混合樹狀結構之音樂資料多特徵索引

羅有隆
朝陽科技大學
資訊管理系副教授
yllo@cyut.edu.tw

王俊雄
朝陽科技大學
資訊管理系研究生
s9514612@cyut.edu.tw

摘要

在多媒體資料庫中如何有效率的管理音樂資料，近年來已獲得了高度的關注。現在大部分的研究工作，多利用音樂資料的特性萃取，例如：旋律、節拍、和弦等等，來建立索引，以加速音樂資料的搜尋。已有許多的研究報告指出，這些音樂資料特性可以轉換為字串或數值形式，再各別發展出有效率的索引架構，以幫助音樂資料的擷取。而這些研究報告中，多數只針對音樂資料之單一特徵索引所設計，而目前已被發表少數的多特徵索引技術，若不是處理查詢沒有彈性，就是必須佔用相當大的記憶體空間，限制了多特徵索引的擴充能力與實用性。在此篇研究報告中，我們將提出一混合樹狀結構的索引架構，以改善索引對記憶體的使用量。我們並經由實驗分析證實，所提出的方法，確實可以大幅降低音樂資料多特徵索引所需大量的記憶體空間，讓多特徵索引的應用，更為實用可行。

關鍵詞：多媒體資料庫、音樂資料庫、多特徵索引、內容擷取

1. 前言

通常多媒體的資料量都很大，媒體的特徵多，相對於純文字與數字的一般資料庫來說，多媒體資料庫是既龐大又難管理。所以早期多媒體資料庫未普遍時，有關多媒體資料庫存取與管理的研究也較少，而資料庫的研究也大多集中在一般文字與數字的資料庫中。而現今硬體和軟體技術發展快速，相對於管理和存取大量的多媒體資料愈來愈迫切。因此，如何為多媒體資料建立有效的搜尋方式，在多媒體資料庫的研究中，便成為一個重要的議題。較早之前的多媒體資料庫大多採用字串的關鍵字來查詢，例如影片名稱、主題曲、演員、作曲、演唱者……等等。然而這幾年有了相當多的探討，集中在以內容為主的多媒體資料擷取(content-based multimedia data retrieval)。不過在初期有關內容為主的擷取方式的議題又大多集中在文件(document)、圖像(image)、影片(video)上[4][5][6][7][12]。然而，聲音與音樂資料的擷取應用，卻較少受到重視。所幸近來在多媒體資料庫中如何有效的管理音樂資料，以提供使用者更好人性化的查詢服務，已獲得了較高度的注意[1][2][3][11][12][13][14][15][17][18][19][21][22]。目前音樂資料庫相關的研究議題，大多集中在音樂資料特徵的萃取與音樂檢索方面。而在音樂資料特徵萃取的研究中，大多集中在解決如何從音樂資料中萃取有意義的特徵，如主旋律(key melody)、節拍(rhythm)、和弦(chord)等部分，以用來提昇音樂資料檢索的效率[10][13][18][19]。另外，在音樂檢索方面，大多利用所萃取出的音樂特徵資料以建立索引，再利用這些索引來加速音樂資料查詢的搜尋[13][16]。

目前許多的研究指出，音樂資料特性可以轉化為符號或字串形式來表示，再根據這些字串表示方式，發展出有效率的字串索引(string indices)，以幫助音樂資料的檢索[8]。雖然也已有許多的研究報告為字串的比對，提供了有效率的解決方案，然而這種以字串比對為基礎，做為音樂資料擷取方式，仍然有許多可以改善的空間。例如：傳統有效的資料庫索引結構不少，但多數專為數值設計，並不全然適用於未定長度的字串比對。而專為字串所設計的索引架構，如要做為音樂資料的近似搜尋(approximate search)，或者擴展為多特徵的索引結構(multi-feature index structure)，到目前為止突破仍然有限，他們常需要大量的記憶體空間，且對於音樂資料特徵的多樣性，也大多缺乏擴充性(scalability)。

本篇論文的主要目的，在於提出一相對於現有技術可更節省記憶體使用空間的音樂多特徵索引結構，同時對各類查詢也能更為有彈性的處理。其成果，我們預期可以對音樂資料庫的管理與應用更為實用外，也希望音樂資料多特徵索引架構，可以應用在其它多媒體資料上。本文共有五節，內容概述如下：除第一節

的前言外，在第二節中我們介紹目前音樂資料的兩類索引形式；接著在第三節探討現有音樂資料的多特徵索引技術；而於第四節詳述我們所要發表的音樂資料混合樹狀結構多特徵索引；之後在第五節，我們設計了實驗以驗證所提出方法的效能。最後一節則為我們的結論與未來可能的研究方向。

2. 音樂資料的索引形式

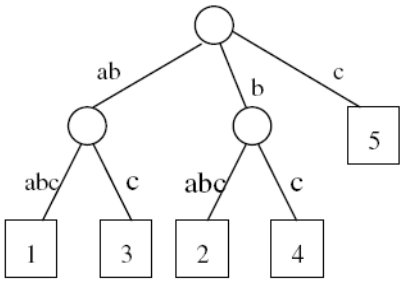
在這節中，我們對目前音樂資料的字串索引與數值索引兩類形式，做進一步的說明。

2.1 字串索引

現有音樂資料的字串索引應用[3][11]，多為Suffix Tree[20]的索引結構，其主要是將字串所有可能的字尾集合(set of suffixes)建立成索引，大多用於字串比對(string matching)，也利用此來幫助字串片段的快速檢索。由於近年來的研究，多將音樂資料特徵以字串型式表示，因此可以利用字串的相關索引結構來處理音樂資料檢索問題，因此Suffix Tree結構就受到較多的關注。在Suffix tree的結構有幾個主要的特性：

