674,617,635,614,602,589

這些節奏數字資料表示前一個音符開始的時間，到下一個音符開始時間之時間差，將每個片段的節奏向量存入資料庫中當作該歌曲的索引值，在比對時將以這些歌曲之資料節奏，用來和使用者輸入之查詢節奏作比對。

第三節　敲擊式查詢介面

現行敲擊式（Query by tapping）音樂檢索系統皆需透過麥克風〔1〕或 MIDI 電子鼓〔12〕等設備才能讓使用者輸入查詢節奏，檢索系統也多以單機執行的方式設計。爲了改善現行使用者操作介面設計不便之缺點，觀音系統設計之初即希望能讓使用者不透過其他輔助設備，直接就能使用電腦鍵盤輸入節奏，而在查詢介面的設計上也採 Web based 類似 Google 搜尋引擎般的簡潔介面。如圖 6 所示為觀音目前設計之使用者介面，在查詢節奏欄位裡的數字串，是系統紀錄使用者每次按下鍵盤空白鍵的時間差值，所使用的單位和 Themes 資料庫內存放的數值單位一樣，均為 1/1000 秒，圖 6 中的查詢節奏為研究者依照小星星這段童謠在研究者心中的節奏速度所敲出來的（501, 481, 510, 511, 531, 481, 1041, 501, 480, 501, 501, 521, 480）。

圖 6：觀音系統敲擊式查詢介面

第四節　近似比對技術

受限於不同演奏者詮釋音樂的不同方式，所以位於音樂資料庫中的音樂片段，並不一定和查詢者想像的旋律（或節奏）完全相同，再加上查詢者本身對音樂片段之記憶能力，及對音樂的表達能力，非常容易造成音高不準、多個音、少個音等現象，因此近似比對演算法設計的優劣，將決定內容式音樂檢索系統是否能擷取出查詢者期望之音樂片段。

在進行資料比對時，幾乎所有現行的內容式音樂檢索系統，比對的方式都是在計算使用者輸入之查詢向量與資料庫中的資料向量的近似值，有些是利用 DTW 的編輯距離判斷向量的相似程度〔1〕〔2〕〔3〕〔4〕，也有利用 Direct Measure 來比較的〔12〕，但是這些方法在比較向量中的元素值時，只是將他們單純地視為一個個符號，比較的結果也只有吻合和不吻合

兩種結果，針對不吻合的比對沒有程度的區別，這種設計方式較適合用於音高資料比對。但若系統提供利用敲擊式查詢音樂資料的功能，則比對時將會以節奏為主，向量內的元素代表音符和音符之間的時間差，故元素值會用數值型態表示，下述兩個問題是當要利用節奏屬性來查詢音樂資料時必須克服的問題。

1. 數字的比對不適合套用文字符號的比對方式，例如若將〝100〞與〝101〞這兩個元素值視為兩個文字資料之比對，則這個比對會視為不同，但若拿 100 與 101 當作數字型態的資料比對時，則這兩個元素值相似的程度將非常高。
2. 雖然使用者輸入之查詢節奏與資料庫中資料節奏代表同一個音樂片段，但可能因為在音樂詮釋上認知的不同而用不同速度來表示節奏，則在兩個向量中的元素數值會不同，但存在某個比例關係。

在觀音系統中提出一個節奏商標準差方法，用來解決上面所提到的兩個問題，先將查詢向量中的元素一個個的與對應之資料向量元素相除，可得到一組節奏商值，再計算這些節奏商值的標準差，得到的標準差可用來衡量這些商值資料與平均數之間的差異量數。假設一組節奏商值有n個資料分別為x_1、x_2、x_3...x_n 標準差可經由下面公式計算得出：

$$s = \sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}$$

當兩個節奏向量的節奏商標準差值越小，則這兩段節奏越近似。

在比較兩個節奏向量的近似程度時還有一個議題需要被討論：當使用

者輸入之查詢向量元素個數不等於資料向量的元素個數時該如何比對？這有可能是在查詢時，使用者只輸入了音樂重複片段中的部分節奏，或是多輸入了一個音符時間差，少輸入了一個音符時間差等。為了解決這種向量長度不一致無法直接計算節奏商的問題，在觀音系統中使用 n-gram〔24〕方法，來解決上面所提到的問題。

舉例而言，使用 4-gram 的做法，將節奏向量中相鄰 4 個元素互相結合的方式，分別切割查詢節奏向量與資料節奏向量，產生兩組 4 個數字的集合，再依序計算節奏商標準差，並設定一個顆粒離散界限值 δ，當標準差低於此門檻值才視為相同，最後再透過 DICE 係數計算兩個向量的相似程度。假設 P 與 T 分別代表查詢節奏與資料節奏的 4-gram 集合，則相似度可表示成：

$$sim(P,T) = \frac{2*|P \cap T|}{|P|+|T|}$$

相似度=2*(P 與 T 相同之 gram 數)/P 與 T 比較之 gram 總數

下面三個步驟將具體說明觀音系統中使用的近似比對演算法：

1. 首先將查詢節奏 P 與資料節奏 T 依 4-gram 的方式由頭至尾切割為數個顆粒(同 n-gram 之方式)，假設較短之節奏長度為 minL，則 P 和 T 依 4-gram 的方式由頭至尾可切割成(minL-3)個顆粒，每個顆粒中有 4 個元素〔4〕〔5〕。
2. 由頭至尾依序比較 T 與 P 每兩個顆粒之相似度，每兩個顆粒之相似度利用節奏商標準差方法決定。
   (1) 先計算 T 和 P 對應 gram 之 4 個元素值的節奏商值 q1、q2、q3、q4：

   q1 = P(1)/T(1)

q2 = P(2)/T(2)

q3 = P(3)/T(3)

q4 = P(4)/T(4)

(2) 再利用標準差公式，計算這 4 個值的變異離散程度。當節奏商標準差小於顆粒離散界限值 δ ，則此顆粒視為相同。

3. 最後再利用 DICE function 決定整個 P 與 T 的相似度。

實際執行方式依下面演算法：

minL = min(Length(P)，Length(T))　//比較時以較短之節奏字串為主

for (i=1 ; i<=minL-3 ; i++) {

q1 = P(i+0)/T(i+0)　　//分別將 P 和 T 對應之顆粒內的 4 個元素

q2 = P(i+1)/T(i+1)　　//相除得 q1、q2、q3、q4 四個節奏商值

q3 = P(i+2)/T(i+2)

q4 = P(i+3)/T(i+3)

