行政院國家科學委員會專題研究計畫　成果報告

音樂資料之分析、塑模、與擷取研究(2/2)

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-004-012-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立政治大學資訊科學系

計畫主持人：陳良弼

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 94 年 10 月 31 日

中文摘要

隨著網路的普及與數位壓縮技術的進步，音樂物件得以以電子格式廣泛地流傳，因此人們對於相關的音樂服務需求也就日益迫切，重要的音樂服務包括了音樂資料的查詢、音樂資料的分析等，其目的都在於節省人力跟時間的花費。在本計畫中，我們分別就兩類重要的課題深入研究，包括了內涵式音樂查詢與音樂分析。在內涵式音樂查詢方面，我們考慮在音樂串流的環境之下，如何處理連續性的音樂片段查詢，並提出一能夠滿足使用者即時需求的處理技術。在音樂分析方面包括了自動化音樂曲式分析與擷取音樂物件中的近似重複樣型：在自動化音樂曲式分析方面，我們根據音樂理論設計出兩個不同的方法來辨認一個音樂物件是否為輪旋曲或是賦格曲，並進一步分析其採用的作曲手法；在擷取近似重複樣型的方面，配合 R*-Tree 的索引架構以及篩選之技術，我們提出了一個有效率的方法來達成此目標。音樂分析的結果將可以用於音樂架構之建立，用於 MusiXML 的音樂模型之下，將可以提供更高階的音樂查詢服務。我們進行了一連串的實驗來驗證我們所發展的技術，其效率與正確性都優於過去的技術。

關鍵字

音樂服務、音樂串流、內涵式音樂查詢、連續性查詢、音樂曲式分析、近似重複樣型擷取、音樂資料模型、索引架構、查詢處理、篩選技術、多媒體

Abstract

With the popularity of network application and the advance of digital compression techniques, music objects in various formats are widespread on the Internet. The demand on critical music services, such as music retrieval and music analysis, are stronger than ever for saving browsing or processing times. In this project, we focus on two kinds of important issues, i.e., content-based music retrieval and music analysis. For content-based music retrieval, we deal with the continuous music segment retrieval under music streaming environment and develop a real-time technology to answer the users' queries on time. For music data analysis, we consider the automatic musical form analysis and approximate repeating pattern extraction. On automatic music pattern analysis, we design two different methods to recognize whether a music object is Fugue or Rondo, and further analysis the compositional rules adopting in the music object. On approximate repeating pattern retrieval, we bring out an effective method based on R*-tree index structure and filtering techniques. The results of music analysis can be used to build up music structure. Combined with the music model of MusiXML, we can provide the state-of-art music query service. Finally, a series of experiments are performed to demonstrate that the proposed methods and system have better effectiveness and efficiency than previous works.

Keywords

Musical Service, Music Streams, Content-Based Music Retrieval, Continuous Query, Musical Form Analysis, Approximate Repeating Pattern Extraction, Music Data Model, Index Structure, Query Processing, Pruning Technique, Multimedia

內容目錄

附圖目錄

附表目錄

一、前言

隨著網路的普及與便利，使得大量的音樂資料能夠以電子格式廣泛地流傳，在此趨勢之下，人們對於相關的音樂服務需求也就日益迫切，重要的音樂服務包括了音樂資料的擷取，以及音樂資料的自動化分析等，其目的都在於節省使用者原本所需耗費的大量時間與精力，因此，如何提供新的技術來支援這些音樂服務便是一重要課題。在本計畫中，我們除了探討如何在即時性的音樂播放環境下，有效率地尋找包含使用者所感興趣之音樂片段的歌曲外，同時也開發了兩種與自動化音樂分析相關之技術，其一是音樂曲式之分析，其二則是重複樣型之擷取（repeating pattern），而這些技術的開發將有助於提升音樂服務的效率與品質。

二、研究目的

在本計畫中分別就三項與音樂服務有關之重要課題加以研究與討論，分別是在串流環境下的內涵式音樂查詢(content-based music retrieval)、音樂曲式之分析(musical form analysis) 以及重複樣型之擷取（approximate repeating pattern extraction），欲達成之目的分述如下：

（一）串流環境下之內涵式音樂查詢

由於數位化音樂資料透過網路大量地散佈，使用者得以在短時間之內取得大量的音樂資料，因此，為了能夠有效率地從音樂資料庫中找到使用者所感興趣的歌曲，過去有許多的研究都著重於讓使用者提供一感興趣之音樂片段，然後再配合一索引架構（index structure），執行所對應的查詢處理方法（query processing），以期能夠快速地找到包含使用者所感興趣之音樂片段的音樂物件，然而這些內涵式的音樂查詢方式只能適用於鮮於變動的大型音樂資料庫，一旦牽涉到大量的音樂資料變動，便必須要花費大量的時間來進行索引架構的新增、移除或是更新，進而導致整體查詢效率的嚴重降低，同時這類的音樂查詢研究僅能支援一次性查詢（one-time query），而無法在音樂資料變動的環境下，持續通知使用者滿足其需求之音樂物件。

網路電台所播放之音樂頻道便是一個音樂資料經常性變動的實例，透過網路的傳播，音樂頻道會持續性地播放不同的音樂物件，這意味著每一個音樂頻道所播放之音樂資料可以被視為一條音樂資料串流，在這樣的前提之下，過去的研究便無法完全地支援，因為在音樂串流的環境之下，使用者所需要的將是能夠持續監視（monitor）每個音樂頻道所播放之音樂物件，進而動態性地通知使用者目前播放的音樂物件是否包含了使用者所感興趣之音樂片段，這樣的查詢需求不同於過去的一次性查詢，而是一種能夠持續回報結果的連續性查詢（continuous query），同時由於音樂資料不斷地流入，對音樂資料建立索引的方式也變的不可行，我們便需要一全新的技術在此串流環境下來滿足使用者連續性查詢的需求；以提供音樂查詢服務系統的角度來看，大量註冊的連續性查詢是不可避免的問題，因此我們也需要考慮到如何能夠同時服務大量同時存在的連續性查詢，以期能夠避免查詢效率的低落。考慮到這些新的環境與需求，我們的目標便是在於能夠發展一新的內涵式音樂查詢方法，能夠讓使用者能夠即時地獲得通知，得知是否某一音樂頻道正在播放其所感興趣之音樂物件。

（二）音樂曲式之分析

過去對於音樂物件的研究，多半著重於如何在龐大的音樂資料庫中進行內涵式的音樂查詢，然而一個音樂物件所包含的不只是音樂的內容(content)而已，其架構(structure)

也會反映該音樂物件所要表達的情感與內涵，以作曲者的角度來看，一個音樂物件的內容與架構是相輔相成的，因此對於音樂架構的分析也是十分重要的研究課題。透過自動化音樂架構的分析，我們可以提供音樂學習的機會，讓音樂領域的學生得以瞭解並學習一首音樂物件所採用的架構與編曲手法，此外，同時考慮音樂內容以及音樂架構將可以讓音樂分類的準確性得以提升。

然而，過去對於音樂架構的研究卻是十分侷限的，例如重複樣型的擷取或是音樂分段（music segmentation）等技術，都未完整考慮到音樂理論，所找出來的重複樣型或是音樂分段可能不滿足音樂理論的規範，在這種情況下便無法提供有意義的音樂架構分析。為此，一個能夠配合音樂理論來分析音樂物件曲式的技術便是需要發展的，透過音樂曲式的分析，便可以完整表達一個音樂物件所採用的音樂架構，以及其所對應的編曲手法，瞭解了一個音樂物件所對應的曲式之後，我們便可以清楚地區分音樂內容中重要的片段以及次要的片段，對於音樂主題（theme）的擷取也會有很大的助益。

由於音樂曲式是十分複雜且多變的，在本計畫中，我們針對了兩種最重要也最常見的曲式提出了對應的分析方法，這兩種曲式分別是輪旋曲（Rondo）以及賦格曲（Fugue），以此為基礎，我們將能夠發展對其他曲式之分析方法。