1. 長度為 *m* 字串建構出的 Suffix tree，會剛好有 *m* 個葉節點。
2. 從非葉節點分支出去的任兩個分支，不能標示相同的字元。
3. 字串中有 *n* 個不同的字元，根節點會產生 *n* 個分支。

假設現有一音樂字串為“ababc”，其所建構而成的 Suffix tree 索引，結果如下圖一。音樂的查詢樣本如果轉換成字串表示，透過在索引中字串的比對，就可以輕易的找尋到查詢的標的。不過以此種字串索引來做為音樂資料的查詢，其較缺乏擴充性。



圖一、字串為“ababc”的 Suffix tree

2.2 數值索引

音樂資料的數值索引應用，多為R-Tree與B-Tree索引結構。先將擷取的音樂資料字串轉換成唯一可表示此音樂資料的整數值[9][15]，再利用R-Tree或B-Tree來建立索引。舉例來說，假設可以呈現的音符(pitch)總類為*m*個，如：Do、Re、Mi、…。每個音符都可依序對應到 0、1、2、3、…、*m*-1 的整數值。如果每次選擇音樂旋律的*n*個連續音符來建立索引，如此連續*n*個音符為p_1、p_2、…、p_n，每個音符所對應的數值可以用公式(1)的*P(x)*表示，而*x*的範圍為 $0 \le x \le n-1$。利用音符的數值轉換函數*v(n)*，可以將連續*n*個音符字串轉換成唯一的整數值。

$$v(n) = \sum_{x=1}^{n} P(x) \times m^{x-1} \quad \text{.... 公式(1)}$$

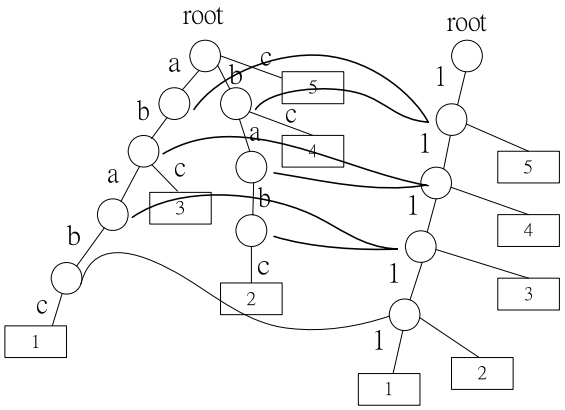
假設不同的音符總類有 10 個，用字串表示為‘a’、 ‘b’、 ‘c’、…、‘j’，所對映的數值為 0、1、2、…、9。如果有一音樂旋律片段為“badc”的 4 個音符，可將此一片段依上述函數轉換成數值：$v(4)=1*10^0+0*10^1+3*10^2+2*10^3=2301$。如此就可以把代表此音樂字串的數值，建立於現有的眾多數值索引中，同樣的查詢時，也必需將音樂樣本資料轉換為數值，再於索引中比對。此種以轉換為整數值來做查詢的方式，因必須設定音樂資料轉換的固定長度，對於查詢資料樣本的不定長度，顯然就較缺乏彈性。

3. 音樂資料多特徵索引之現有技術

目前國內外探討音樂資料庫索引方面的研究多集中在單特徵索引，其代表性的研究：在 1999 年有，Tseng的Key Melody Extraction與N-note Indexing[20]、Uitdenbogerd等的Melodic Matching Techniques[21]、以及Liu等的Approximate StringMatching Algorithm[14]。2000 年有，Chen等的Music Segments[3]。以及 2002 年Lo等的Numeric Index[15]。在多特徵索引方面，其相關研究報告較少，而較具代表的研究報告為，2000 年 Lee 等所發表的 Multi-Feature Index Structure[11]，以及 2003 年 Lo等的Multi-feature Numeric Index[17]，接下來我們介紹此兩現有的多徵索引技術。

3.1 Grid-Twin Suffix Trees

W. Lee與A. L. P. Chen所提出的Efficient Multi-Feature Index Structure for Music Data Retrieval [11]，是利用音樂資料的特性轉換為字串的表示方式，並以Suffix tree為基礎，發展出四種多特徵索引結構(multi-feature index structure)，以利音樂資料多特徵檢索。文裡所提出的四種索引結構中，以第四種Grid-Twin Suffix Tree索引結構效率最好。然而Grid-Twin Suffix Trees則是由第三種索引Twin Suffix Trees所發展出來。Twin Suffix Trees是將兩種特徵的Suffix trees給鏈結起來，以做為雙特徵的比對搜尋。以“$a_1b_1a_1b_1c_1$”為例，“ababc”代表旋律特徵，“11111”代表節拍特徵，則所建構之Twin Suffix Trees之多特徵索引，如圖二。