σ = Stddev(q1,q2,q3,q4)　　//計算 q1、q2 、q3 、q4 之標準差值 σ

StdFactor = (q1+q2+q3+q4)/4

//當 q1~q4 之商值愈大，則其 σ 允許在愈大範圍間變化。

//設定顆粒離散界限值為 δ = 0.3*stdFactor，

//若節奏商標準差 σ 小於 δ ，則將此次顆粒比較視為相同。

If　σ < (0.3*StdFactor) {

match++

}

//利用 DICE function 決定整個 P 與 T 的相似度

Similarity = 2*match / [(minL-3) + (minL-3)]

}

下面實際舉三個例子，分別計算這些顆粒之相似程度，目前觀音系統設定之顆粒離散門檻值〝δ = 0.3*StdFactor〞，只要算出來的節奏商標準差值 σ 小於 δ，則這兩個顆粒視為相同：

1. P= 200、210、220、200

T= 220、231、242、220

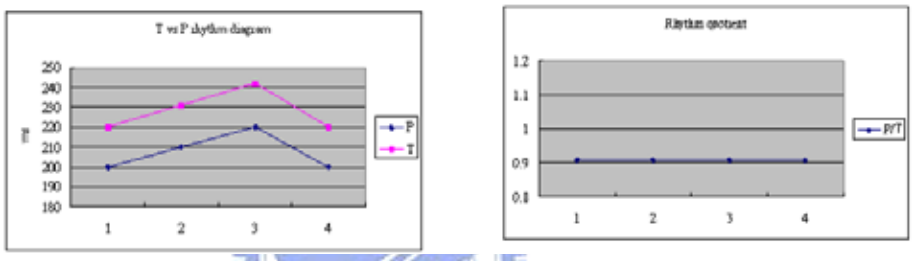


圖 7：顆粒相似程度比較 1

Std factor = (0.91+0.91+0.91+0.91)/4 = 0.91

δ = 0.3*0.91 = 0.273

σ = stdev(0.91,0.91,0.91,0.91) = 0

所以這兩個顆粒可視為相同。

2. P= 435、347、350、429

T= 220、231、242、220

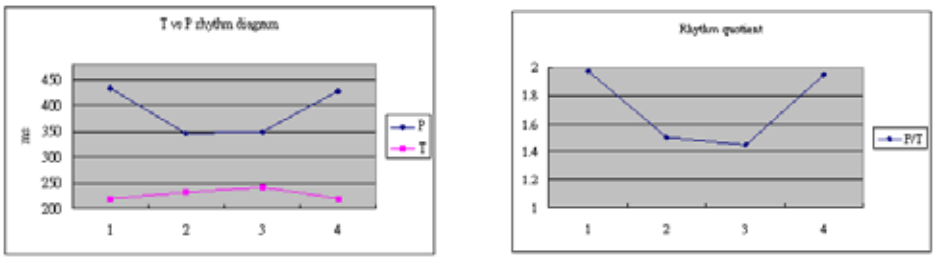


圖 8：顆粒相似程度比較 2

Std factor = (1.98+1.50+1.45+1.95)/4 = 1.72

δ = 0.3*1.72 = 0.516

σ = stdev(1.98,1.50,1.45,1.95) = 0.284

所以這兩個顆粒也可視為相同。

3. P= 250、190、150、300

T= 220、231、242、220

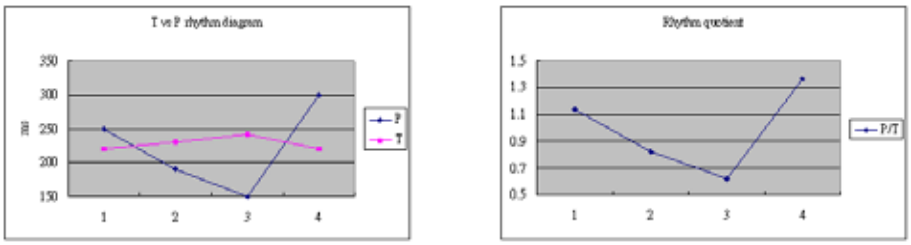


圖 9：顆粒相似程度比較 3

Std factor = (1.14+0.82+0.62+1.36)/4 = 0.99

δ = 0.3*0.99 = 0.3

σ = stdev(1.14,0.82,0.62,1.36) = 0.33

所以這兩個顆粒須視為相異。

在接下來的例子中，要直接利用 4-gram 演算法計算整個向量的相似

度，目前觀音系統設定之〝向量相似度門檻值= 0.5〞，只有當向量相似度大於 50%之節奏向量，系統才會將對應之音樂片段輸出，輸出結果依向量相似度排序，若以下面的兩個向量為例，計算向量相似度的步驟如下：

1. 查詢節奏向量 P=(200，200，600，200，300，600)

   資料節奏向量 T=(100，100，300，100，100，300，200，100，100)

   顆粒離散門檻值 δ = 0.3*StdFactor

2. 去除資料節奏向量後面 3 個元素，讓兩各向量的元素個數相同。

   T = (100，100，300，100，100，300)

3. 切割兩個向量得到的 4-gram 集合如下：

   P_{4_gram} =[(200,200,600,200)，(200,600,200,300)，(600,200,300,600)]

   T_{4_gram} =[(100,100,300,100)，(100,300,100,100)，(300,100,100,300)]

4. 計算節奏商：

   Rhythm quotient =[(2,2,2,2)，(2,2,2,3)，(2,2,3,2)]

5. 計算每個顆粒之顆粒離散門檻值 δ：

   δ = [2*0.3，2.25*0.3，2.25*0.3] = [0.6，0.675，0.675]

6. 計算每個顆粒之節奏商標準差值 σ：

   σ = [0，0.5，0.5]

7. 計算 DICE 相似度係數：

   SIM (P，T) = 2*(3)/(3+3) = 1

   可得到查詢節奏向量 P 與資料節奏向量 T 的相似度達 100%。

這裡提出的節奏商標準差演算法，不僅適合比對數值資料的向量，也不需要像〔1〕般在比較向量前要先進行正規化（Linear Scaling）的動作，因為正規化的動作常會因為向量中某一個異常（Outlier）音符的輸入錯誤，而影響到整組數據的對應關係，舉例來說若以上面例子為例，分別比較查

詢節奏第五個數字輸入 200 與輸入 300：

1. Case 1: 若查詢節奏第五個數字輸入 200

   查詢節奏向量 P=(200，200，600，200，200，600)

   資料節奏向量 T=(100，100，300，100，100，300)