（三）重複樣型之擷取

對於內函式的音樂檢索和音樂風格分析，基本需求要從音樂著作的原始資料提取音樂特性。

音樂的一個重要特性是結構特點，以古典音樂為例，他們大多數根據一個基本架構而形成音樂形式。此基本架構為重複樣型，也就是不斷重複出現在音樂各個地方的旋律片段，這些重複的音型被定義為音樂的原型旋律，通常這是分析樂曲的基礎。原型旋律必然是常常出現在樂曲的很多地方且不是完全相同的樣型，而當近似重複樣型被找出後，通常可用其來代表該歌曲的特徵，而後可延伸作為歌曲分類或是音樂檢索的依據。

傳統上找尋重複樣型的方法通常是計算兩兩音樂片段的差異，如果差異在容許範圍內則認為彼此是相似的，以任一片段為核心，統計相似的片段個數，如果多餘門檻值則認為該核心片段為一個可能的重複樣型，但這種作法往往會花費太久的時間，尤其當使用者找尋重複樣型時，不斷調整可能的長度或相似度的門檻時，這種作法就非常不切實際，因此，我們需定義一個旋律相似度的量測法，並設計一個能有效找出近似重複樣型的方法。此技術將能應用於自動化曲式分析方法之上，進一步提升音樂分析的準確性。

三、文獻討論

（一）內涵式音樂查詢

因為目前的音樂物件多是以複音的格式存在，也就是在音樂物件播放時，同一個時間點會有一個以上的音符同時發聲，此外，大部分的使用者或是音樂查詢系統均是以音高為比對的標準，因此在考慮複音音樂的查詢時，我們可以視為是在進行一多數值字串的比對工作。對於複音音樂之查詢，Lemstrom[Lems00]根據 shift-or 演算法發展一新演算法，但是與 shift-or 演算法相同，其查詢樣型的長度受限於計算機中的字元長度，而且其查詢只限於單音的形式；為了提供複音查詢的能力，Clausen 等人在[Clau00]中提出以集合的方式儲存樂曲中音符出現的時間與該音符的音高，形成類似總譜表（score-like）的格式，依此來處理音樂查詢，但是這種技術只有在提供精準答案時具有效率；此外，Dovey[Dove99]則是提出一個以 Dynamic Programming 為基礎之演算法，希望能夠在音樂查詢時提供近似

答案，然而由於該技術的限制，在音樂資料庫過大或是音樂字串過長時，會因為計算量過大而導致效率明顯降低。

由於內涵式音樂查詢可以轉換成多數值字串的比對工作，相關的研究還包括了許多不同的字串比對法，其中，KMP 演算法[Jaga00]與 Boyer-Moor 演算法[Boye77]是兩個廣為人知的精確字串比對（exact string matching）演算法。然而，要比對一個查詢字串，所有儲存在資料庫當中的字串必須要一一的擷取出來與查詢字串作比較，此等比對過程效率明顯不佳。字尾樹（suffix tree）[Mccr76][Wei76]是一個針對部分字串比對問題所提出來的索引結構，資料庫當中所有字串的字尾（suffix）都會被紀錄在一個樹狀結構之中。要處理一個使用者的查詢只需要比對樹狀結構中的一條路徑（path）即可，並不需要將資料庫中所有的字串擷取出來比對，因此，查詢處理的效率也就大幅度的增加。然而，建立字尾樹索引的過程是十分費時的。此外，字尾樹索引所佔的記憶空間也較龐大。1D-List 結構[Liu99]是針對音樂資料建立索引的一個鏈結串列結構。要處理一個查詢字串，只需要將與查詢字串相關的串列資料擷取出來就可以了，不需要將整個完整的索引結構載入記憶體之中，因此，記憶空間的需求也相對較少。除此之外，作者也開發了一個有效率的演算法來進行精確以及近似的字串比對。除了精確字串比對之外，由於多媒體資料的特性，近似比對也就成為了一個重要的考量。在過去的研究之中[Baez92][Corm93][Kahv01][Nava01][Wu92]，通常以兩個字串之間的編輯距離（editing distance）作為兩個字串之間的相似程度，因此在我們的方法中，也採用了編輯距離來作為兩兩音樂片段間相似度衡量的依據。

在本計畫中，我們所考慮的環境乃是音樂串流環境，因此所考慮的課題便不同於上述用於解決傳統資料庫之下的技術，前述的那些索引架構與比對技術無法直接套用於資料串流的環境之下，因此我們必須要開發出一套新的技術，以便在串流式的環境之下能夠有效率地找到使用者所感興趣的答案。

（二）音樂分段技術

為了達到音樂分析的目的，過去的研究首先探討的課題便是對於音樂物件的分段技術，透過音樂分段的技術，我們便可以擷取出一個音樂物件所包含的樂句或是樂段，樂句和樂段都是音樂結構中的單位，對於音樂曲式的分析有著很重要的關連性。[Frib98]和[Yana99]考慮休止符或是長音的音長來進行音樂分段的工作，然而這樣的作法並沒有完整考慮到音樂物件的其他特徵，因此分段的準確性並不高。[Cambl]則是介紹了一個 Local Boundary Detection Model 來計算在一個音樂物件中連續音符間音高、音長以及休止符間的變化程度，當這些變化的加權總和高於所設定的門檻值時，就認為是一個合適的切割點，然而如何選擇合適的權重便是這個方法最為困難的部分，因為不同的音樂物件可能有各自合適的權重選擇方法。而[Chen04]結合上述方法的優點，並且結合了 melodic shape 的觀念[Huro95]來進行樂節的擷取工作，更進一步地利用樂節分群以及資料探勘[Pei01]的技術，將所擷取出的樂節連接成有意義的樂句。

（三）重複樣型之擷取

如何從音樂中找出所有的重複樣型曾於 [Gusf97]中被討論，其作法是利用 Suffix tree 的方法來作為其索引架構。這類的方法在處理前會先建立一棵 Suffix tree，其中每條到葉結點的路徑表示一種重複樣型，當走過索引樹後，所有的重複樣型將能被找出。但這類的方法所找出的樣型必需是完全一樣的，也就是樣型間不容許有任何一點的差異，然

而這樣的限制並不能滿足實際應用的需求。在[Hus01]中，定義出所謂 Non-trivial 的重複樣型，此重複樣型的特性在於不會被其他較長的重複樣型所包含，為了找出此種重複樣型，[Hus01]提出了兩種不同的演算法並加以比較，其中一個是利用 correlative-matrix，另一個則為 String-join 的方法。但與前面的問題相同，這兩個方法還是只能找出完全相同的樣型。Shih 等人[Shih01]曾提出一個演算法來找出音樂中的重複樣型，他們先將音樂切割成小節為單位，然後利用編碼的技巧將音樂重新編碼，以加快萃取的速度。Pienimaki[Pien02]曾經考慮過近似樣型的搜尋，但其方法只能處理移調的問題，而且根據實驗，該演算法相當費時。Rolland[Roll01]提出近似重複樣型的定義，並設計出利用 Dynamic programming 特性的演算法來找出音樂資料庫中所有的近似重複樣型。但此演算法的時間複雜度相當高。因此，我們考慮到音樂分析上實際應用的需求，尤其是樂曲中重複的音樂片段通常都會加入一些裝飾音來加以變化，一個能夠有效率地找出音樂物件中的近似重複樣型近似重複樣型技術便是一值得研究的課題。

四、研究方法

（一）音樂串流查詢技術

在音樂頻道播放的音樂都是以複音音樂為主，所謂複音音樂即是在單一時間內會有一個以上的音符同時發音，因此在處理音樂串流時，我們必須先提出一複音音樂的表示法，使得我們能夠表示描述出一條音樂串流，同時也能夠描述使用者所下的查詢，以利於查詢比對的進行，因此我們定義了事件串流（event stream）來描述音樂串流，並用事件序列（event sequence）來描述使用者查詢，所謂的事件（event）可以包含一個以上的物件（item），例如<63,65,67>這一個事件即是代表了音高為 63、65 與 67 的三個音符同時發聲，透過這種方式，我們便可以描述在音樂串流上流過的複音音樂，以及所用者所查詢的複音音樂片段。