圖二、Twin Suffix Tree 索引架構

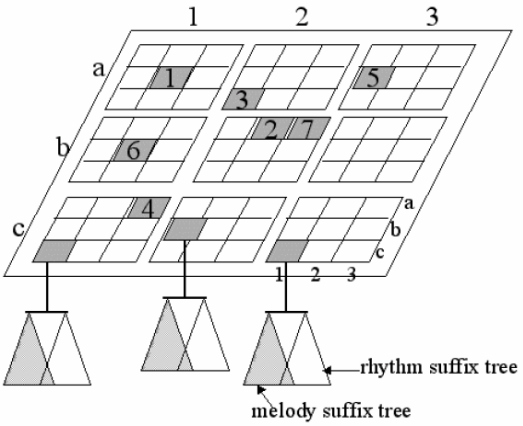
Grid-Twin Suffix Trees(GTST)則是利用設計好的函數，截取音樂特徵字串的前 n 個特徵樣本(或符號)，計算出相對應二維格狀空間的座標，而於各座標下再各連接一 Twin Suffix Tree，如此採部分數值座標比對，可加快音樂資料的搜尋。多特徵值的座標轉換函數如公式(2)所示：

$$P(x,y) = \sum_{i=1}^{n} \left(\frac{(Num_m)^n}{(Num_m)^i} M_i, \frac{(Num_r)^n}{(Num_r)^i} R_i \right)$$

….**公式(2)**

其中，$P(x,y)$為音樂旋律所對應到二維格狀空間坐標，Num_m與Num_r各為代表音符與節拍的特徵符號個數。M_i 與R_i分別為第i個音符與節拍所表示的數值。作者認為音符可以‘a’, ‘b’, ‘c’…等字元符號表示，節拍可以‘1’, ‘2’, ‘3’..字元符號表示，且每個音符與節拍皆可對映到0,1,2,3..等整數值。

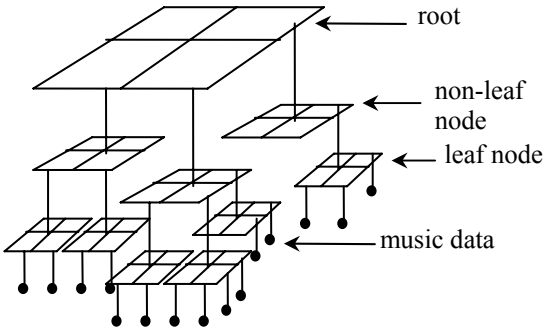
假設我們有一段音樂旋律的特徵字串為“$a_1b_2a_2c_1a_3$”，各取前面兩個特徵值，即n=2。如此‘a_1’可計算出對應的二維坐標，$P(x,y)_1$ = $(3^2/3^1*0, 3^2/3^1*0)$ = (0,0)。接下來取 ‘b_2’計算出對應的二維坐標，$P(x,y)_2$ = $(3^2/3^2*1, 3^2/3^2*1)$ = (1,1)。然後再將兩座標加總，即為音樂旋律“$a_1b_2a_2c_1a_3$”所對應到二維格狀空間的坐標位置，即(1,1)。GTST的索引結構呈現如圖三所示。而使用者在下達搜尋需求時，也是利用同一算式找到對應的座標值，並對底下的Twin-Suffix trees做後續比對的動作，如此便可減少前面n個特徵值的比較。



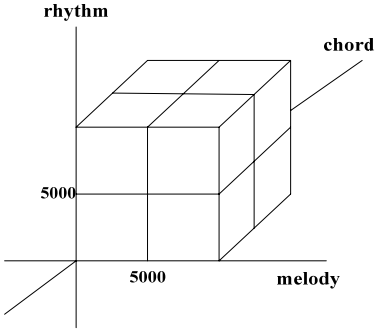
圖三、Grid-Twin Suffix Tree索引架構(資料參考來源[11])

3.2 Multi-Feature Numeric Index

Lo等的Multi-Feature Numeric Index [17]，是將音樂數值索引的概念[15]，應用到多特徵索引上，索引結構裡的節點，每一座標，代表著一個特徵值，依多特徵的共同中心座標，將每個特徵值依大於或小於中心值劃分兩個分支度，如此結合多特徵的一個節點將有多個的分支度，假設有d個特徵，則分支度為 2^d。其二維特徵索引的範例架構如圖四所示，圖五則是一個三特徵的索引節點示意圖。



圖四、 音樂資料二維特徵索引結構的範例



圖五、三特徵的索引節點範例

4. 混合樹狀結構的音樂資料多特徵索引

現有的兩種音樂資料多特徵索引技術，Grid-Twin Suffix Tree[11]是轉換特定長度的特徵值，再建構於多維的空間索引之中，此外超過此特定長度特徵的音樂字串，則是儲存於此多維空間下的Twin Suffix Tree索引結構中。這樣的索引架構，可以提高搜尋音樂資料的速度，但是當每個特徵的特徵值種類較多(如音域高低的範圍)，或是索引採用較多的特徵數，則多維空間的索引，將占用非常大的記憶體空間，甚至易形成稀疏矩陣的浪費情形。例如採用 3 個特徵，而每個特徵值各有 20 個種類，建立索引時各特徵取 2 個樣本(或特徵符號)，則其所建立的二維空間將有$(20\times20\times20)^2$ =6 千 4 百萬個格狀空間(Grid)入口。而Multi-Feature Numeric Index[17]是將音樂資料轉換為數值後，建立起多特徵索引。如前面所介紹到的，音樂資料轉換成數值必須設定特定的轉換長度，當音樂的查詢範本(Query By Example, QBE)與所建索引的特徵資料長度相同時，將可以被快速的搜尋到，但若不是相同的長度，則將增加查詢的複雜程度，且搜尋時間是倍增計，相當的缺乏彈性。

而在此篇的研究報告中，我們將針對較具查詢彈性的 Grid-Twin Suffix Tree 索引結構來做改良，結合 Multi-Feature Numeric Index 的原理，以混合樹狀索引結構，解決現有技術的缺點。我們所提出的索引架構，將可以比 Grid-Twin Suffix Tree 節省大量的記憶體空間。