   正規化為 0~1000 的整數後可得到

   查詢節奏向量 P’=(100，100，300，100，100，300)

   資料節奏向量 T’=(100，100，300，100，100，300)

   故經過正規化後為相似度 100%的兩個向量

2. Case 2: 若查詢節奏第五個數字輸入 300

   查詢節奏向量 P=(200，200，600，200，300，600)

   資料節奏向量 T=(100，100，300，100，100，300)

   正規化為 0~1000 的整數後可得到

   查詢節奏向量 P’=(95，95，286，95，143，286)

   資料節奏向量 T’=(100，100，300，100，100，300)

只有輸錯一個音符卻得到完全不同的兩個向量。

第四章 實驗與討論

觀音系統提供了一個直覺的方式，讓使用者很輕易輸入查詢片段，直接查詢音樂內容，雖然系統介面操作非常友善，但是系統之準確度、容錯能力、及反應時間等相關指標，也是非常重要的議題，必須被評估與討論。

傳統上用來評比檢索系統準確度的兩個指標有查全率（Recall Rate）與查準率（Precision Rate），定義如下〔6〕：

$$\text{Precision} = \frac{\text{Number of retrieved references that are relavant}}{\text{Number of references that are retrieved}}$$

$$\text{Recall} = \frac{\text{Number of retrieved references that are relavant}}{\text{Number of relevant references}}$$

由於音樂資料的特性，要判斷音樂資料庫中哪個音樂資料確實和查詢片段有關，或是分析檢索出來的音樂資料中是否和查詢片段有關，這些都非常主觀且困難，故下面另外設計一些實驗，針對不同的近似比對模式比較觀音系統在容錯能力及準確度上的表現。

第一節 實驗資料

在進行以下評估實驗時，觀音系統收集到的 MIDI 音樂約有 120 首，挑選的歌曲都是一般人熟悉之古典音樂，歌曲樂派則包括了巴洛克、古典、浪漫、國民、現代等不同時期之曲目，由其中擷取出的關鍵節奏片段約 200 段並存放於 Themes 資料庫中，而每一段之平均長度約為 18.87 個音符。

在本評估方式中，將採第三章童謠小星星為例，利用相同的查詢節奏向量，比較在不同檢索模式的表現，P 表示所輸入的查詢節奏，T 表示小星星的資料節奏索引片段，圖 10 為小星星之 P vs T 圖。

P=（501,481,510,511,531,481,1041,501,480,501,501,521,480）。

T=（591,592,635,585,588,638,1225,644,599,629,614,608,624）。

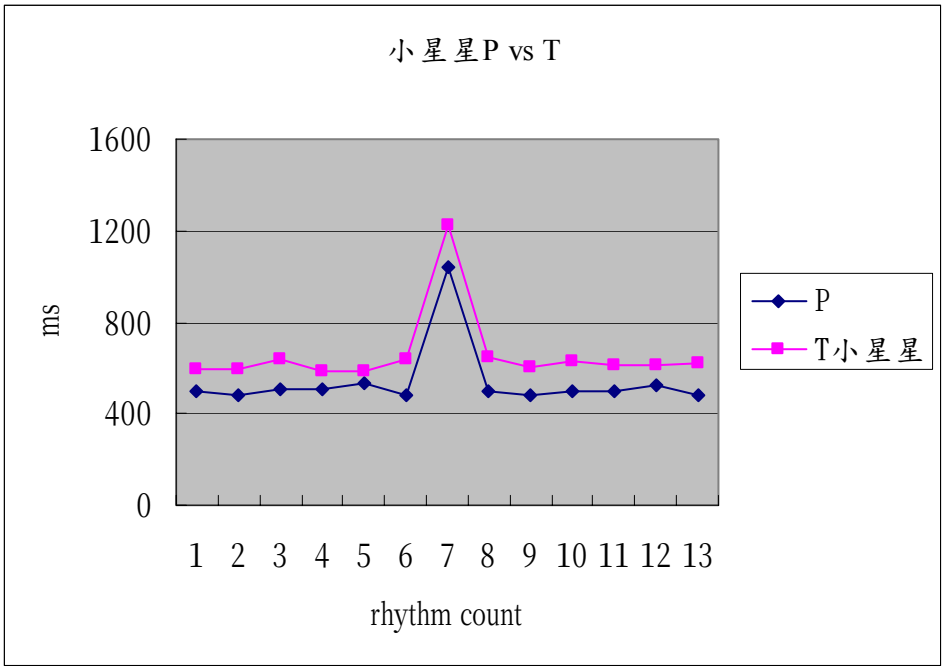


圖 10：小星星 P vs T

第二節　顆粒離散程度評估

直接利用系統預設的顆粒離散程度界限值（δ = 0.3*StdFactor）進行小星星音樂片段檢索，當輸入 P 查詢節奏為(530, 511, 551, 561, 550, 541, 1042, 530, 491, 541, 491, 560, 531,)，系統回應之音樂片段如圖 11 所示：

圖 11：系統回應(顆粒離散程度界限 $\delta = 0.3*StdFactor$)

由系統的回應可以看出，觀音在 0.36 秒內回應了 16 首與查詢節奏相似度大於 50%之歌曲，小星星雖然被找出來了且相似度為 100%，但是因為海頓的驚愕交響曲節奏也極為類似且相似度一樣達 100%，所以小星星被排到第二名，這是因為觀音系統在排序歌曲相似度時，若有多首歌曲相似值一樣，則系統預設將後登錄之歌曲排在前面。圖 12 比較小星星和驚愕的資料節奏片段 T 與查詢節奏 P：