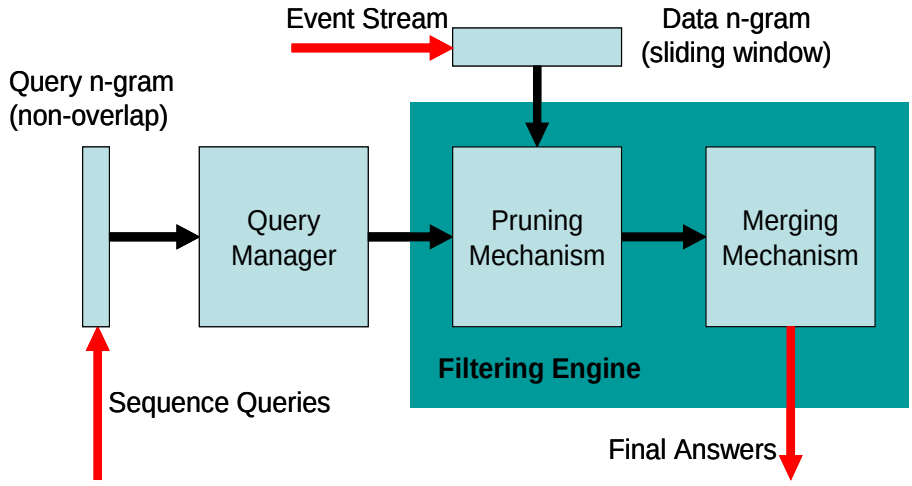
在此串流環境之下，使用者透過其註冊的查詢片段，希望系統能夠即時通知是否某一音樂頻道目前正在播放包含其感興趣之音樂片段之音樂物件，因此其查詢便可以視為是一種連續性查詢，而且由於使用者並不能下達完全精確的查詢，或者是音樂串流在傳輸的過程中會有資料誤差的情況發生，因此我們的方法必須要讓每一個查詢能夠獲得近似答案而非只有精確答案，如此一來才能夠符合實際應用的需求。在此方法中，我們採用edit distance來衡量答案與查詢間的距離，只有當某一音樂物件的片段與查詢的距離低於查詢所要求的誤差值時，該音樂片段才會成為查詢的答案之一，所對應的音樂物件才會被回報給使用者，三種不同的edit operator以及對應的花費如下式所列，其中a_i與b_i代表兩個不同的event：

$$\begin{aligned} &\text{Deletion: } cost(a_i,\lambda) = 1 \\ &\text{Insertion: } cost(\lambda,b_j) = 1 \\ &\text{Replacement: } cost(a_i,b_j) = 1 - SIM(a_i, b_j) \end{aligned} \quad (1)$$

兩個 event 間的相似度計算方式則是採用 *Jaccard coefficient*，其算式如下：

$$SIM(a_i,b_j) = \frac{|a_i \cap b_j|}{|a_i \cup b_j|} \quad (2)$$

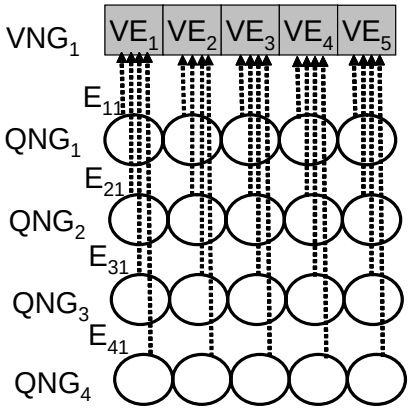
在我們的方法中，針對上述問題提出一完整之方法來加以解決，其流程如圖一所示，一共包含了三個部分：Query Manager、Pruning Mechanism 與 Merging Mechanism，以下我們將逐項介紹這些部分的作法：



圖一：查詢方法流程圖

（I）Query Manager

考慮到同時處理大量的查詢，Query Manager 的目的在於為這些查詢建立所對應的索引架構，以加速查詢比對的速度，由於查詢間的長度可能會有所不同，為了能夠利用查詢間的共通性，我們先將每一個查詢切成數個 n-gram，然後透過分群機制（clustering mechanism），相似的 query n-gram（QNG）將會被分在同一群中，為了加速後續的查詢比對工作，每一個 query n-gram 分群都會依據其所包含的 query n-gram，計算出一個對應的 virtual n-gram（VNG）來作為該分群的摘要（summarization），在 virtual n-gram 的每一個 virtual event（VE）即為該分群的 query n-gram 對應位置的 event 的聯集，同時我們會記錄構成每個 virtual n-gram 的那些 event 所包含的物件數的最大值與最小值，virtual n-gram 形成的示意圖如下所示：



圖二：Virtual n-gram 形成示意圖

（II）Pruning Mechanism

當音樂資料串流持續不斷地留入組成某一個音樂物件的事件時，我們利用一長度為 n 的 sliding window，不斷地擷取出最新的 data n-gram 來進行比對，方法的基本原理在於，如果我們能為某一個查詢的 query n-gram 找到一個對應的 data n-gram，則這些對應的 data n-gram 所組成的音樂片段便有機會成為滿足該查詢的答案，當然，能滿足某一 query n-gram 的 data n-gram，這個 data n-gram 與該 query n-gram 的距離絕不會大於查詢所要求的誤差值。因此，為了避免每一個流入的 data n-gram 必須跟所有的存在的 query n-gram 做距離的計算，當一 data n-gram 流入時，我們將此 data n-gram 與每個 query n-gram 分群所對應之 virtual n-gram 做距離的運算，我們提出了一個方式能夠計算出一個 data n-gram 與一個 virtual

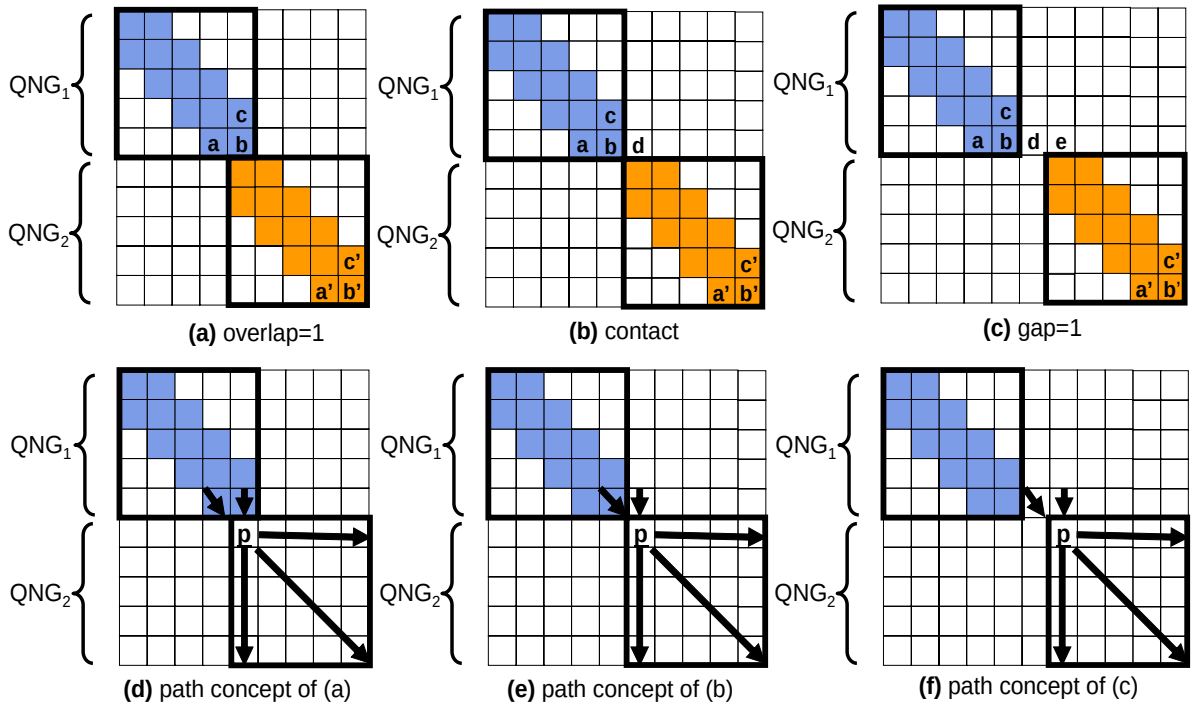
n-gram 所可能存在的最短距離，利用這個方式，如果某一個 virtual n-gram 與一 data n-gram 的距離大於查詢所容忍之誤差值，則我們可以很快地濾掉該 virtual n-gram 所對應之分群下的所有 query n-gram，反之，該 data n-gram 很可能是分群中某一 query n-gram 能夠滿足的答案，我們就會進行後續的確認動作。