4.1 混合樹狀索引架構

我們將提出的音樂資料多特徵索引，結合了數值樹狀索引與字串樹狀索引的混合多特徵索引架構(Hybrid Multi-Feature Index)，是以一多特徵數值索引(Multi-Feature Numeric Index)取代了 Grid-Twin Suffix Tree 之二維格狀空間的部分，以當做進入字串比對前的快速導引。而原 Grid-Twin Suffix Tree 裡各二維格狀空間座標下所連接的 Twin Suffix Tree，在混合樹狀多特徵索引架構裡，則是連接於葉節點的下面。如此，建構音樂資料的混合樹狀結構多特徵索引，將由下面幾個步驟來完成：

Step 1. 對於準備建立於索引中的音樂資料(或音樂特徵字串)，假設共有 d 個音樂特徵，從各個特徵將各取出長度為 n 的特徵值樣本(特徵值取樣長度)，將其轉換為多維空間座標，所使用的計算公式，是我們重新設計的多特徵數值轉換函式，如公式(3)所示。

$$P(x_1,..,x_d)=\sum_{i=1}^{n}(F_1(i)\cdot N_1^{n-i},..,F_d(i)\cdot N_d^{n-i}) \quad \text{....公式(3)}$$

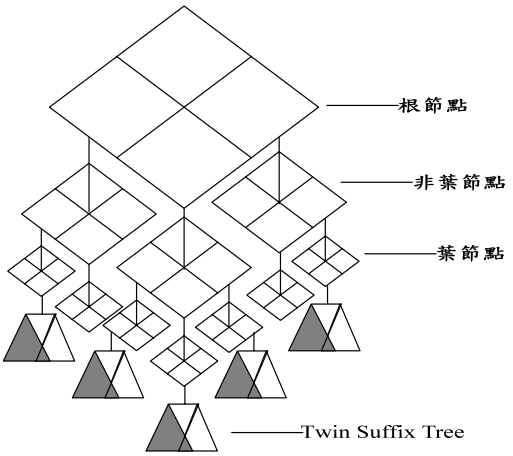
其中，$P(x_1, .., x_d)$為所轉換出之d維空間座標；$F_1(i), .., F_d(i)$代表第 1 個至第d個特徵分別在音樂字串資料中的第i個特徵樣本值；而$N_1, .., N_d$則是各特徵的種類，如旋律有Do、Re、Mi、…或節拍有½拍、1 拍、2 拍等等的變化種類。將不同種組合的特徵值帶進公式計算，每一個都將得到代表其唯一的多維空間座標。

Step 2. 將所轉換出來的多維空間座標，插入多維特徵數值索引(Multi-Feature Numeric Index)中。假設特徵數為d的話，此索引中的每個非葉節點，都將會有 2^d個分支度，以及一中心座標，各特徵的中心座標值為其下所有子節點的相同特徵座標值的平均。以二個特徵為例，$2^d=4$，假設中心座標為(x_c, y_c)，則其 4 個分支度的資料條件分別為$(\geqq x_c, \geqq y_c)$、$(< x_c, \geqq y_c)$、$(\geqq x_c, < y_c)$、與$(< x_c, < y_c)$。三特徵或以上依此類推。我們也要特別註明，為了保持樹狀結構的平衡，我們採用R-tree的特性，每個非葉節點必須維持半滿的條件，如此，節點的中心座標可能因新節點的加入而重新計算，以及部分節點位置重新的調整，以讓各節點以下的子節點數盡量達到均衡。

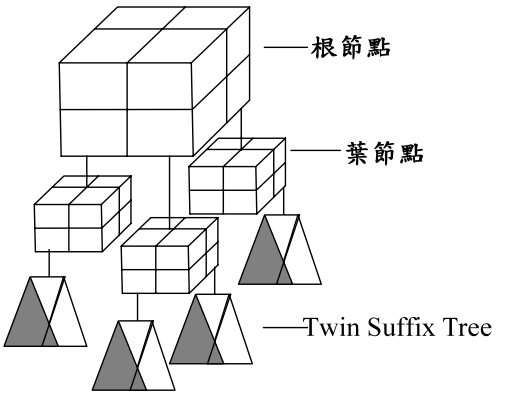
Step 3. 跟在音樂資料各n個特徵符號之後，所還有的音樂資料特徵字串，則依Twin Suffix Tree [11]的建構原則，一一的插

入其所屬葉節點下的Twin Suffix Tree字串索引中。

圖六是一個混合樹狀結構的兩個特徵索引範例。同樣也可以推衍到混合樹狀結構之三特徵索引，如圖七所示。新的多特徵索引架構結合了數值索引與字串索引，將可比Multi-Feature Numeric Index 多了查詢的彈性，也可比Grid-Twin Suffix Tree節省記憶體空間。