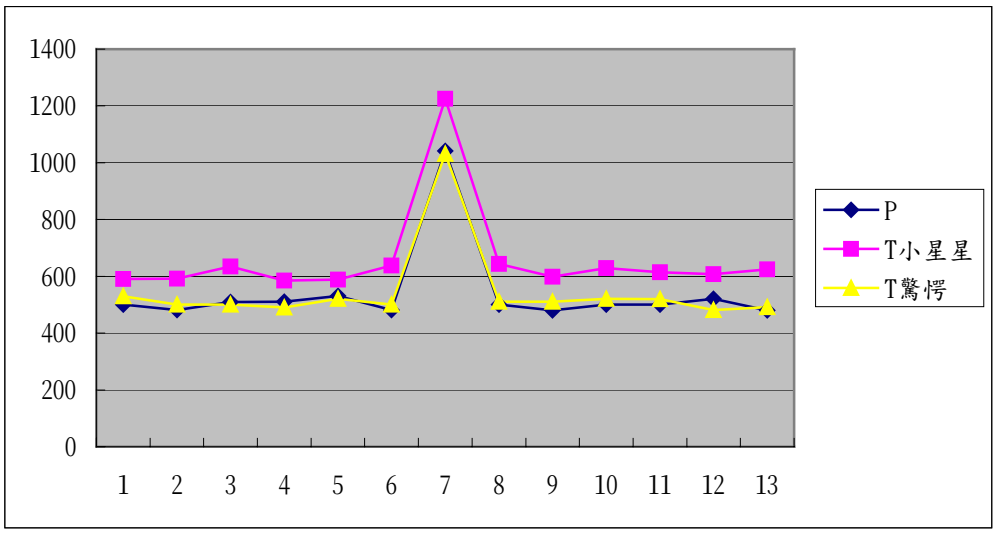


圖 12：小星星 vs 驚愕

由圖 12 可以看出查詢節奏 P 和驚愕交響曲資料節奏 T 更是接近，但是因為系統比較的基準為節奏商標準差，所以很難由絕對座標直接看出 P 比較像小星星或驚愕，對於這種需要很精確地鑑別節奏相似度的情況下，觀音系統也允許調整顆粒離散界限值來達到，圖 13 表示小星星與驚愕在當顆粒離散界限值由 $\delta = 0.30 * StdFactor$ 調整至 $\delta = 0.01*StdFactor$ 時相似度變化的情形：

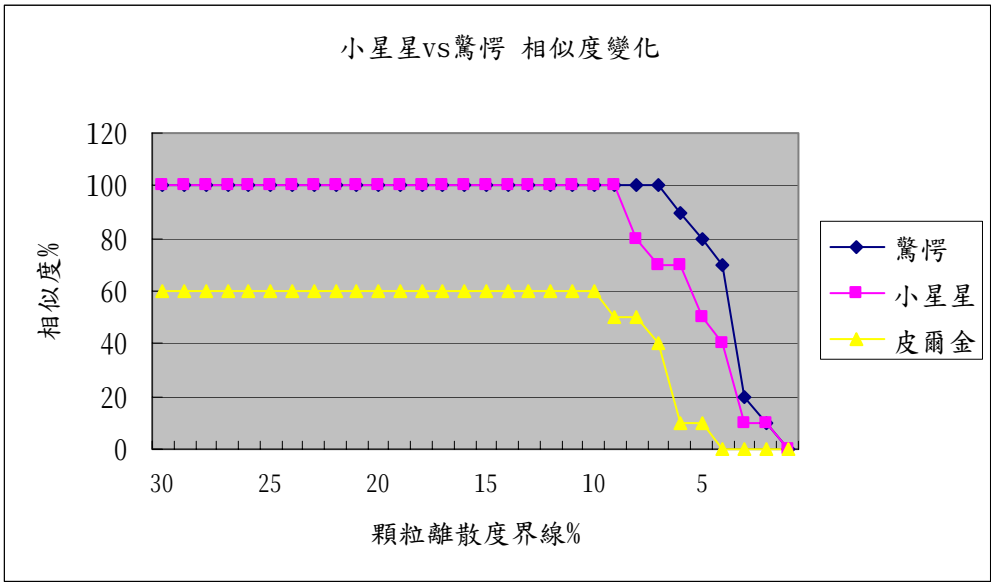


圖 13：小星星 vs 驚愕 相似度變化趨勢

由圖 13 可以看出這兩首歌曲在調整顆粒離散界限值後的變化情形，當 δ 在 30%~9%之間這兩首歌曲都維持 100%與查詢節奏 P 相似，但是到了 8%與 7%小星星的相似度開始往下掉，可是驚愕還是維持在 100%，實驗結果得知查詢片段 P 相似於驚愕更甚於小星星，這種現象會發生在當 Themes 資料庫中不同節奏片段卻擁有極其類似的節奏，故觀音可能會回應出讓使用者〝驚愕〞的查詢結果，這也能讓愛樂者來感受類似的節奏搭配不同旋律在不同歌曲之詮釋方式。

在圖 13 中另外挑了一首和 P 相似程度不高的葛利格皮爾金組曲，由調整顆粒離散界限值可觀察到，它在顆粒離散界限值為 9%的位置處相似度已經由原來的 60%掉到 50%的位置了，所以由這些現象可以歸納出，相似度越不高的歌曲在往下調整顆粒離散界限值時，其相似度會越早開始往下掉。

雖然將顆粒離散界限值調整的很小可得到精準的節奏辨識能力，但是除非使用者有把握輸入的查詢節奏能很接近資料節奏，否則目標音樂片段有可能因為顆粒離散界限值太小，導致整體相似度低於 50%，而被系統由查詢結果中過濾掉。觀音目前預設的顆粒離散界限值為 30% (δ = 0.3*StdFactor)，即是希望犧牲一點準確率，但是能換來更高的查全率。

第三節　顆粒數目評估

觀音系統在檢索時預設會將查詢節奏與資料節奏依 4-gram 的方式，依序切割節奏向量，並比較每個 gram 之相似度。接下來的實驗要分別進行依 2-gram、3-gram、4-gram、至 11-gram 共 10 種切割方式進行檢索，並比較其對查詢結果的影響。

依舊以輸入 P 查詢節奏（530, 511, 551, 561, 550, 541, 1042, 530, 491, 541, 491, 560, 531, ）為測試資料進行檢索，觀察不同的切割方式對小星星、驚愕、皮爾金這三首歌曲與 P 的相似度變化情形，測試結果如圖 14，原本就和查詢節奏 P 相似度極高的小星星與驚愕在不同的 gram 切割方式下，這兩首歌曲始終維持在 100%的與 P 完全相似，但是原來相似程度不高的皮爾金卻有當切割 gram 數增加，其相似度也有慢慢增加的趨勢，故可推論，當 gram 的切割顆粒越大時，會讓相似度的比對越寬，原來相似度較差的音樂都會因調整增加切割 gram 數，而增加其相似度，但是對於鑑別節奏高度類似的片段，無法透過藉由調整切割 gram 數的方式得到改善。