為了能夠計算出一 data n-gram 與一 virtual n-gram 所可能存在的最短距離，我們提出了下列的公式(3)來取代公式(1)中的 replacement cost，如此只要利用 edit distance 原先的計算方法，我們便能夠預估出此一可能的最短距離，在公式(3)中 DE 代表 data n-gram 中的一個事件，而 VE 則是代表了 virtual n-gram 中的一個事件，我們同時也證明了此一公式的正確性，並確保我們所預估出的最短距離一定會是實際存在的最短距離的 lower bound。

$$cost(DE,VE_i)=\begin{cases} 1-\dfrac{|DE\cap VE_i|}{|DE|}, if\ MIN_i \le |DE\cap VE_i| \le MAX_i \\ 1-\dfrac{MAX_i}{|DE|}, if\ |DE\cap VE_i| > MAX_i \\ 1-\dfrac{|DE\cap VE_i|}{|DE|+MIN_i-|DE\cap VE_i|}, if\ |DE\cap VE_i| < MIN_i \end{cases} \quad (3)$$

（III）Merging Mechanism

由於我們將查詢切成數個 n-gram 來加以處理，因此在執行的過程之中，我們會獲得很多滿足某一 query n-gram 的部分解答，由於構成某一查詢的 query n-gram 彼此之間是有順序性的，而且是缺一不可的，所以對某一個查詢來說，當我們得到了其第 k 個 query n-gram 所對應的部分解答 A 時，我們應該已經要獲得前 k-1 個 query n-gram 所對應的部分解答，否則 A 這個部分解答便無價值而可以直接拋棄，即使前面的 k-1 個部分解答已經存在，後續的檢查機制也會啟動，預估合併之後的結果是否有可能形成未來的答案，檢查分為三個部分，第一個是檢查合併之後的部分答案是否過長或是過短，過長或是過短都會使得合併之後的部分答案與查詢間的誤差大於所容許的範圍，第二個部分則是評估部分解答 A 與其對應之 query n-gram 所可能存在的最短距離，如果這個距離超過了查詢的容錯範圍，則合併動作就會被終止，第三個部分則是檢查合併之後的部分解答與查詢間可能存在的最短距離，如果此一距離已經超出了查詢所容許的誤差值，則該合併動作就不會被實際執行，當兩個部分解答要進行合併時，只有三種可能的情形會發生，分別是 overlap、contact 與 gap，其情形如圖三所示，我們可以透過評估距離的方法來決定這樣的合併是否有執行的價值。每個部分解答都有其生命週期，如果有一個部分解答 B 是由某一個查詢前 k-1 個部分解答連結而成，但是在超過誤差值的 gap 後還是無法連結第 k 個 query n-gram 的部分解答時，B 就會被淘汰。我們方法的特點便是在於用評估距離的方式來減少測量實際距離所必須花費的昂貴代價。



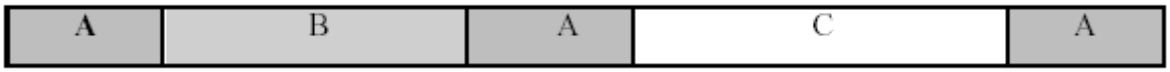
圖三：三種部分解答合併情況之示意圖

對一個查詢來說，當我們可以合併每一個 query n-gram 所對應的部分解答時，這個由部分解答所組合出的片段便有機會成為該查詢的真正答案，在此階段，我們便會實際計算出該音樂片段與查詢的實際距離是否在容許的誤差範圍之內，以便獲得最後的答案。

（二）音樂曲式分析技術

根據音樂理論以及作曲法則，我們所設計的方法可以辨認一首歌曲的曲式是否為輪旋曲或是賦格曲，此外，如果是的話，我們也會一併擷取出其主旋律並且找出其使用的對位技巧。

輪旋曲的特點是一個主題會不斷地出現，而且在兩個主題間會穿插不同的副題（couplet），因而可以創造出連綿不絕的流暢風格，圖四為一可能的輪旋曲架構，其中 A 為主題，B、C、D 則為穿插之副題，需要注意的是輪旋曲的結尾一定會回到主題之上。



圖四：輪旋曲架構圖

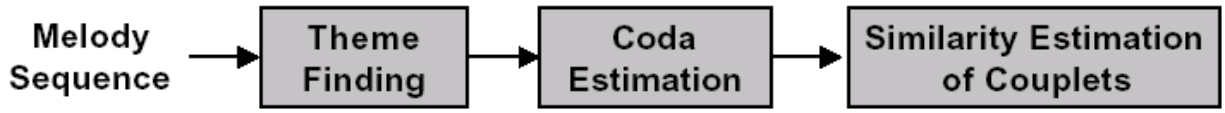
賦格曲則是一種多聲部的樂曲，在賦格曲中，會有一個主題穿插出現在樂曲中，同時在樂曲中會使用大量的模仿與對位技巧，圖五為一個四聲部的賦格曲架構，第一聲部會先進入主題，而後第二聲部會在主題結束時進行模仿，而第一聲部則會進入陳述主題完之後的對位主題，這樣的模式會直到第四聲部進入主題之後結束。

圖五：四聲部賦格曲架構圖

我們提出了兩個不同的流程來檢測一首音樂物件是否為輪旋曲以及是否為賦格曲，我

們將分別介紹這兩個流程：

圖六為輪旋曲曲式分析的方法流程，首先我們會先找出音樂物件的主旋律（melody sequence），接著套用我們所發展的重複樣型擷取技術，在主旋律中找出所包含的重複樣型，其中出現次數為三次且其長度最長者即為主題，如果我們找不到滿足的重複樣型，則此音樂物件就不屬於輪旋曲。



圖六：輪旋曲曲式分析方法流程

由於輪旋曲在結尾時，可能會會在主題之後加入一段尾聲（coda）以求樂曲的完整及圓融性，因此我們在找出主題之後，我們必須檢查在最後一個主題出現後是否有出現另外的音樂片段，這個音樂片段必須要滿足尾聲的樂理要求，否則此一音樂物件便不滿足輪旋曲曲式的要求。根據音樂理論，雖然尾聲與主題的旋律可能有所不同，但是兩者的音調卻必須要一致，因此如果樂曲中最後的主題之後還有一段音樂片段，我們便利用下列的算式來檢查該音樂片段是否為合法的尾聲。

$$\text{Sim(Coda, Theme)} = |\text{Key(Coda)} - \text{Key(Theme)}| \quad (4)$$

$$\mathbf{Key(M_i)} = \sum_{i=1}^{N} NPitch(i) * Duration(i) / \sum_{i=1}^{N} Duration(i) \quad (5)$$

Where $NPitch(i) = (Pitch(i) \bmod 12) + 1$

$Pitch(i)$ *is the pitch (MIDI note number) of note* i

$Duration(i)$ *is the duration of note* i

N *is the length of the theme or Coda*

最後，在輪旋曲中，主題間會穿插不同的副題，而結尾不包含在副題之中，這些副題彼此之間並不相同，也就是副題並不會重複地出現在輪旋曲中，我們採用 longest common subsequence 的觀念來檢查兩兩副題之間的相似度，如果任兩個副題的相似度大於設定的門檻值，則此音樂物件也不滿足輪旋曲式的要求，在檢查兩兩副題的相似度時，我們會以音樂輪廓（contour）的表示法來描述這些副題，其計算公式如下所列：

$$\text{Contour\_Sim}(CM_i, CM_j) = |LCS(c_i, c_j)| / MAX(|c_i|, |c_j|) \quad (6)$$

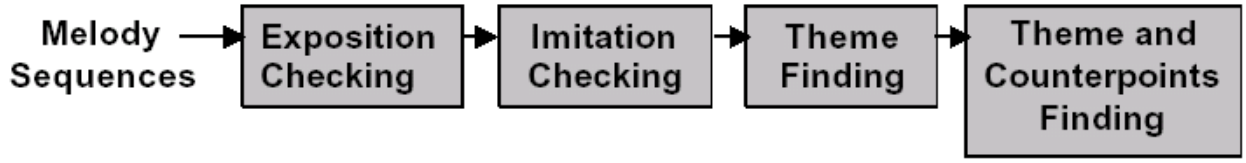
Where c_i *is the contour of couplet* CM_i *and* c_j *is the contour of couplet* CM_j