圖六、混合樹狀結構之兩個特徵索引架構



圖七、混合樹狀結構之三個特徵索引架構

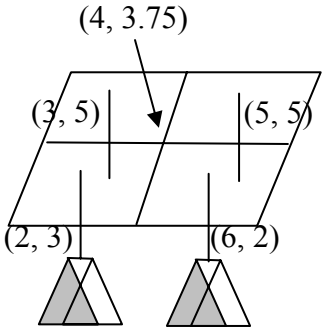
4.2 索引建構範例

由於混合式樹狀多特徵索引與Grid-Twin Suffix Tree僅差別在第一層我們用了樹狀結構來做多特徵索引，而Grid-Twin Suffix Tree用二維陣列。其後之下所連接的Twin Suffix Tree則都是相同。在此節中，我們將針對建立多特徵索引樹的部分，以範例來說明。至於Twin Suffix Tree的建構，為了節省篇幅，則請參閱[11]裡的詳細說明。我們以兩個特徵的音樂範例，來說明如何建立混合式樹狀多特徵索引之數值索引層的部分。假設不同音符種類有3個，‘a’、‘b’、‘c’其對應的數值依序為0、1、2。節拍種類‘1’、‘2’、‘3’也依序對應到0、1、2數值。如果現有音樂資料，包含著旋律與節拍之特徵字串：“$a_2c_1a_3b_2$”、“$b_2a_3b_1a_3$”、“$c_1a_3b_2b_2$”、“$b_2c_3b_1b_2$”與“$a_1b_2c_1b_2$”。若建立索引時每個特徵值取樣長度n=2，將音符與節拍各別之前兩特徵值帶入公式(3)，其轉換結果如表一所示：

表一、特徵值轉換成數值

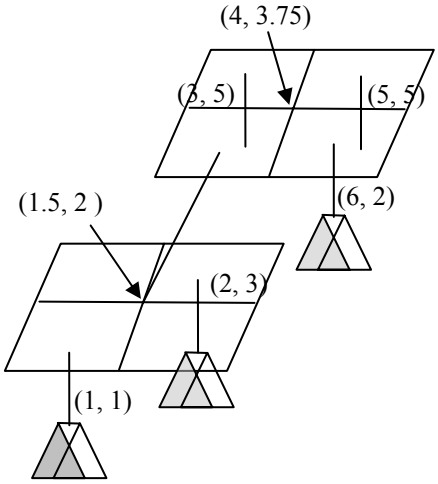
音樂片段	數值轉換過程	$P(x_1, x_2)$
$\underline{\mathbf{a}_2\mathbf{c}_1}a_3b_2$	$(0*3^1, 1*3^1) + (2*3^0, 0*3^0)$	(2,3)
$\underline{\mathbf{b}_2\mathbf{a}_3}b_1a_3$	$(1*3^1, 1*3^1) + (0*3^0, 2*3^0)$	(3,5)
$\underline{\mathbf{c}_1\mathbf{a}_3}b_2b_2$	$(2*3^1, 0*3^1) + (0*3^0, 2*3^0)$	(6,2)
$\underline{\mathbf{b}_2\mathbf{c}_3}b_1b_2$	$(1*3^1, 1*3^1) + (2*3^0, 2*3^0)$	(5,5)
$\underline{\mathbf{a}_1\mathbf{b}_2}c_1b_2$	$(0*3^1, 0*3^1) + (1*3^0, 1*3^0)$	(1,1)

由於是建構兩個特徵的數值索引，每個非葉節點的分支度為4。我們首先將表一的前4個座標新增入索引中，所產生的第一個節點，其中心座標為(4, 3.75)，表示於圖八，四個座標則剛好落於節點的四個區域性中，以下則連接著其所屬的Twin Suffix Tree。



圖八、數值索引範例

接下來我們將表一的第5個座標(1, 1)新增入索引樹，必須從第1個節點的左下區域插入，造成其下的連結必須新增一個節點，此新增節點由於目前只有兩個分支連到葉節點—(2, 3)與(1, 1)，此節點的中心座標為(1.5, 2)，目前索引結構如圖九所示。



圖九、新增節點範例

5. 實驗驗證

由於在前面我們已經有提到Multi-Feature Numeric Index[17]受限於查詢範例的長度，查詢範例必須有與建立索引時相同音樂資料字串的長度，才能順利與快速的搜尋與比對，如此較無彈性，使得實用性較不理想。而Grid-Twin Suffix Tree[11](GTST)與我們此研究報告所新提出的混合多特徵索引架構(Hybrid Multi-Feature Index，以下簡寫為HMFI)，則是對查詢沒有特別的偏好與限制。因此，我們對多特徵索引技術所需記憶體的效能分析，將只針對GTST與HMFI來做比較。又因為GTST與HMFI只差別於索引入口的導引，HMFI採用多特徵數值索引，而GTST採用二維陣列空間，而兩者下面所連接的Twin Suffix Tree卻又是完全一樣。因此，我們進一步只要比較多特徵樹狀索引與二維陣列空間所佔用記憶體的大小，就可以評判出HMFI是否比GTST更為節省記憶體的使用。

在考量 GTST 的實驗參數中，只要確定了建立索引所要使用到的特徵有幾種，而每一特徵的特徵值有多少種的變化，以及每個特徵將取用多長的特徵值樣本來建索引，如此，其二維空間陣列的大小就可以下列公式(4)求得：

$$M_{GTST}(d, n) = (N_1 \times N_2 \times \ldots \times N_d)^n \times PointerSize \quad \ldots\text{公式(4)}$$

其中N_1、N_2、N_d分別代表每個特徵值的變化種類；d為特徵數；n為每一特徵值樣本長度；*PointerSize*為向下連接到Twin Suffix Tree的鏈結(4Bytes)。

至於HMFI需佔用多少的記憶體空間，最主要是與資料庫的大小有關，當資料庫大時，節點數變多，索引結構也跟著增大，但是多特徵樹狀索引中的每個非葉節點，除了中心座標外，就只是連到其子節點或Twin Suffix Tree的鏈結，而其鏈結數(或分支度)為 2^d。因此，只要知道樹狀索引中有多少節點，即可輕易估計需佔用多少的記憶體空間。