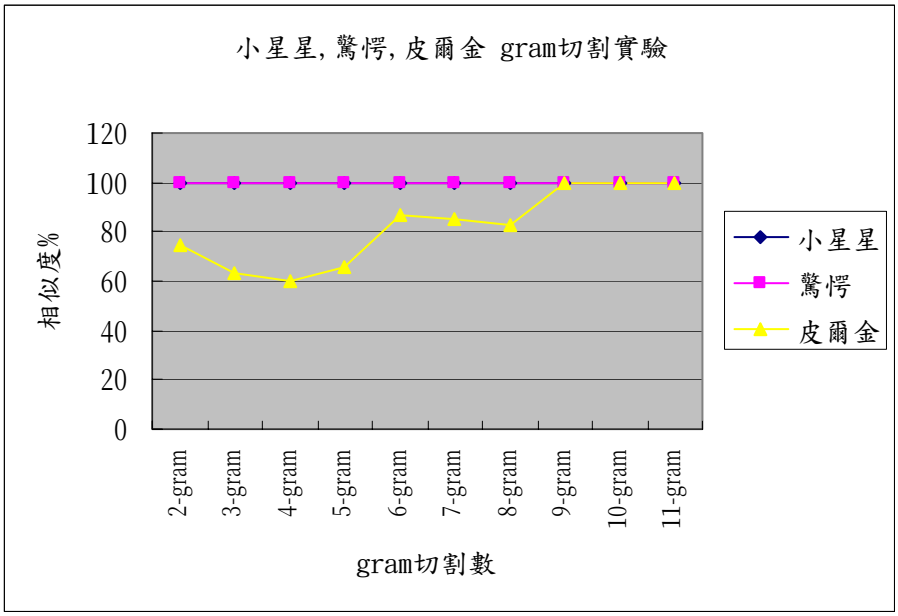


圖 14：小星星 vs 驚愕 vs 皮爾金 gram 切割實驗

第四節　節奏速度不定、時間誤差、多一音、少一音的容錯能力

本節將分別討論利用節奏商標準差演算法來評估，在當使用者輸入之查詢節奏速度不定、時間不準、多輸入一個音、少輸入一個音對相似度的影響。

由於不同人對相同的音樂片段可能會感受到不一樣的節奏速度，以第一節的小星星查詢節奏 P 為例，假設 6 個使用者分別用 P 乘上 0.7、0.8、0.9、1.1、1.2、1.3 的節奏進行查詢，並以 P1、P2、P3、P4、P5、P6 表示，圖 15 為不同查詢節奏 Px 與小星星資料節奏 T 相似度之關係：

P1 = P*0.7 =（351,337,357,358,372,337,729,351,336,351,351,365,336）。

P2 = P*0.8 =（401,385,408,409,425,385,833,401,384,401,401,417,384）。

P3 = P*0.9 =（451,433,459,460,478,433,937,451,432,451,451,469,432）。

P4 = P*1.1 =(551,529,561,562,584,529,1145,551,528,551,551,573,528)。

P5 = P*1.2 =(601,577,612,613,637,577,1249,601,576,601,601,625,576)。

P6 = P*1.3 =(651,625,663,664,690,625,1353,651,624,651,651,677,624)。

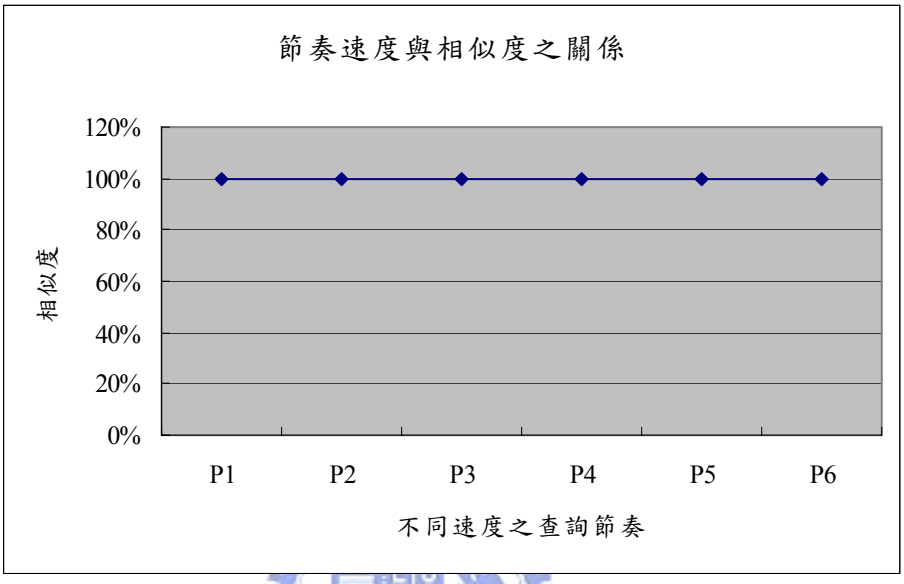


圖 15：節奏速度與相似度之關係

由圖 15 可知，不管是利用 P、P1、P2、P3、P4、P5、P6 任一查詢節奏進行查詢，都可得到完全相同之查詢結果，所以節奏商標準差演算法與時間的絕對長短無關，只要查詢節奏中音符之間的時間差能固定在相同的速度下進行，也就是不要以忽快忽慢的速度輸入查詢節奏，觀音系統均能回應使用者期望之資料節奏片段。