$|LCS(c_i, c_j)|$ *is the length of the longest common subsequence of the contours* c_i *and* c_j

$MAX(|c_i|, |c_j|)$ *is the maximum length of the contours* c_i *and* c_j

能夠通過上述三種檢查的音樂物件其曲式則為輪旋曲。

圖七為賦格曲曲式分析的方法流程，首先我們必須先將音樂物件分成不同的聲部並分別擷取每個聲部的主旋律，其後，依據賦格曲的曲式特性，檢查是否不同的聲部是一個接著一個進入的，這件工作便是檢查每一個聲部的第一個起始音是否有錯開，一旦有兩個以上的聲部的起始音在相同的時間點，則此音樂物件不屬於賦格曲；接著，我們檢查第一聲部的主題結束後，第二聲部是否有緊接著出現主題，但是由於我們還不確定主題，所以我們可以先檢查第一聲部的起始音跟第二聲部的起始音是否屬於相差五個半音或是七個半音，如果不是的話，則此音樂物件不屬於賦格曲。



圖七：賦格曲曲式分析方法流程

在輪旋曲中，主題會先在第一聲部開始，主題結束的附近，第二聲部也會開始進入，而第三聲部在進入時也會隨即開展一樣的主題，依據這個觀察，我們就可以將第一聲部的開頭片段擷取出來，此開頭片段的結束為第二聲部發聲的地方，這個片段就可以視為是主題，然後再比對是否與第三聲部的開頭部分吻合，如果不吻合的話，則此音樂物件不屬於賦格曲。由於我們將主題擷取出來了，其他的音樂部分就可以被切割出來，根據樂理，我們就可以觀察這些部分是採用哪些對位技巧，其中包括了 augmentation、diminution 以及 melodic inversion 等。

（三）近似重複樣型擷取技術

首先我們利用 edit distance 作為相似度量測的依據，並明確的定義所謂近似重複樣型，其中樣型的最大及最小長度可由使用者設定以減少不必要的音樂片斷被當作重複樣型輸出，例如過短的旋律。另外使用者可設定最少出現次數的門檻(min_sup)來過濾出現次數太少而不具代表性的樣型。

定義一：編輯距離(Edit distance)

Three types of *edit operations* that transform segment P (denoted as $p_1 \ldots p_m$) into segment Q (denoted as $q_1 \ldots q_n$) are insertion, deletion and replacement. The edit distance between segments P and Q denoted as $edit(P,Q)$, is the minimum number of edit operations required to transform P into Q.

定義二：距離門檻(Distance threshold)

A *distance threshold* for a pivot P is $\delta_P = |P| * \gamma$, where $|P|$ is the segment length of the pivot and γ is the *distance threshold ratio*, $0 \le \gamma < 1$.

定義三：相似片段(Similar segment)

Given two segments P and Q, satisfying *max_len* and *min_len*, Q is a *similar segment* of P if $edit(P,Q) \le \delta_P$.

定義四：重疊程度(Overlapping degree)

Given two similar segments $S[a:b]$ and $S[c:d]$ where $a \le c \le b$, the overlapping degree of them is $(b-c+1)/min(b-a+1, d-c+1)$ if $b<d$. Otherwise it equals 1.

定義五：重疊門檻(Overlapping threshold)

An *overlapping threshold* for two similar segments I and J of a pivot is $O_{IJ} = min(|I|,|J|) * \rho$, where $|I|$ and $|J|$ are the segment lengths and ρ is the *overlapping threshold ratio*, $0 \le \rho \le 1$.

定義六：擴展(Extension)

Given a pivot P and the set of all its similar segments **S**, an *extension* of P (denoted as *Ext(P)*) is a subset of **S**, where every two segments in it satisfy the overlapping threshold. The number of segments in an extension is called the *support* and denoted as $|Ext(P)|$.

定義七：近似重複樣型(Approximate Repeating Pattern)

A pivot P is called an *approximate repeating pattern* (abbreviated as ARP) if there exists at least one $Ext(P)$ satisfying the support threshold, i.e., $|Ext(P)| \ge min\_sup$.

傳統的作法是將所有可能長度的樣型產生出來後，在兩兩計算彼此間的距離，去除過遠的樣型關係後，統計出現的次數並與最少出現次數門檻後，將符合條件的樣型輸出。

由於使用 dynamic programming 的方法計算兩兩音樂片段間的相似度是很花時間的，因此，我們的目標在於減少不必要的計算來加速整個找尋的處理。首先，我們會將所有的音樂片段轉換並對應至向量空間的點，利用一新設計的距離計算方法來評估真正的距離，由於此方法的計算複雜度較原有的方法低，且能保證是低於真正的距離，因此可以利用它來刪除不用計算的點，每個音樂片段在向量空間中的表示法如下列定義。

定義八：長條圖向量(Histogram vector)

Let D be a string with $\Sigma_D = \{a_1, a_2, \ldots a_n\}$, S be a segment of D, and h_k^S be the count of a_k in S. The *histogram vector* (abbreviated as *Hvector*) is defined as follows:

$$HV(S) = <h_1^S, h_2^S, \ldots, h_n^S>$$

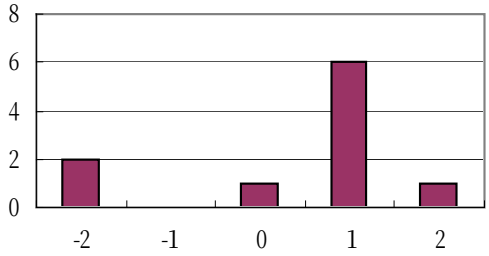
String: D=(0,1,1,-2,0,1,1,-2,2,1,1,-1)

Σ_D = {-2,-1,0,1,2}

S=(1,1,-2,0,1,1,-2,2,1,1)

$HV(S)$=<2,0,1,6,1>

$|HV(S)|$=10

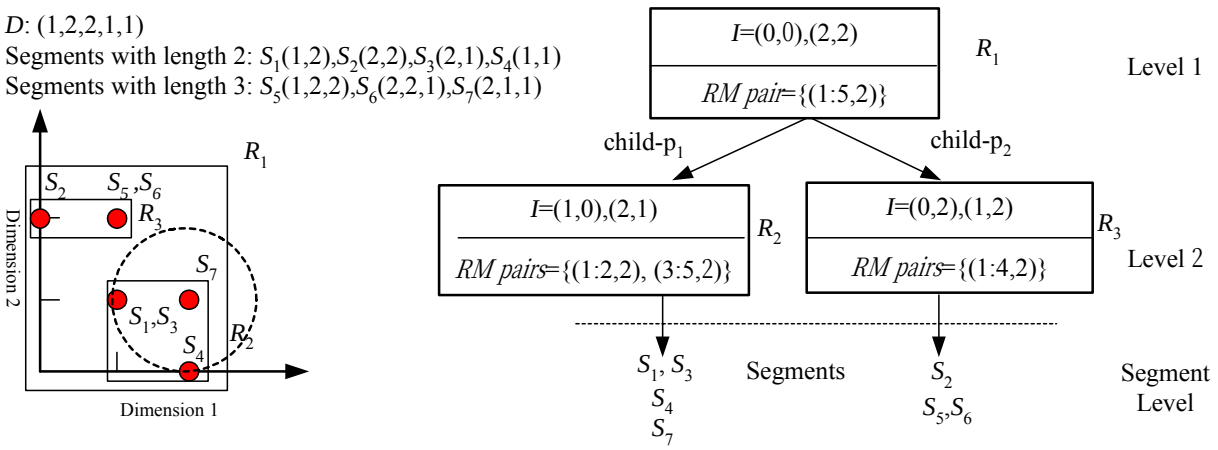


圖八：音樂片段轉換至對應向量空間之範例

如圖八，一個音樂片段D可統計每個音高出現的次數，轉換該次數成為一個向量HV(S)。利用長條圖的差異，我們可以知道其差異的量會比實際的編輯距離要短，所以我們設計的向量空間兩點的距離公式如下:

$$ins(HV(S_1), HV(S_2)) = \sum_{i=1}^{|\Sigma_D|} d_i, where \qquad d_i = \begin{cases} h_i^{S_2} - h_i^{S_1} & if \quad h_i^{S_2} > h_i^{S_1} \\ o & otherwise \end{cases} \tag{7}$$