為了驗證所提出的混合多特徵索引架構可以節省記憶體空間，從將分別從下列四個因素來探討：

一、特徵數的多寡(d)對索引的影響
二、建索引時每個特徵值的取樣個數(n)對索引的影響
三、特徵值的變化種類(N_i)對索引的影響
四、資料庫的大小對索引的影響

我們的實驗參數設定範圍如表二所示：

表二、實驗參數

變數	值的範圍
特徵數(d)	2 ~ 5
特徵值取樣長度(n)	2 ~ 5
特徵值的種類(N_i)	10 ~ 25
資料庫的大小	2~60 萬筆

變化各種不同參數的實驗結果，分別討論於下列各節。

5.1 特徵數對所需記憶體空間的影響

在這個實驗裡，我們要探討特徵數的多寡，對GTST與HMFI所需佔用記憶體的量有多大的差異。我們實驗的音樂資料有 10 萬筆，並設定每個特徵值的平均種類(N_i)各有 20 個，而特徵數(d)由 2 增至 5 個，每個特徵取樣做索引的長度(n)有 2 與 3 個的兩組實驗。我們再次說明，因GTST與HMFI兩索引結構下半部分連接的Twin Suffix Tree完全相同，因此我們只做GTST索引的二維空間與HMFI的多特徵數值所引所需記憶體空間的比較，實驗結果統計於表三與表四。

表三是建構索引時特徵值取樣長度為 2，特徵數由 2 增至 5 的結果，GTST 與 HMFI 所需的記憶體空間皆是隨著特徵數的增加而增加，特別是 GTST 所需的記憶體量增加非常的快速，例如特徵數為 4 時，它已需要超過 97GB

的記憶體空間。反觀 HMFI，所需的記憶體成長的較為緩和，甚至到特徵數 5 時，仍只需要 14.11MB 的記憶體空間而已，這是受到特徵數增加，使得每個節點的分支度增加的影響。此外，必須特別說明的是，在特徵數為 2 時，HMFI 比 GTST 佔用更多的記憶體空間，這是因為此時 GTST 的二維陣列空間還不大，但 HMFI 卻已經必須有為量不少的節點數來容納 10 萬筆資料所建立的索引。此情況在特徵數增加、特徵值取樣長度增加、或者每一特徵值的種類增加時，HMFI 在記憶體的佔用量上，將會迅速的優於 GTST。

再繼續探討表四，特徵值取樣長度為 3，GTST 所需的記憶體空間，幾乎以指數倍數成長，這可從公式(4)了解其原因。但是再觀察 HMFI，整組的數據與表三中完全一樣，這是因為，在這個實驗條件下，會影響 HMFI 的只有特徵數，而特徵值取樣長度不論為多少，所計算出來的，都只是一個數值座標而已。由此我們初步可以判斷，在特徵數與特徵值取樣長度的擴充能力上 HMFI 是優於 GTST 的。

表三、特徵值取樣長度為 2 的記憶體需求(單位：MB)

特徵數	GTST	HMFI
2	0.61	2.29
3	244.14	4.20
4	97656.25	7.63
5	39062500	14.11

表四、特徵值取樣長度為 3 的記憶體需求(單位：MB)

特徵數	GTST	HMFI
2	244.14	2.29
3	1953125.00	4.20
4	15625000000.00	7.63
5	125000000000000.00	14.11

5.2 特徵值取樣長度對所需記憶體空間的影響

雖然由上一節的實驗討論中，我們了解到，徵值取樣長度似乎並不影響 HMFI 所需的記憶體空間。在這一節，我們還是更進一步的做實驗來驗證，並探討特徵值取樣長度對 GTST 的影響如何。我們實驗的音樂資料同樣有 10 萬筆，而音樂特徵數，我們分別取 2 個與 3 個來做實驗。2 個的音樂特徵，我們用音符與節拍，音符我們從高音、中音、低音中取最常用的連續 20 個音，也就是音符的種類為 20。此外節拍有 1/8、1/4、1/2、3/8、3/4、1½、1¼、1¾、2、…等等，我們也取最常用的 15 個，讓節拍種類為 15。而當採 3 個特徵時，我們多增加了前後音差的特徵，因為它較容易表示與被了解，一般使用者在輸入查詢範例時(Query By Example)，可能會有起音的高低不同，或者說 key 的升降可因人而異，但整首旋律每個音的前後音差，卻是不會隨著演唱者起音(key)的不同而有所變化。因考量我們有 20 個音符種類，若最高音後接著一個最低音，或者最低音後接著一個最高音，則其音差值分別為-19 與 19，如此，其前後音差的範圍值為 -19~19，總共有 39 種的變化，也就是音差種類有 39 種。而在建立索引時，所測試變化特徵值的取樣長度，從 2 個到 5 個特徵值。針對 2 個特徵與 3 個特徵索引的實驗結果，分別列於表五與表六。

由於 HMFI 會受特徵數的影響，但對特徵值的取樣個數無關，以至於 HMFI 對記憶體的需求在表五是固定值 2.29MB，而在表六裡也是沒有變化的固定值 4.20MB。反觀 GTST 對特徵數與特徵值取樣的長度，卻是相當的敏感，對記憶體的需求，變動幅度非常的大。甚至在 2 個特徵時，特徵值取樣長度 4 或以上，或者 3 個特徵時，特徵值取樣長度 3 或以上，其所需的記憶體空間，必須以 Tera Byte 為單位，已經超過目前一般電腦所配備的記憶體(RAM)容量所能負擔。如此，在實際的應用上，就有其困難度。