接下來的實驗將探討若輸入查詢節奏中有部分音符之時間間隔不準的問題，一樣以小星星查詢節奏 P 為例：

P=（501,481,510,511,531,481,1041,501,480,501,501,521,480）。

若 P 中的第七個音與第八個音之時間差變化由 400~1800(原始的 P 為 1041) 對小星星資料節奏 T 相似度之影響如圖 16：

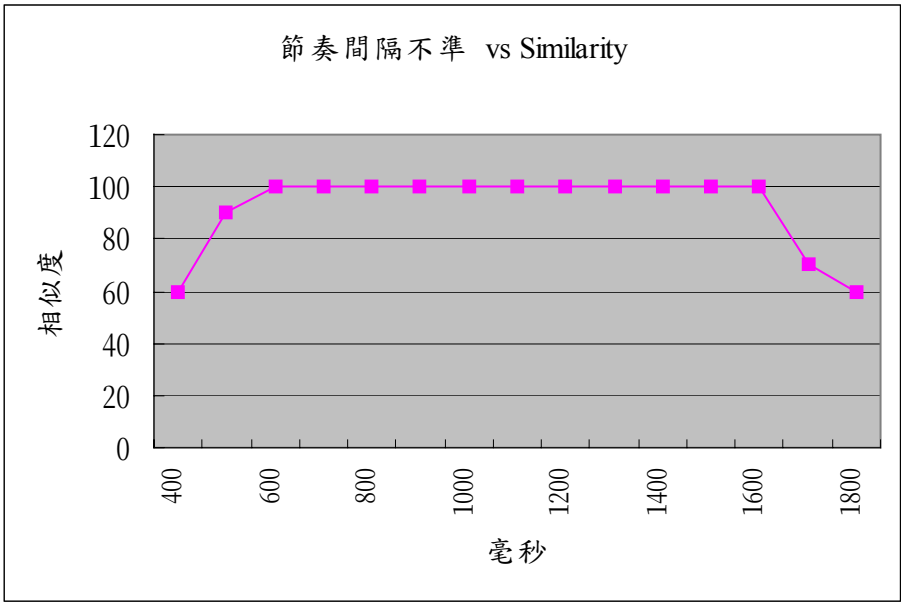


圖 16：P 中時間差變化對整體資料節奏相似度之影響

由圖 16 得知當查詢節奏 P 的第七個音與第八個音在 600~1600 毫秒之範圍區間內變化，近似度都能維持在 100%，故若只有輸入少數錯誤時間差音符，且能將錯誤範圍控制在正負 0.5 倍的範圍內，均能得到合理的相似度值，當然若需要提高系統的準確度，也能藉由調低顆粒離散界限值來達到目的。

接下來討論若在查詢節奏 P 多輸入一個音對近似值的影響，因為觀音系統對多輸入一個音的容錯能力須靠 n-gram 演算法來克服，故多一音在 P 裡面的位置將與近似值有很大的關係，若多輸入一個音的時間差為 200，P1~P14 分別表示 200 出現在 P 的位置，圖 17 表示使用 P1~P14 之查詢節奏對相似度的影響：