為了讓整個過程能再加速，我們對向量空間的點建立R*-tree索引，並在索引節點中加入一些參數來再減少比較的次數。此修改後的R*-tree稱為parametric R*-tree，範例如下圖:



圖九：利用R*-tree架構查詢近似重複樣型之圖例

在 R*-tree 中的節點中，我們使用 RM pair 資料結構來記錄在此 MBR 所包含的連續子字串資料的起點與終點，另外還記錄此子字串最短的片段長度，利用這兩個資料我們可以評估出在此 MBR 中所包含合乎重疊條件的片段個數。

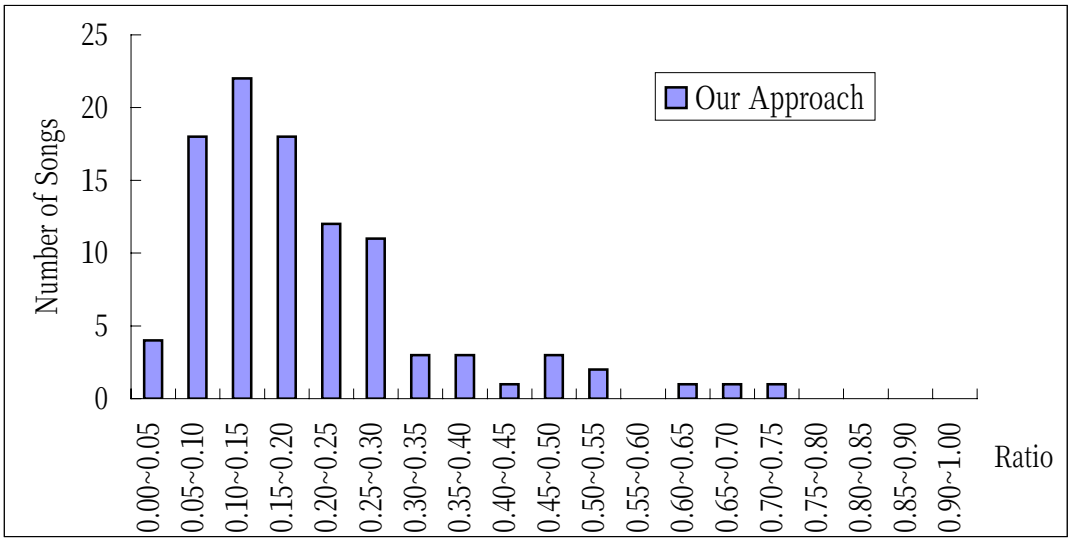
在整個搜尋過程我們是採用 range query，以每個可能成為重複樣型的片段為中心，並以距離的最大容許值為半徑，逐層找尋與此圓疊覆的 MBR。這些 MBR 所包含片段可能是他的近似片段，然後再依據這些 MBR 中的 RM pairs 的資料估算可能會通過重疊條件的片段最大值，並與數量門檻值比較，如果高於門檻值則進一步往 R*-tree 的下一層進行 range query，如果已經到達底層時，則計算這些片段彼此間的向量空間距離，去除超過距離門檻的數量後，再一次統計該評估的數量，如果再一次通過，才真正計算他們的 edit distance，然後統計低於距離門檻的音樂片段的數量，比較是否高於數量門檻，如果通過則稱此搜尋的核心音樂片段為近似重複樣型。

五、結果與討論

（一）音樂串流查詢技術實驗成果

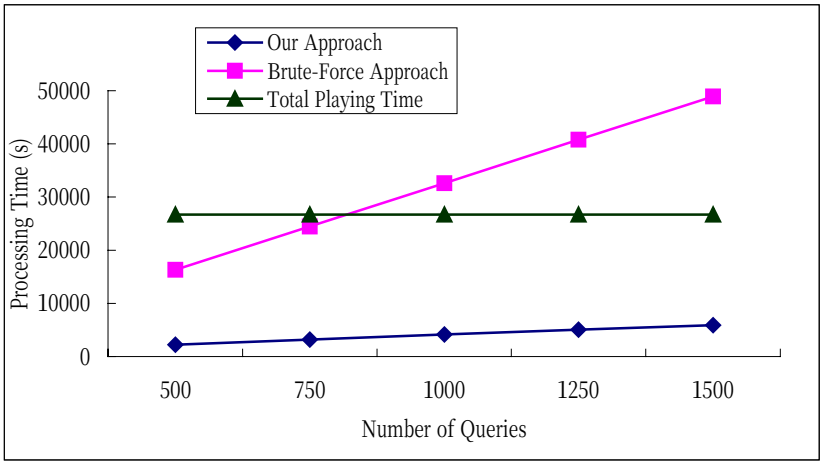
為了驗證我們所發展的音樂串流查詢技術，我們設計了一個系統，讓使用者能夠在系統上註冊其所感興趣之音樂片段，系統會持續監控音樂頻道，當音樂頻道上所播放的音樂包含使用者所查詢之音樂片段時，系統則會即時回報此結果給使用者，此外，每一個使用者註冊的音樂片段都只考慮其音高的部分，因為這是最常被使用也是最重要的音樂特徵。音樂資料格式的轉換並不是我們著重的地方，因此我們採用模擬的方式，預先選定了一百首 MIDI 格式的音樂，由一虛擬的電台亂數播放這 100 首歌曲，亦即是系統所監控之音樂頻道。為了能夠反映出我們所發展之方法的效率，我們設計了一個暴力法，這個暴力法是採取 linear scan 的方式在音樂頻道播放的音樂物件中找尋能夠滿足某一個註冊的查詢的答案。

第一個實驗便是測試我們的技術能夠滿足即時性的需求（real-time requirement），這意味著當答案出現時，亦即是當使用者所感興趣之音樂物件出現時，我們應該在其播放結束前通知使用者，否則這樣的通知就會失去了意義。首先，我們在系統中註冊了 1000 個不同的音樂片段，這些音樂片段都是從原本的 100 首歌曲中亂數取出的，我們觀察平均處理一首歌曲要花多少時間，其結果如圖十所示，我們可以發現，當系統為 1000 個查詢檢查是否一個音樂物件有滿足它們的答案時，平均處理的時間僅為該音樂物件播放時間的 0.163 倍，例如一首歌曲播放的長度為 3 分鐘，則我們只需要花費 29 秒來檢查其中是否有滿足這 1000 個查詢的答案。如果跟暴力法比較，暴力法平均處理的時間為一個音樂物件播放時間的 1.22 倍，也就是有超過 60%的歌曲播放完畢時，還沒有辦法檢查完這些歌曲是否有滿足查詢的答案。暴力法效率低落的原因在於查詢間的計算處理並沒有共享的機制，而且也沒有好的刪除技術來快速地排除那些不可能的答案，而這些正是我們所發展之技術所擁有的。當我們將查詢的數目增加至 1500 個時，平均處理的時間僅為該音樂物件播放時間的 0.225 倍，而暴力法則增加至 1.825 倍。



圖十：歌曲播放時間與查詢處理時間之比較

當查詢數目增加時，所需要的查詢處理時間也會隨之增加，因此在第二個實驗中，我們藉著增加查詢的數目來觀察查詢處理時間的變化，並同時與暴力法的處理時間作一比較，其結果如圖十一所示，當查詢數目從 500 增加至 1500 時，我們的方法所需要的處理時間總和仍然低於 100 首歌曲的總播放時間，然而當查詢超過 750 個時，暴力法所需要的處理時間總和已經開始超過歌曲的總播放時間，因此我們的方法比較具有 scalability 的特性的。