表五、使用音符與節拍兩種特徵(單位：MB)

特徵值取樣個數	GTST	HMFI
2	2.32	2.29
3	1810.27	2.29
4	1412012.33	2.29
5	1101369616.70	2.29

表六、使用音符、節拍與音差三種特徵(單位：MB)

特徵值取樣個數	GTST	HMFI
2	522.19	4.20
3	6109669.00	4.20
4	71500000000.00	4.20
5	836000000000000.00	4.20

5.3 特徵值種類對所需記憶體空間的影響

在這節的實驗，我們要探討特徵值的種類多寡，對兩索引技術的影響。我們實驗的音樂資料同樣有 10 萬筆，而音樂特徵我們取 3 種，建索引特徵值的樣本長度，則是取 2。為了簡化實驗，三個特徵的各別特徵值平均種類，我們統一設定為 10~30 個。實驗結果則呈現於表七。再次的，特徵值種類對 HMFI 沒有任何影響，不管多少的特徵值種類，都只是取其所需建索引的樣本數(此實驗為 2)，轉換出一個座標值帶入索引，因此表七中 HMFI 所需的記憶體維持不變的值。反觀 GTST 所需的記憶體空間，卻是隨著平均特徵值種類的增加而呈數倍的成長。此外，我們也要說明，當平均特徵值種類為 10 時，HMFI 略遜於 GTST，那是因為，對 GTST 來說，平均特徵值種類較少，為其節省不少的記憶體空間，而實際的應用上，特徵值的種類卻往往比 10 還多。以擴充能力來說，建索引時，平均特徵值種類的增加、特徵數的增加或特徵值取樣的增加，都嚴重的影響 GTST 擴充的能力與實用性。相對的 HMFI 所受的影響較小，有著較好的特徵擴充能力。

表七、3 種特徵、取樣長度為 2 (單位：MB)

各特徵值平均種類數	GTST	HMFI
10	3.81	4.20
15	43.45	4.20
20	244.14	4.20
25	931.32	4.20
30	2780.91	4.20

5.4 資料庫大小對所需記憶體空間的影響

前面三個實驗皆在檢驗特徵條件的變化，對兩索引結構的影響，以及所需記憶體的佔用。而於此節，我們將以資料庫的大小來檢測索引對資料儲存的擴充能力。我們以較前幾個實驗還大的資料庫(10~60 萬筆資料)來檢測 3 特徵的索引，而此三個特徵採與 5.2 節一樣，音符有 20 種類、節拍有 15 種類、以及音差有 39 種類。此外，建索引每一特徵值所取的樣本長度(n)為 2。最後的實驗結果統計於表八。

由於這個實驗的特徵數、特徵值種類與特徵值取樣長度都固定，使得 GTST 有著固定大小的記憶體佔用量，那是因為資料庫大小，並不影響 GTST 二維格狀空間的大小。所影響到的，只是在下層 Twin Suffix Tree 所儲存的資料量會增加。而此 Twin Suffix Tree 的資料增加，同樣情形也會發生在 HMFI 中。至於 HMFI 是樹狀結構，其多特徵數值索引所需的節點數，會隨著資料庫的增大而增加，也就是表八中，其所需的記憶體空間是持續緩慢的在成長，但即使在 60 萬筆資料時，其所需的記憶體空間，仍然遠小於 GTST 的需要量。如此，HMFI 是遠優異於 GTST 的，也就是在我們現有的音樂資料庫應用上，若適當的規劃建立索引的條件，則採用 HMFI 可以節省大量的記憶體空間。且較大型的資料庫，為了能夠更精確與快速的查詢比對出使用者的需求，增加特徵數與適度的特徵值樣本長度，是有其必須性與實用性。如此，則更能凸顯 HMFI 的優異性。

表八、資料庫筆數大小的影響採 3 個特徵 (單位：MB)

資料筆數(萬)	GTST	HMFI
10	522.19	4.2
20	522.19	8.39
30	522.19	12.59
40	522.19	16.78
50	522.19	20.98
60	522.19	25.18

6. 結論與未來研究方向

音樂資料可以萃取出許多的特徵來幫助使用者查詢的搜尋比對，但是音樂資料多特徵索引的研究報告並不多，許多的研究只將各種特徵各別建立專屬的單特徵索引，如此要利用多特徵以幫助更精確的搜尋，效率將非常的有限。而現有少數的音樂資料多特徵索引，不是對查詢處理不夠彈性，就是索引結構需佔用大量的記憶體空間，使得實際的應用受到了限制。本研究報告提出了一混合樹狀多特徵索引結構HMFI (Hybrid Multi-Feature Index)，它結合了Multi-Feature Numeric Index [17]與Twin Suffix Tree [11]兩種索引，形成二層式的樹狀索引結構，上層為數值索引，下層則為字串索引，它改善了GTST(Grid-Twin Suffix Tree)[11]的二維格狀空間所需佔用大量記憶體空間的缺點。同時也經由我們的實驗證實，HMFI對於特徵個數、特徵值種類、以及建立索引時的特徵值取樣長度，具有相當好的擴充能力。 反觀GTST，就沒有那麼理想了。

而未來我們的研究將有兩個方向，其一為 GTST 的二維空間陣列，若形成浪費記憶體空間的稀疏矩陣，是否能將其壓縮，而不破壞原格狀區域對 Twin Suffix Tree 的一對一連結。其次，我們將進一步以改善 HMFI 與 GTST 往下連結的 Twin Suffix Tree 為目標，因為當特徵數增加時，Twin Suffix Tree 中跨索引樹間的結構將變得相當的複雜，也會相當程度的影響索引所佔用的記憶體空間，以及影響查詢的比對速度。同時，我們也期望所提出的多特徵索引架構，不僅只能應用在音樂資料上，凡具有多特徵性質的多媒體資料，也都能有其應用的價值。

參考文獻

[1] S. Blackburn and D. DeRoure, "A Tool for Content-based Navigation of Music," *In Proc. Of ACM Multimedia*, Pages 361-368, 1998.