P=（501,481,510,511,531,481,1041,501,480,501,501,521,480）。

P1=（200,501,481,510,511,531,481,1041,501,480,501,501,521,480）。

P2=（501,200,481,510,511,531,481,1041,501,480,501,501,521,480）。

P3=（501,481,200,510,511,531,481,1041,501,480,501,501,521,480）。

P4=（501,481,510,200,511,531,481,1041,501,480,501,501,521,480）。

P5=（501,481,510,511,200,531,481,1041,501,480,501,501,521,480）。

P6=（501,481,510,511,531,200,481,1041,501,480,501,501,521,480）。

P7=（501,481,510,511,531,481,200,1041,501,480,501,501,521,480）。

P8=（501,481,510,511,531,481,1041,200,501,480,501,501,521,480）。

P9=（501,481,510,511,531,481,1041,501,200,480,501,501,521,480）。

P10=（501,481,510,511,531,481,1041,501,480,200,501,501,521,480）。

P11=（501,481,510,511,531,481,1041,501,480,501,200,501,521,480）。

P12=（501,481,510,511,531,481,1041,501,480,501,501,200,521,480）。

P13=（501,481,510,511,531,481,1041,501,480,501,501,521,200,480）。

P14=（501,481,510,511,531,481,1041,501,480,501,501,521,480,200）。

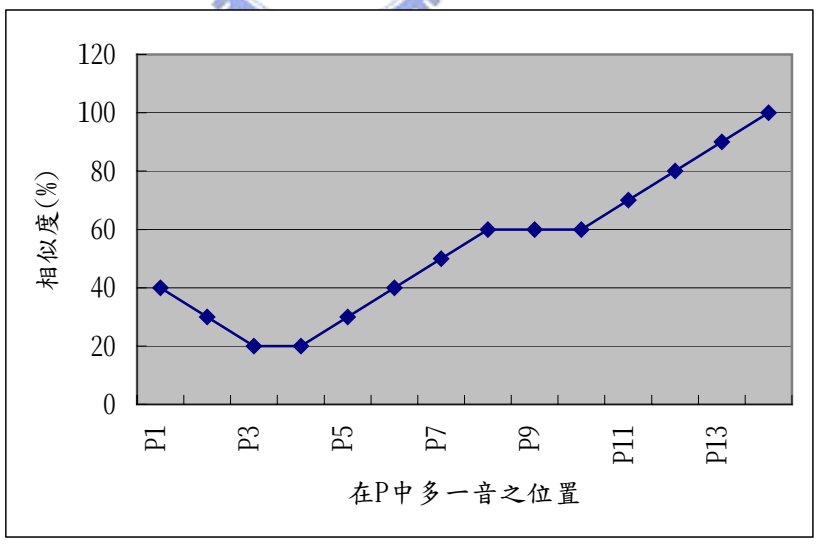


圖 17：在 P 中多一個音對整體資料節奏相似度之影響

繼續討論若在查詢節奏 P 少輸入一個音對近似值的影響，P1~P13 分別

表示在 P 中少一個音的位置，圖 18 表示使用 P1~P13 之查詢節奏對相似度的影響：

P=（501,481,510,511,531,481,1041,501,480,501,501,521,480）。

P1=（481,510,511,531,481,1041,501,480,501,501,521,480）。

P2=（501,510,511,531,481,1041,501,480,501,501,521,480）。

P3=（501,481,511,531,481,1041,501,480,501,501,521,480）。

P4=（501,481,510,531,481,1041,501,480,501,501,521,480）。

P5=（501,481,510,511,481,1041,501,480,501,501,521,480）。

P6=（501,481,510,511,531,1041,501,480,501,501,521,480）。

P7=（501,481,510,511,531,481,501,480,501,501,521,480）。

P8=（501,481,510,511,531,481,1041,480,501,501,521,480）。

P9=（501,481,510,511,531,481,1041,501,501,501,521,480）。

P10=（501,481,510,511,531,481,1041,501,480,501,521,480）。

P11=（501,481,510,511,531,481,1041,501,480,501,521,480）。

P12=（501,481,510,511,531,481,1041,501,480,501,501,480）。

P13=（501,481,510,511,531,481,1041,501,480,501,501,521）。

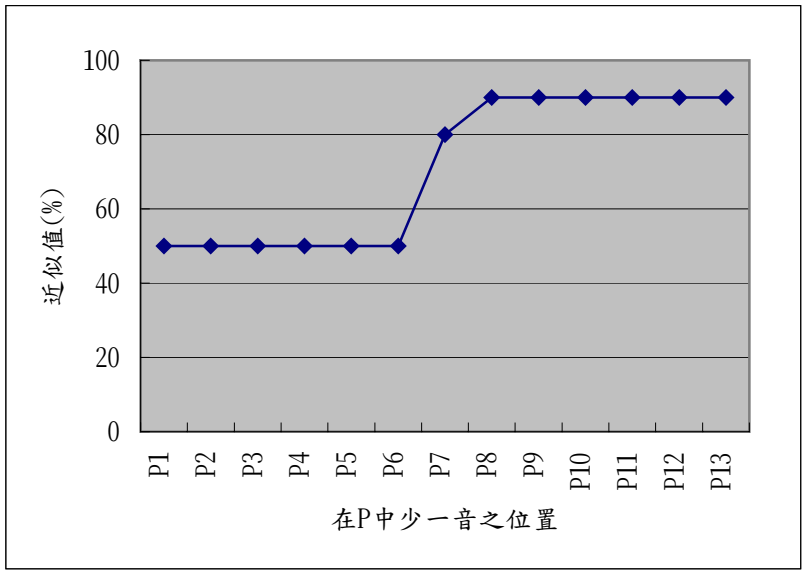


圖 18：在 P 中少一個音對整體資料節奏相似度之影響

由圖 17 與圖 18 可以觀察到，不管是多一個音或少一個音，如果發生之位置在查詢節奏 P 的前面對相似度之影響將大於在 P 的後面，這種現象也恰巧符合一般使用者在輸入查詢節奏時的習慣，節奏準確度的掌握能力往往在開始處能控制的較精確，但是很難一直持續到結束〔1〕。所以若使用者在輸入查詢節奏時發生多一音或少一音的情形，只要發生的位置位於後段，則觀音系統還是能保持不錯的容錯能力，將查詢者要求的音樂片段標示高度的相似值。

第五節　節奏屬性近似比對法的比較

節奏商標準差演算法為觀音系統中首次被提出引用，相較於其他以節奏屬性比對為主的內容式檢索系統〔1〕〔12〕，其近似比對之方式有很大的不同，表 2 比較了這些演算法間不同的特性，假設查詢節奏 P 的長度為 M，資料節奏 T 的長度為 N。

表 2：近似比對演算法比較

系統名稱	超級點歌王	Beat Bank	觀音
如何克服不同速度對節奏的影響	Linear scaling	Linear scaling	節奏商
前置處理	各增加與減少 2 個資料節奏向量維度	將節奏向量的元素以累加之方式表示	利用 4-gram 之方式切割節奏向量
近似比對方式	Dynamic time warping	Direct Measure	節奏商標準差
時間複雜度	O (M*N)	O (M+N)	O (M)

第五章 結論與未來發展

第一節　結論

綜上所述觀音在內容式音樂檢索系統的三個主要問題領域上，使用的解決方式為：

1. 擷取之音樂屬性：節奏。
2. 使用者查詢介面之設計：敲擊式。
3. 近似比對演算法：節奏商標準差。

當使用者無法由查詢結果發現目標音樂片段時，根據實際測試與參考其他音樂檢索系統評估報告〔1〕可以歸納出幾個原因：

1. 一般人在輸入查詢節奏時，一開始拍子及節奏都能掌握得不錯，可是越到後面越容易錯亂掉。由於輸入節奏擁有這種特性，恰好能克服 n-gram 在這裡容錯能力不佳的問題，若使用者在輸入查詢節奏時發生多一音或少一音的情形，只要發生的位置位於後段，則觀音系統還是能保持不錯的容錯能力，將查詢者要求的音樂片段標示高度的相似值。
2. 大部分的 MIDI 檔均不只包括主旋律片段，過多的特殊演奏風格程式將不易分析主旋律，取而代之需透過人工著錄，有人工的部份就易造成人為的疏失。這將是最難克服的一個問題，也就是因為目前由網路上收集之 MIDI 古典音樂結構型式太過複雜，導致自動化工具無法完全自動分析主旋律並著錄，目前觀音系統中大部分的歌曲皆利用手動分析的方式判斷主旋律，如果能將收集的音樂資料定位在較單純之形式，如手機的

鈴聲，相信能有更多的歌曲能透過自動化的方式建立主旋律索引檔。

3. 即使為同一首歌曲，系統對主旋律的認定不一定和查詢者一致。問題出在當使用者查詢的節奏片段恰巧沒被系統拿來當索引值時，也會導致檢索失敗，此時可能系統需要提醒使用者進行查詢擴展（Query Expansion），用另一段節奏進行檢索。

而一個成熟且普及之內容式音樂檢索技術，預期可達到的效益包括：

1. 現行圖書館視聽媒體中心的音樂資料檢索系統多僅提供詮釋資料之查詢，在搜尋音樂資料庫時必須透過文字來進行搜尋，所以檢索條件僅限於歌曲名稱、作曲者、等文字型態資料欄位。本論文提出一種方法，使用者即使只記得部份的歌曲旋律，也可以針對音樂資料庫進行搜尋，讓這種直接輸入〝音樂內容〞進行檢索的方式變得可能。
2. 作曲者可能隨時產生之創作靈感，要確認其在創作中之節奏是否曾有人發表過類似的作品，可以事先以音樂內容查詢音樂資料庫，避免發表後產生的智財糾紛；此外對於音樂領域的研究學者，內容式音樂檢索系統也提供一種方法讓他們能查詢一些節奏近似的作品，進行研究分析。
3. 待內容式音樂檢索相關技術發展成熟，不僅對廣大的音樂愛好者提供更直覺的檢索方式，同時也會刺激音樂資訊提供者加速現有音樂資料的數位化，進而能在網路上發展出更多元化的服務項目，使得一般大眾能夠利用更容易的方法獲得個人喜好之音樂商品。
4. 目前世界各先進國均有數位圖書館相關計畫在推行，各種不同媒體資料數位化之研究更是其核心工作，由於音樂資料本身具有的獨特性質，目前仍有許多問題與挑戰需要克服。本論文所提的數位音樂內容檢索技術是目前仍在發展中的重要主題，亦是數位化圖書館所需的技術。透過本論文系統的發展，除了增加使用者利用音樂資訊的便利性外，進而希望

提升音樂教育及創作的社會環境，讓更多人能感受到音樂藝術裡的喜悅。

第二節　未來發展

在本論文中展示了一個敲鍵式音樂檢索系統，並提供直覺的輸入介面讓使用者使用，及發展合適之近似比對方式，在未來還有一些研究方向可以努力，以加強觀音系統在音樂檢索上的能力：