圖十一：總播放時間與查詢處理時間之比較圖

最後，選定一個合適長度的 n-gram 對系統的效能是有決定性的影響的，因此我們在實驗中探討 n 與使用者查詢的誤差容許值ε的關係，以及 n 與查詢長度的關係。為了觀察 n 與ε的關係，我們將 n 設為 6，然後改變ε的值，則當系統處理 1000 個查詢時，其結果如表一所示，如果把ε等於 1 時的總處理時間當作基準，則ε等於 2 時，總處理時間會變成ε等於 1 時的 1.07 倍，由表一我們可以觀察到，如果 n 跟ε過於接近時，查詢處理的效能便會降低，其原因便是如果 n 跟ε過於接近，則 pruning mechanism 的效能就會降低，因為大部分的 data n-gram 都可通過其測試而達不到事先過濾的效果。

表一：n 與誤差容許值ε的關係

Error Bound (ε)	1	2	3	4	5
Ratio	1	1.07	1.20	1.32	1.58

為了觀察 n 與查詢長度的關係，我們將 1000 個查詢的長度全部固定為 24，然後改變 n

的值，其結果如表二所示，如果把 n 等於 6 時的總處理時間當作基準，則 n=8 時，總處理時間會變成 n 等於 6 的 2.58 倍，由表二我們可以觀察到，如果 n 太過接近查詢的長度的話，查詢處理的效能便會降低，其原因便是如果 n 太接近查詢長度，query n-gram 分群的數目會越來越接近原本查詢的數目，這將會使得我們無法減少計算的數量而達不到計算共享的目標。

表二：n 與查詢長度的關係

Parameter (n)	6	8	10
Ratio	1	2.58	4.49

經由上述實驗的呈現，我們可以發現這項技術在處理音樂串流的查詢上，可以達到很好的效率，足以應付實際應用上的需求。

（二）自動化音樂曲式分析實驗成果

為了驗證我們所發展的自動化音樂曲式分析技術，我們收集了實際的歌曲並且由專家來實際分析這些實驗歌曲的曲式以及其音樂架構，這些由專家分析的結果便是我們方法所要達成的正確解答。

首先，我們收集了 30 首知名的輪旋曲及其他非輪旋曲式之音樂物件，以自動化的方式來分析其曲式與架構，並與專家的結果作一比較，結果如表三所示，我們的方法可以達到百分之百的正確率，表示我們的方法不但可以成功辨認一個音樂物件的曲式是否為輪旋曲，同時我們也可以成功地擷取其主題及副題的部分。

接著，我們收集了 48 首知名的輪旋曲及其他非輪旋曲式之音樂物件，以自動化的方式來分析其曲式與架構，並與專家的結果作一比較，結果顯示我們的方法可以百分之百分辨出一個音樂物件的曲式是否為輪旋曲式，而對輪旋曲而言，主題擷取的正確率可以達到 95.64%，而每一個模仿主題的部分所採取的編曲手法我們也可以百分之百的辨認出來。

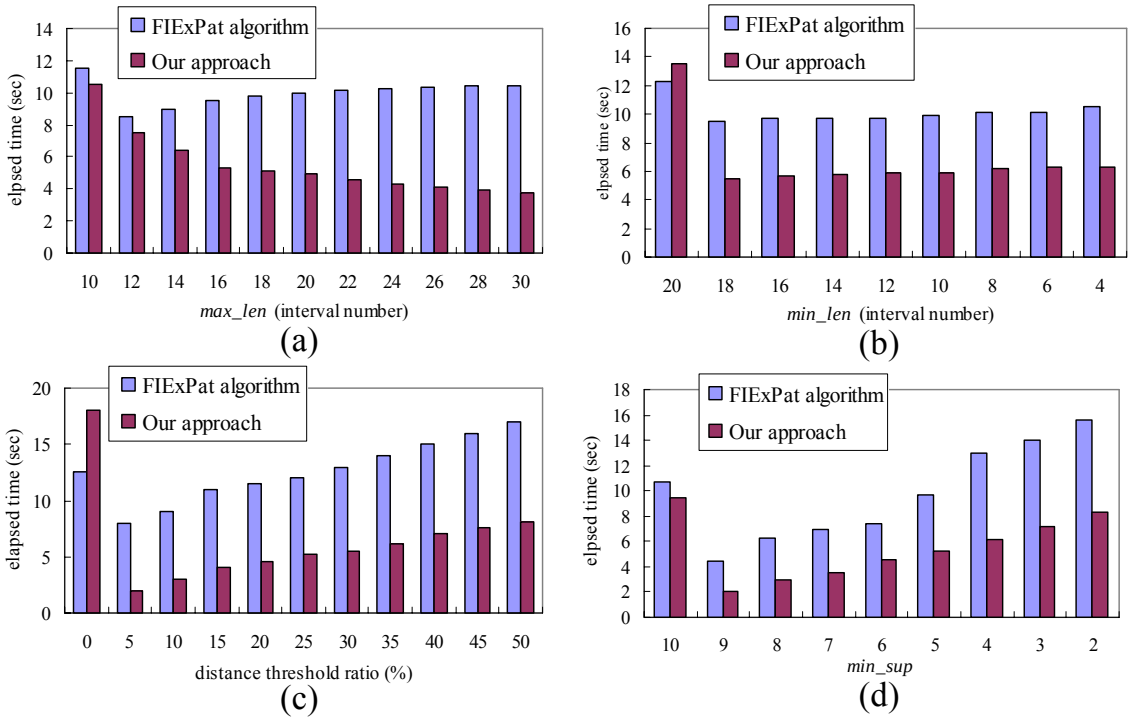
經由這些實驗結果的呈現，我們驗證此自動化音樂曲式分析技術可以達到很好的效果，因此可供相關的音樂分群以及音樂塑模等研究所採用。

表三：自動化輪旋曲式分析結果

No.	Composer	Title	Expert	Our Method
1	Bach	Concerto in E major Mov.3 Allegro Assai	ABACADAEA	ABACADAEA
2	Bach	Partita No.3 in E, BWV1006 Mov.3 Rondo	ABACADAEA	ABACADAEA
3	Beethoven	Sonata No.20 in G, Op.49 No.2	ABACA	ABACA
4	Beethoven	Piano Sonata No.10 in G, Op.14 No.2 Mov.3	ABACA	ABACA
5	Beethoven	Sonata Op.13 No.8	ABACA	ABACA
6	Brahms	Rondo nach Weber	ABACA	ABACA
7	Couperin	Soeur Monique: Rondeau for Clarinet	ABACADA	ABACADA
8	Couperin	Les Barricades Mysterieuses	ABACA	ABACA
9	Couperin	La Linotte Effarouchée	ABACADA	ABACADA
10	Couperin	Les Moissonneurs	ABACA	ABACA
11	Couperin	Le Tic-Toc-Choc	ABACA	ABACA
12	Daquin	Le Coucou	ABACA	ABACA
13	Haydn	Piano sonata No.50 in D Mov. Finale	ABACA	ABACA
14	Haydn	String Quartet in C major 'The Bird', Op.33 No.3 Mov.4 Rondo: Presto	ABACA	ABACA
15	Hummel	Strings Trio in G , WoO.4 Mov.4 Rondo	ABACA	ABACA
16	Mozart	Viennese Sonatinas, K.439b (No.1 in C) Mov. 4 Rondo	ABACA	ABACA
17	Mozart	Viennese Sonatinas, K.439b (No.4 in Bb) Mov.3 Rondo	ABACA	ABACA
18	Schubert	Piano Sonata No.17 in D, Op.53 D.850 Mov.4 Rondo	ABACA	ABACA
19	Schubert	Rondo in D major "Notre amitie est invariable", Op.138 D.608	ABACADAEA	ABACADAEA
20	Schumann	Novelette Op.21 No.5	ABACADAEA	ABACADAEA
21	Schumann	Nachtstücke op.23 (1839) II. Markiert und lebhaft	ABACA	ABACA
22	Rameau	Rondeau La Joyeuse	ABACA	ABACA
23	Rameau	Gigue en rondeau	ABACA	ABACA
24	Rameau	Tambourin	ABACA	ABACA
25	Rameau	Musette en Rondeau	ABACA	ABACA
26	Rameau	Les Tendres Plaintes	ABACA	ABACA
27	Rameau	La Follette	ABACA	ABACA
28	Rameau	Les Tourbillons	ABACA	ABACA
29	Rameau	Deuxième Gigue en Rondeau	ABACADA	ABACADA
30	Rameau	Sommeil: Rondeau tendre	ABACA	ABACA

（三）近似重複樣型擷取實驗成果

針對我們的演算法，我們設計了一系列的實驗來驗證我們的方法。比較的對像就是 Rolland 所提出的 FIExPat 演算法[5]。實驗結果如下:



圖十二：擷取近似重複樣型之效能比較

在圖十二(a)中，我們比較了樣型最大長度對效能的影響，我們的方法比 FIExPat 要好。而在圖十二(b)中，針對樣型最小長度變化對效能的影響，我們除了一開始需花較多的時間來建立索引外，其它的時間均較短。圖十二(c)及圖十二(d)則分別對於距離的設定及最少次數的設定來作效能的比較，我們發現我們的方法平均都優於 FIExPat。

參考文獻

[Baez92] Baeza-Yates, R., and G.H. Gonnet, "A New Approach to Text Searching," *Communications of the ACM*, 1992.