[2] James C.C. Chen and Arbee L.P. Chen, "Query by Rhythm An Approach for Song Retrieval in Music Databases," *In Proc. Of Int'l Workshop on Research Issues in Data Engineering*, Pages 139-146, 1998.

[3] Arbee L.P. Chen, M. Chang, J. Chen, J.L. Hsu, C.H. Hsu, and Spot Y.S. Hua, "Query by Music Segments: An Efficient Approach for Song Retrieval," *In Proc. of IEEE Int'l Conf. on Multimedia and Expro*, 2000.

[4] G. Davenport, T.A. Smith, and N. Pincever, "Cinematic Primitives for Multimedia," *IEEE Computer Graphics & Applications,* Pages 67-74, July 1991.

[5] Y.F. Day, S. Pagtas, M. Iino, A. Khokhar, and A. Ghafoor, "Object-Oriented Conceptual Modeling of Video Data" *In Proc. Of IEEE Data Engineering*, Pages 401-408,1995.

[6] E.A. El-Kwae and M.R. Kabuka, "Efficient Content-Based Indexing of Large Image Databases," *ACM Trans. On Information Systems*, Vol. 18, No. 2, Pages 171-210, April 2000.

[7] Shen-Tat Goh and Kian-Lee Tan, "MOSAIC: A Fast Multi-Feature Image Retrieval System," *Data & Knowledge Engineering 33*, Pages 219-239, 2000.

[8] J.L. Hsu, C.C. Liu, and Arbee L.P. Chen, "Efficient Repeating Patterrn Finding in Music Databases," *In Proc. of ACM Int'l Conf. on Information and Knowledge Management*, 1998.

[9] H.V. Jagadish, N. Koudas, and D. Srivastava, "On Effective Multi-Dimensional Indexing for Strings," *In Proc. of ACM SIGMOD*, Pages 403-414, May 2000.

[10] C. L. Krumhansl, "Cognitive Foundations of Musical Pitch," *Oxford University Press*, New York, 1990.

[11] W. Lee and A.L.P. Chen, "Efficient Multi-Feature Index Structures for Music Data Retrieval," *In Proc. Of SPIE Conf. on Storage and Retrieval for Image and Video Database*, 2000.

[12] Chia-Han Lin and Arbee L. P. Chen, "Indexing and Matching Multiple-Attribute Strings for Efficient Multimedia Query Processing," *IEEE Transactions On Multimedia*, Vol. 8, No. 2, April 2006.

[13] C.C. Liu, J.L. Hsu, and Arbee L.P. Chen, "Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases," *In Proc. of IEEE Data Engineering*, Pages 14-21, 1999.

[14] C.C. Liu, J.L. Hsu, and Arbee L.P. Chen, "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval," *In Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems*, Pages 451-456, 1999.

[15] Yu-lung Lo and Shiou-jiuan Chen, "The Numeric Indexing For Music Data," *Proceedings of the IEEE 22nd Int'l Conference on Distributed Computing Systems (ICDCS'2002) Workshops – the 4th Int'l Workshop on Multimedia Network Systems and Applications (MNSA'2002)*, Vienna, Austria, July, Pages 258-263, 2002.

[16] Yu-lung Lo and Shiou-jiuan Chen, "Numeric Indexing Approach for Music Database Retrieval," *Journal of Chaoyang University of Technology*, ISSN 1026-244X, Vol. 2, Pages 141-163, June 2002.

[17] Yu-lung Lo and Shiou-jiuan Chen, "Multi-feature Indexing For Music Data," *IEEE 23nd International Conference on Distributed Computing Systems (ICDCS'2003) Workshops – the 5th International Workshop on Multimedia Network Systems and Applications (MNSA'2003)*, Providence, Rhode Island, USA, Pages 654-659, May 19-22, 2003.

[18] Yu-lung Lo, Ho-cheng Yu, and Mei-chin Fan, "Efficient Non-trivial Repeating Pattern Discovering in Music Databses," *Tamsui Oxford Journal of Mathematical Sciences*, Vol. 17, No. 2, Pages 163-187, Nov. 2001.

[19] Yu-lung Lo and Wen-Lin Lee, "Linear Time for Discovering Non-trivial Repeating Patterns in Music Databases," *the 2004 IEEE International Conference on Mulitmedia and Expro (ICME)*, Taipei, Taiwan, June 27-30, 2004.

[20] E. McCreight, "A Space-Economical Suffix Tree Construction Algorithm," *Journal of Association for Computing Machinery*, Pages 262-272, 1976.

[21] Y.H. Tseng, "Content-Based Retrieval for Music Collections," *In Proc. of ACM SIGIR'99*, Pages 176-182, 1999

[22] A.L. Uitdenbogerd and J. Zobel, "Melodic Matching Techniques for Large Music Databases," *In Proc. of ACM Multimedia*, Pages 57-66, 1999.