〝節奏〞這個音樂屬性是否可能做為檢索條件，用來檢索音樂資料？這也是發展觀音系統成功與否的關鍵，所以觀音系統在選擇音樂資料庫中的音樂資料時，即以樂風節奏拍子要求規則且鮮明的古典音樂為主，這樣將能讓使用者有充分的把握能夠利用心中的節奏片段，檢索到資料庫內相似的節奏片段。雖然同一首曲子可能在不同樂團或演奏者的詮釋下有不同的感受，但是基本上不同版本之詮釋方式其音樂節奏還是維持在差異性不大且可識別的狀態下發展。然而爵士樂的演奏方式則不適合用節奏來檢索，因為在爵士樂中有許多即興的片段，提供演奏當時樂手自由發揮表現的空間，這也導致相同的曲子卻有差異很大的節奏表現，所以針對這種樂風的音樂勢必要設計另外一種新的索引策略。

節奏商標準差演算法也有如其他動態演算法的缺點，就是在做節奏的近似比對時只限於從頭開始比對，亦即若使用者輸入的查詢節奏不是由資料節奏的前面開始，而是由資料節奏的中間片段開始，則在計算此二節奏之相似度時將無法得到滿意的結果，雖然在切割節奏 gram 後，要依次地將查詢節奏由資料節奏的每一個 gram 開始處比對，再分別計算相似度也是可

行，但是隨著資料節奏的長度可能很長，這種比對方式將耗費驚人的時間，要克服這個問題可能需要再設計一種新的比對方式。

本論文所應用的 n-gram 近似比對計算方式，對多一個音或少一個音的容錯能力，也無法如預期般的產生效果，若是增減音符的現象發生在後面幾個顆粒則對整體相似值的影響程度較小，若是從查詢節奏的前面開始就多一個或少一個音，則整體相似度就會被嚴重影響（如第四章第四節所述），所以要克服這種前半段容錯能力較差的現象，需要再設計一種合適之顆粒切割方式。

參考文獻

〔1〕 JS Roger Jang, Hong-Ru Lee, Chia-Hui Yeh: *Query by Tapping: A New Paradigm for Content-based Music Retrieval from Acoustic Input*: The Second IEEE　Pacific-Rim Conference on Multimedia, Beijing, China, 2001

〔2〕 A. Ghias, J. Logan, D. Chamberlain, B. C. Smith: *Query by humming-musical information retrieval in an audio database*: ACM Multimedia ’95 San Francisco, 1995.

〔3〕 Rodger J. McNab, Lloyd A. Smith, David Bainbridge, Ian H. Witten: *The New Zealand Digital Library MELody inDex*: D-Lib Magazine, May, 1997

〔4〕 Y.H. Tseng: “*Conetent-Based Retrieval for Music Collections*,” In Proc. Of ACM SIGIR”99, Berkley, CA, USA, Pages 176-182, 1999.

〔5〕 曾元顯: *音樂內容查詢不匹配問題與檢索模式之研究*: 資訊傳播與圖書館學, 第 6 卷, 第 4 期, 2000 年 6 月

〔6〕 Ian H. Witten, Alistair Moffat, Timothy C. Bell: *Managing Gigabytes: Compressing and Indexing Documents and Images*: MK Morgan Kaufmann, 1999.

〔7〕 *http://www.midi.org*

〔8〕 Michael Mitzenmacher, Sean Owen : *Estimating Resemblance of MIDI Documents*: , Third International Workshop on Algorithm Engineering and Experimentation , p79–90 , 2001.

〔9〕 Alexandra. L. Uitdenbogerd, Justin Zobel: *Matching Techniques for large music databases* : In Proc. of ACM Multimedia, Pages 57-66, 1999.

〔10〕Arbee L.P Chen, M.Chang, J.Chen, J.L Hsu, C.H. Hsu and Spot Y.S Hua: “*Query by music Segment: An Efficient Approach for song Retrieval* “: In Proc. Of IEEE Int”l Conf. on Multimedia and Expro, 2000.

〔11〕Alexandra. L. Uitdenbogerd, Justin Zobel: *Manipulation of Music for Melody Matching*: Proc. ACM International Multimedia conference, Bristol, UK, 1998.

〔12〕Gunnar Eisenberg, Jan-Mark Batke, Thomas Sikora: *BeatBank- An MPEG-7 compliant Query by Tapping System*: Audio Engineering Society, Convention Paper 6163, 2004.

〔13〕M. Hawley, “*The personal Orchestra,* “Computing system, Vol. 3, No. 2, PP.289~329, 1990.

〔14〕J.R.J.G Proakis, J.H.L Hansen: *Discrete-time processing of speech signals*: New York, Macmillan Pub. Co., 1993.

〔15〕R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R van Oostrum: *Using transportation distances for measuring melodic similarity*: In Prooceedings of the 4th International Conference on Music Information Retrieval. ISMIR, 2003.

〔16〕V.I. Levenshitein : *Binary codes capable of correction deletions, insertions and reversals*: Soviet Physics Doklady, 6:707-710, 1966.

〔17〕*http://www.greenstone.org*

〔18〕ISO/IEC JTC 1/SC 29: *Information Technology – Multimedia Content Description Interface*: ISO/IEC FDIS 15938, 2002.

〔19〕R. Frances: *La Perception de la Musique*: Translate by W. J. Dowling, 1958.

〔20〕M.T. Chen and J. Seiferas: *"Efficient and Elegant Subword-Tree Construction"* : In A. Apostolico and Z. Galil. Editors, Combinatorial Algorithms on Words, Vol. 12 of NATO ASI Series F, Computer and System Sciences, Springer-Verlag, Berlin, Germany, Pages 97-107, 1984.

〔21〕E. McCreight: *"A Space-Economical Suffix Tree Construction Algorithm"* : Journal of Association for Computing Machinery, Pages 262-272, 1976.

〔22〕羅有隆、游合成: *以線性記憶體空間發現音樂資料中非不重要重覆片段之技術*. 朝陽科技大學資訊管理學系碩士論文, 2002

〔23〕Dan Gusfield: *Algorithm on String, Trees, and Sequence - Computer Science and Computational Biology*: CAMBRIDGE University Press,1997.

〔24〕Salton, G: *Automatic text processing. The Transformation, Analysis and Retrieval of Information by Computer*: Reading, MA: Addison Wesley, 1989.

〔25〕William Birmingham, Bryan Pardo, Colin Meek, Jonah Shifrin, Dept. of Electrical Engineering and Computer Science, The University of Michugan : *The MusArt Music-Retrieval System* : D-Lib Magazine, February 2002 Volume 8 Number 2.

〔26〕Jeff Glatt : *MIDI file Disassembler* *(http://www.borg.com/~jglatt/progs/software.htm)* : 1998.

〔27〕V. Bakhmutova, V.D. Gusev, T.N. Titkova: *The search for adaptations in song melodies*: Computer Music Journal, vol. 21, no.1, page58~67, Spring 1997.