[Boye77] Boyer, R.S., and J.S. Moore, "A Fast String Searching Algorithm," *Communications of the ACM*, Vol. 20, October 1977.

[Camb01] Cambouropoulos, E., "The Local Boundary Detection Model (LBDM) and its Application in the Study of Expressive Timing," *Proc. International Computer Music Conference*, 2001.

[Chen04] Chen, H. C., C. H. Lin, and A. L. P. Chen, "Music Segmentation by Rhythmic Features and Melodic Shapes," *Proc. IEEE International Conference on Multimedia & Expo*, 2004.

[Clau00] Clausen, M., R. Engelbrecht, et al., "PROMS: A Web-based Tool for Searching in Polyphonic Music," *Proc. International Symposium on Music Information Retrieval*, 2000.

[Corm93] Corman, T.H., C.E. Leiseson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press: McGraw-Hill, 1993.

[Dove99] Dovey, M.J. "An Algorithm for Locating Polyphonic Phrases within a Polyphonic Musical Piece," *AISB Symposium on Musical Creativity*, 1999.

[Frib98] Friberg, A., R. Bresin, L. Fryden, and J. Sunberg, "Musical Punctuation

on the Microlevel: Automatic Identification and Performance of Small Melodic Units," *Journal of New Music Research*, 1998.

[Gusf97] Gusfield, D., *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, 1997.

[Huro95] Huron, D., "The Melodic Arch in Western Folksongs," *Computing in Musicology*, Volume10, 1995.

[Hsu01] Hsu, J. L., C. C. Liu, and A. L.P. Chen, "Discovering Non-trivial Repeating Patterns in Music Data," *IEEE Transactions on Multimedia*, Vol. 3, No. 3, 2001.

[Jaga00] Jagadish, H.V., N. Koudas, and D. Srivastava, "On Effective Multi-dimensional Indexing for Strings," *Proc. ACM SIGMOD Conference*, 2000.

[Kahv01] Kahveci, T., and A. Singh, "Efficient Index Structures for String Databases," *Proc. International Conference on Very Large Data Bases*, 2001.

[Lems00] Lemstrom, K., and S. Perttu, "SEMEX - An Efficient Music Retrieval Prototype," *Proc. International Symposium on Music Information Retrieval*, 2000.

[Liu99] Liu, C.C., J.L. Hsu, and A.L.P. Chen, "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval," *Proc. IEEE Conference on Multimedia Computing and Systems*, 1999.

[Mccr76] McCreight, E., "A Space-Economical Suffix Tree Construction Algorithm," *Journal of Association for Computing Machinery*, 1976.

[Nava01] Navarro, G., "A Guided Tour to Approximate String Matching," *ACM Computing Surveys*, 2001.

[Pei01] Pei, J., J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," *Proc. International Conference on Data Engineering*, 2001.

[Pien02] Pienimäk, A. "Indexing Music Databases Using Automatic Extraction of Frequent Phrases," *Proc.International Symposium on Music Information Retrieval*, 2002.

[Roll01] Rolland, P. Y., "FIExPat: Flexible Extraction of Sequential Patterns," *Proc. IEEE International Conference on Data Mining*, 2001.

[Shih01] Shih, H. H., S. S. Narayanan, and C. C. Jay Kuo, "Automatic Main Melody Extraction From MIDI Files with a Modified Lempel-Ziv Algorithm," *Proc. International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001.

[Wein73] Weiner, P., "Linear Pattern Matching Algorithms," *IEEE 14th Annual Symposium on Switching and Automata Theory*, 1973.

[Wu92] Wu, S., and U. Manber, "Fast Text Searching Allowing Errors,"

Communication of the ACM, 1992.

[Yana98] Yanase, T., "Phrase Based Feature Extraction for Musical Information Retrieval," *Proc. Communications, Computers and Signal Processing, IEEE Pacific Rim Conference*, 1999.

計畫成果自評

本計畫已經順利執行完畢，並且完成了多項與音樂服務相關技術之開發，希望能夠提供更有效率且更好的音樂服務品質。不同於過去對於大量音樂資料庫所提供之查詢技術，我們考慮的乃是在音樂頻道上提供使用者所需的查詢服務，在此串流環境之下，我們的方法結合了查詢結果共享之觀念，配合查詢分群、篩選技術與評估方式的整合，提出了一個能夠滿足使用者即時性需求的查詢索引架構以及查詢處理方法，透過實驗的驗證，我們可以確知該方法是確實有效而可行的。然而，這樣的音樂查詢只涵蓋了音樂內容，而沒有考慮到音樂架構的部分，為了能夠提供更進一步的查詢，我們進行了音樂分析的深入研究，包括了近似重複樣型的擷取以及自動化曲式的分析，近似重複樣型的擷取可應用於有音樂物件的主題的確認，我們的技術配合了索引架構以及篩選技術，得以更有效率地擷取出一個音樂物件的近似重複樣型，其成效在實驗中已經證明比過去的研究更佳，而且能夠配合使用者的需求找到其所需的近似重複樣型；配合樂理的基礎，我們也同時開發了曲式分析的技術，目前針對兩類最為常見的音樂曲式：輪旋曲及賦格曲，分別提出了兩種辨認的技術，其中關於主題擷取的部分便可以利用到前述之技術來達成，除了曲式的分析之外，我們可一併將該音樂物件所採用的作曲技巧加以辨認，可以提供更為豐富的音樂資料給使用者參考。近似重複樣型的擷取與曲式分析之成果都可以直接用於 MusiXML 的音樂資料模型，使得該資料模型可以包含更多的音樂內涵，並進一步提供更為高階的音樂查詢服務，包括了音樂架構的查詢或是音樂類型的查詢等。我們所提出的近似重複樣型擷取技術以及自動化曲式分析方法等兩篇論文皆已經被國際會議所接受，而串流環境下之音樂擷取技術以及近似重複樣型擷取技術的延伸版論文也已投往相關的國際期刊。

已發表之論文列表

[1] Liu, N.H., Y.H. Wu, and A.L.P. Chen, “An Efficient Approach to Extracting Approximate Repeating Patterns in Music Databases,” *Proc. International Conference on Database Systems for Advanced Applications*, 2005.

[2] Weng, P.H., and A.L.P. Chen, “Automatic Musical Form Analysis,” *Proc. International Conference on Digital Archive Technologies*, 2005.

已投稿之論文列表

[1] Chen, H.C, Y.H. Wu, and A.L.P. Chen, “Continuous Query Processing over Event Streams Based on Approximate Matching Mechanisms,” *submitted for journal publication*.

[2] Liu, N.H., Y.H. Wu, and A.L.P. Chen, “Identifying Prototypical Melodies by Extracting Approximate Repeating Patterns from Music Works,” *submitted for journal publication.*

附件二

可供推廣之研發成果資料表

☐ 可申請專利　☐ 可技術移轉　　日期：__年__月__日

國科會補助計畫	計畫名稱： 計畫主持人： 計畫編號：　　學門領域：
技術/創作名稱	
發明人/創作人	
技術說明	中文： （100~500 字）
	英文：
可利用之產業及可開發之產品	
技術特點	
推廣及運用的價值	

※ 1.每項研發成果請填寫一式二份，一份隨成果報告送繳本會，一份送　貴單位研發成果推廣單位（如技術移轉中心）。

※ **2.本項研發成果若尚未申請專利，請勿揭露可申請專利之主要內容。**

※ 3.本表若不敷使用，請自行影印使用。